В. М. Глушков, Е. Л. Ющенко

Вычислительная машина „Киев"

Математическое описание

Государственное издательство технической литературы УССР

Киев · 1962

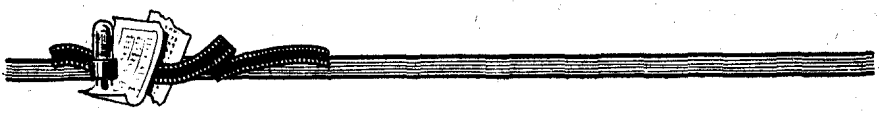
Настоящая книга посвящена описанию разработанной в ВЦ АН УССР универсальной вычислительной машины «Киев».

В книге приводятся обоснование выбора математических параметров машины, описание набора операций, стандартных и тестовых программ, а также излагаются вопросы автоматического программирования, включая разработанный в ВЦ АН УССР специальный алгоритмический язык.

Книга рассчитана на инженеров, научных работников, студентов и аспирантов, работающих в области вычислительной математики и вычислительной техники, а также на специалистов, желающих использовать в своей работе электронные вычислительные машины.

Рецензент
д-р физ-мат. наук проф. *Л. А. Калужнин*

Редакция литературы
по вопросам энергетики, радио и телевидения
Заведующий редакцией
инж. *М. Г. Писаренко*



ПРЕДИСЛОВИЕ

Работы по созданию универсальной электронной вычислительной машины «Киев» были начаты в лаборатории вычислительной техники Института математики АН УССР по инициативе акад. Б. В. Гнеденко коллективом, разработавшим в свое время под руководством акад. С. А. Лебедева первую в Европе электронную вычислительную машину МЭСМ. Совместно со старшими научными сотрудниками Л. Н. Дашевским и Е. Л. Ющенко Б. В. Гнеденко осуществлял руководство начальной стадией разработки машины. На заключительных этапах разработкой машины «Киев» руководили акад. АН УССР В. М. Глушков, Л. Н. Дашевский и Е. Л. Ющенко. Завершающие работы над машиной «Киев» (техническое проектирование, сборка и наладка) были произведены в Вычислительном центре АН УССР, созданном в декабре 1957 г. [4, 6].

В разработке, постройке и наладке машины участвовали многие научные работники и инженеры Вычислительного центра АН УССР. Важные самостоятельные работы были выполнены С. Б. Погребинским, Е. А. Шкабарой, а также Л. М. Абалышниковой, А. И. Кондалевым, В. В. Крайницким, В. Д. Лосевым, А. А. Барабановым, Л. П. Быстровой, В. И. Дворцыным, З. С. Зориной, А. Я. Зубатенко, Л. Н. Иваненко, А. А. Летичевским, В. С. Моторной и др. В начальный период в работе по выбору математических параметров машины приняли участие старшие научные сотрудники Института математики В. С. Королюк и И. Б. Погребыский.

При разработке машины «Киев» ставилась задача создать достаточно мощную вычислительную машину, которая могла бы войти в состав основного оборудования Вычислительного центра АН УССР. Этой разработкой коллектив откликнулся на постановление XX съезда КПСС о необходимости широкого развития электронной вычислительной техники.

Решение поставленной задачи потребовало прежде всего нахождения разумного компромисса между быстродействием, объемом памяти и удобствами программирования, с одной стороны, и надежностью машины — с другой.

В основу построения машины был положен асинхронный принцип, особенно удобный как с точки зрения бо́льшей простоты комплексной отладки, так и с точки зрения возможности последовательной (устройство за устройством) модернизации машины. Кроме того, этот принцип открывает большие возможности для использования не только отдельных ячеек, но и целых устройств машины при создании специализированных вычислительных и управляющих машин.

Следует отметить, что конструкция машины подвергалась и продолжает подвергаться модернизации не только в процессе комплексной отладки, но и после пуска ее в эксплуатацию. Поэтому описание машины в книге соответствует ее состоянию на 1 января 1960 г. Некоторые из произведенных или планируемых переделок описаны в монографии.

В книге описаны математические принципы, реализованные в машине «Киев», система команд, стандартные и тестовые программы, а также система автоматического программирования, основанная на разработанном в Вычислительном центре АН УССР алгоритмическом языке, применяемая в качестве входного языка программирующих програм и на других машинах.

Первая и вторая главы книги написаны В. М. Глушковым, остальные главы — Е. Л. Ющенко.

Отзывы и пожелания по этой книге просим направлять по адресу: Киев, 4, Пушкинская, 28, Гостехиздат УССР.

ГЛАВА I

ОСНОВНЫЕ ХАРАКТЕРИСТИКИ МАШИНЫ «КИЕВ»

ОБЩИЕ ДАННЫЕ

Электронная цифровая машина «Киев» является универсальной автоматической вычислительной машиной, предназначенной для научно-технических расчетов. Это трехадресная машина с достаточно высокой скоростью работы — до 15 тыс. сложений, 5 тыс. умножений и 3 тыс. делений в секунду. Если принять, что 80% всех операций являются операциями типа сложения, 15% — операциями умножения, а 5% — операциями деления, то средняя скорость работы машины (при работе с оперативной памятью) составит не менее 10 тыс. операций в секунду. Действительная скорость работы еще выше, поскольку в машине предусмотрены некоторые дополнительные меры для увеличения быстродействия (например, пропуск нулевых адресов).

Машина «Киев» оперирует 40-разрядными двоичными числами (не считая знакового розряда) с запятой, фиксированной перед старшим разрядом. Наличие в составе операций специальной операции нормализации чисел допускает относительно простую программную реализацию режима плавающей запятой. При последующей модернизации машины предусматривается возможность схемной реализации режима плавающей запятой.

Оперативное запоминающее устройство (ОЗУ) в 1024 41-разрядных двоичных кода выполнено на ферритах с прямоугольной петлей гистерезиса. Кроме того, имеется пассивное (также ферритное) запоминающее устройство (ПЗУ) на 512 кодов. В дальнейшем предполагается увеличить емкость ОЗУ еще на 512 кодов.

Цикл обращения к ОЗУ занимает около 10 *мксек*, а цикл обращения к ПЗУ — всего лишь 4 *мксек*. Это создает возможность увеличения быстродействия, так как в ПЗУ хранятся стандартные подпрограммы и универсальные константы, широко используемые в самых различных программах. Указанная возможность фактически реализуется в машине, поскольку «Киев» как асинхронная машина максимально использует быстродействие каждого отдельного устройства.

Внешняя память (ВЗУ) в машине «Киев» выполнена на трех магнитных барабанах общей максимально допустимой емкостью свыше 9 тыс. кодов (точнее 9864 кода). Благодаря применению

последовательно-параллельной системы записи кодов на барабан достигается малое время выборки одного кода (60 *мксек*) при относительно тихоходных барабанах (1500 *об/мин*).

Каждый барабан в свою очередь разделен на тракты. Среднее время ожидания (с учетом времени переброса реле, переключающих тракты) при обращении к барабану составляет около 25 *мсек*.

Дальнейшее расширение внешней памяти предусматривается за счет присоединения дополнительного блока магнитной ленты.

В качестве внешних устройств машины «Киев» (ввод с перфоленты, цифропечатка, выходной перфоратор, клавишное и контрольно-считывающее устройства) временно применены модернизированные внешние устройства машины «Урал» и ввод с перфокарт.

Ввод и вывод чисел осуществляется в десятичной системе счисления (10 десятичных разрядов и знак), а ввод и вывод команд — в восьмеричной системе (14 восьмеричных разрядов). Скорость ввода — около 75 кодов (чисел или команд) в секунду, скорость вывода: на печать — около 100 кодов в минуту, на перфорацию — около 150 кодов в минуту. Перевод чисел из десятичной системы в двоичную осуществляется непосредственно в процессе ввода по специальной команде «ввод чисел» (без применения специальной подпрограммы). При выводе массив чисел должен быть предварительно преобразован по специальной подпрограмме из двоичной системы счисления в двоично-десятичную.

Для ввода команд используется специальная команда «ввод команд», исключающая преобразование кодов; переход от восьмеричной системы записи команд к двоичной осуществляется при перфорировании ленты на клавишном устройстве. Переключение печати с десятичной системы счисления (для чисел) на восьмеричную (для команд) и обратно осуществляется перебросом специального тумблера на выводном устройстве. В машине имеется также перфокартная система ввода и вывода.

Основные устройства машины — арифметическое (АУ), управления (УУ), оперативное запоминающее (ОЗУ), пассивное запоминающее (ПЗУ) и управления магнитной записью (УМЗ) — выполнены в виде отдельных шкафов, каждый из которых имеет отдельный блок питания и связан с остальными шкафами минимальным количеством связей. Магнитные барабаны помещаются на трех отдельных стойках рядом со шкафом УМЗ, в отдельных тумбах размещаются вводное и выводное устройства. Имеется также центральный пульт управления.

Источником энергии служит трехфазный переменный ток напряжением 380/220 *в*; потребляемая мощность около 25 *квт*. В машине используется около 2300 электронных ламп пальчиковой серии (в основном 6Н1П) и 10 000 германиевых диодов (в основном Д1Д).

При кодировании чисел (рис. 1) разряды ячеек машины «Киев» нумеруются справа налево от 1-го до 40-го разряда. 41-й разряд — это разряд знаковый, причем знак 0 соответствует положительным числам, а знак 1 — отрицательным. Запятая фиксируется перед старшим разрядом. Таким образом, диапазон чисел представлен в машине от -1 до $+1$ с точностью до $2^{-40} \approx 10^{-12}$.

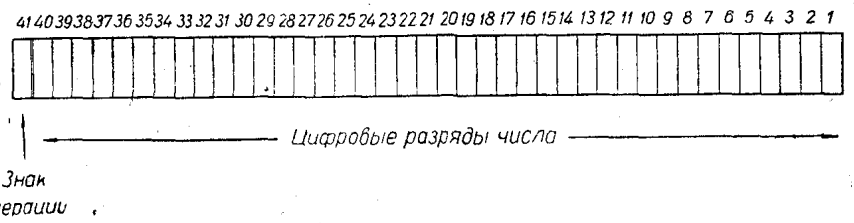


Рис. 1.

В двоично-десятичной системе числа занимают также 40 двоичных разрядов, не считая знакового разряда. Поскольку для кодирования десятичных цифр используются 4-разрядные двоичные коды, то в этом случае представляемые числа имеют лишь 10 десятичных знаков с точностью представления до 10^{-10}. Запятая, как и в предыдущем случае, фиксируется перед старшим разрядом.

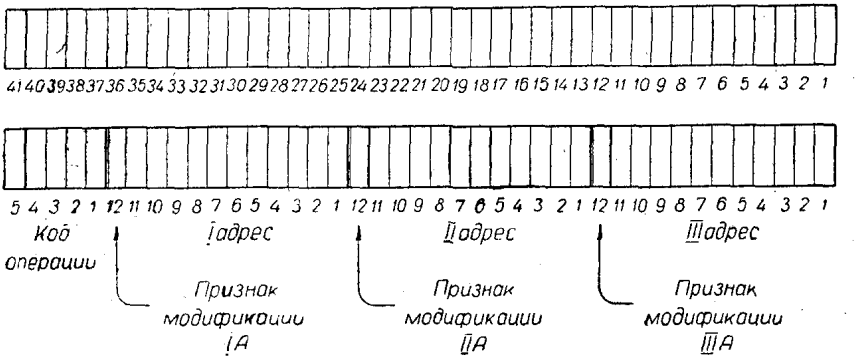


Рис. 2.

Заметим, что двоично-десятичное кодирование чисел в машине принято лишь для чисел, пробиваемых на перфоленте, и для тех чисел в ОЗУ, которые подготовлены к выводу специальной подпрограммой преобразования из двоичного в двоично-десятичный код. В остальных случаях числа в запоминающих устройствах хранятся в прямом двоичном коде. В арифметическом

устройстве и в устройстве управления передача чисел осуществляется как прямым, так и обратным двоичными кодами.

Команды в машине «Киев» занимают по 41 двоичному разряду. Структура команды показана на рис. 2. Как видно из рисунка, 5 старших разрядов (от 37 до 41 включительно) кода команды заняты под код операции. Остальные 36 разрядов составляют адресную часть команды. Таким образом, на каждый из трех адресов отводится по 12 двоичных разрядов.

При вводе и выводе на печать команд для их представления используется восьмеричный код. При этом по четыре восьмеричных разряда отводится на каждый из трех адресов и по два восьмеричных разряда — на код операции. Таким образом, каждая команда представляется 14-разрядным целым восьмеричным числом.

Принято следующее кодирование адресов (одинаковое во всех трех адресах команды): ячейки оперативной памяти ОЗУ имеют восьмеричные адреса от 0000 до 1777 включительно, при этом по нулевому адресу всегда хранится число 0.* Ячейки ПЗУ имеют адреса от 3000 до 3777 включительно. При этом пассивная память в машине «Киев» имеет ячейки трех родов:

1) 8 ячеек — наборные коды, адреса 3000-3007;
2) 184 ячейки — постоянно-спаянные коды, адреса 3010-3277;
3) 320 ячеек — сменно-спаянные коды, адреса 3300-3777.

Последняя ячейка пятого блока сменно-спаянной памяти (имеющая адрес 3777) используется для подсоединения датчика случайных чисел.

Наборные коды представляют собой тумблерные 41-разрядные ячейки (выведенные на пульт управления), в которых могут набираться вручную любые коды.

Постоянно-спаянная память (ПСП) содержит в себе впаянные коды констант, наиболее часто встречающиеся при решении задач, и некоторые стандартные подпрограммы,

В постоянно-спаянную память включены:

1. Константы (номера ячеек от 3010 до 3077).
2. Подпрограмма перевода из двоичной системы счисления в десятичную.
3. Подпрограмма вычисления логарифма $\ln x$.
4. Подпрограмма вычисления $\frac{1}{2}\sin x$.
5. Подпрограмма вычисления $\frac{1}{2}\cos x$.
6. Подпрограмма вычисления $\sqrt{a}$.
7. Подпрограмма вычисления $\frac{1}{4}e^x$.

---

* Все адреса и константы модификации адресов здесь и ниже записываются в восьмеричной системе счисления.

8. Подпрограмма вычисления $\frac{1}{\pi}\arcsin x$.

9. Подпрограмма вычисления $\frac{1}{\pi}\arccos x$.

Сменно-спаянная память (ССП) реализуется в виде отдельных блоков по 64 кода в каждом, на которых размещаются библиотечные подпрограммы. Каждый из блоков ССП укреплен на разъемах и может быть в любой момент заменен (независимо от других блоков) новым, имеющим такую же емкость; в зависимости от необходимости в машину подключаются те или иные из них.

Для кодирования адресов ОЗУ и ПЗУ достаточно иметь 11 двоичных разрядов, причем наличие единицы в 11-м разряде адреса означает признак использования ПЗУ, а наличие в 11-м разряде адреса нуля — признак использования ОЗУ.

Имеющийся в каждом адресе дополнительный 12-й разряд при принятом объеме памяти для кодирования адресов не нужен, он используется как признак модифицируемости адреса. В случае, когда в 12-м разряде какого-нибудь из адресов команды стоит нуль, машина производит обращение к памяти по этому адресу. Если же в 12-м разряде адреса — единица, то соответствующее обращение осуществляется по так называемому *эффективному адресу*, который получается прибавлением к адресу, указанному в команде, содержимого специального *регистра модификации адресов*, или *А*-регистра. Этот регистр состоит из 10 разрядов и может хранить положительные числа от нуля до 1777 включительно. При модификации адреса блокировка переполнения отсутствует, благодаря чему можно, например, модифицировать адрес 2667 с помощью числа 1323 в *А*-регистре и получить в качестве модифицированного адреса $2667 + 1323 - 4000 = 213$.*

Регистр модификации адресов представляет собой один из программных регистров машины. *Программными* будем называть такие регистры, которые переносят информацию от одного цикла работы машины (времени выполнения одного рабочего приказа) к другому.

Кроме регистра модификации адресов, машина «Киев» имеет еще два специальных программных регистра: *регистр возврата* (или *Р*-регистр) и *регистр циклов* (или *Ц*-регистр). Оба эти регистра содержат по 11 двоичных разрядов,

К числу программных регистров относятся также 11-разрядный *счетчик команд* (или *С*-регистр) и 41-разрядный *регистр команд* (или *К*-регистр). Остальные регистры машины «Киев» при составлении программ не используются и не переносят информации от цикла к циклу. Их можно назвать *микропрограммными* регистрами, поскольку обмен информацией с ними происходит по жестким микропрограммам, заложенным в машину при ее постройке и не изменяемым по желанию программиста.

---

* См. сноску на стр. 8.

ОПЕРАЦИИ, ВЫПОЛНЯЕМЫЕ МАШИНОЙ

Операционная часть команды машины «Киев» содержит пять двоичных разрядов, что позволяет кодировать 32 операции. Поскольку операция с нулевым кодом является нулевой, т. е. не выполняется, то имеется возможность для кодирования только 31 операции. В машине фактически реализовано 29 операций.

Выполняемые машиной «Киев» операции делятся на следующие пять групп: арифметические, логические, передачи управления, с регистром модификации адресов, с внешними устройствами (включая память на магнитных барабанах).

Цикл выполнения каждой операции начинается после установления адреса соответствующей команды на счетчике команд и заканчивается установлением на счетчике команд адреса следующей из подлежащих выполнению команд. При этом каждый очередной цикл начинается с передачи на регистр команд команды, адрес которой установился в счетчике команд в конце предыдущего цикла. Обычно содержимое счетчика команд увеличивается в течение цикла на единицу, поэтому следующая по порядку команда выбирается из ячейки памяти, имеющей адрес, на единицу больший адреса предыдущей команды. В этом случае будем говорить, что выполняется следующая по порядку команда. Такой *естественный* порядок выполнения команд может нарушаться только при выполнении операций передачи управления и групповых, когда при выполнении тех или иных дополнительных условий в счетчик адреса засылается содержимое III или II адреса текущей команды или же содержимое регистра возврата, которое (без увеличения на единицу) и служит адресом следующей команды. Условимся говорить для краткости, что в данном случае происходит передача управления по II или III адресу или же по регистру возврата.

Условимся обозначать через a_1, a_2, a_3 соответственно I, II и III адреса команды и, кроме того, для любого адреса c через $'c$ будем обозначать код, хранящийся по этому адресу.

Переходя к характеристике операций, выполняемых машиной, раньше опишем *арифметические операции*. Таких операций в машине насчитывается девять: обычное сложение $+$ (восьмеричный код операции 01), вычитание $-$ (код 02), вычитание модулей $|-|$ (код 06), умножение без округления $\times$ (код 10), умножение с округлением $\boxtimes$ (код 11), деление $:$ (код 12), нормализация чисел H (код 35). К числу арифметических операций отнесем также сложение команд *СлК* (код 03) и циклическое сложение $Ц+$ (код 07).

Все арифметические операции характеризуются тем, что в случае наличия единицы в 12-м разряде любого из трех адресов команды к соответствующему адресу прибавляется содержимое A-регистра, т. е. все операции выполняются с эффективными адресами. Передача управления — нормальная: после выполне-

ния команды с арифметической операцией всегда выполняется следующая по порядку команда.

При выполнении операций сложения, вычитания, деления и сложения команд результат по модулю может оказаться бо́льшим или равным единице. В этом случае предусмотрен останов машины с соответствующей сигнализацией. С помощью переключения специального тумблера на пульте управления можно блокировать останов при выходе из располагаемого числа разрядов; машина будет продолжать работу даже при наличии переполнения, пропуская при этом следующую команду и теряя, разумеется, старшие разряды чисел.

При выполнении операции *обычного сложения* (+′) к числу $'a_1$ прибавляется (алгебраически) число $'a_2$, а полученная сумма засылается по адресу a_3 (все адреса здесь и ниже — эффективные!)

Сложение команд (СлК) отличается от обычного сложения тем, что число $'a_1$ прибавляется не к числу $'a_2$, а к его модулю (т. е. к числу $'a_2$, взятому со знаком плюс), полученной же сумме снова присваивается знак числа $'a_2$. Эта операция используется при переадресации команд: переадресуемая команда хранится по II адресу, а восстановление знака в сумме обеспечивает неизменность старшего разряда кода операции после переадресации (напомним, что старший разряд кода операции в команде соответствует знаковому разряду в коде числа).

Циклическое сложение (Ц+) отличается от обычного сложения лишь тем, что в нем отсутствует блокировка при выходе из располагаемого числа разрядов. Перенос из знакового разряда поступает в младший разряд сумматора. Операция циклического сложения употребляется главным образом в программах контроля.

При операции *вычитания* (—) разность $'a_1 - 'a_2$ засылается по адресу a_3; при операции *вычитания модулей* (|—|) то же самое выполняется с разностью модулей $|'a_1| - |'a_2|$.

В операциях *умножения без округления* (×) и *деления* (:) по II адресу соответственно засылаются числа $'a_1 \times 'a_2$ и $'a_1 : 'a_2$. При этом в случае *умножения с округлением* (⊠) в старший из отбрасываемых разрядов добавляется единица.

Более подробно следует остановиться на операции *нормализации чисел (Н)*. Число n мы будем называть нормализованным (в двоичной системе счисления), если оно представлено в виде

$$n = \pm 2^k m,$$

где $m = 0{,}1\ m_1 m_2 \ldots$, а k — целое положительное или отрицательное число. Число k называется *порядком*, а число m — *мантиссой* числа n. Для мантиссы всегда имеют место неравенства

$$1 > m \geqslant \frac{1}{2}.$$

---

* Через $'a_i$ здесь и ниже обозначается содержимое ячейки памяти, имеющей адрес a_i.

В машине «Киев» максимальная величина порядка принята равной 63. Таким образом, для представления порядка требуется шесть двоичных разрядов. Операция нормализации заключается в том, что число $'a_1$ нормализуется, причем порядок нормализованного числа засылается по II (занимая шесть младших разрядов кода), а мантисса — по III адресу.

Перейдем теперь к логическим операциям, которых в машине насчитывается четыре: логический сдвиг $Л \rightarrow$ (восьмеричный код операции — 13), логическое сложение $\vee$ (код 14), логическое умножение $\wedge$ (код 15) и поразрядная операция неравнозначности $\cong$ (код 17). При выполнении всех логических операций осуществляется нормальная передача управления (выполнение следующей по порядку команды). Как и в случае арифметических операций, возможна модификация всех трех адресов.

При операции *логического сдвига* ($Л \rightarrow$) осуществляется сдвиг всех 41 разрядов числа $'a_2$ (включая знаковый разряд) на число разрядов, равное абсолютной величине константы сдвига, размещаемой в шести младших разрядах ячейки a_1. Если константа сдвига положительна, то сдвиг осуществляется влево; если отрицательна — вправо. Разряды числа, вытесняемые при сдвиге за пределы разрядной сетки, теряются, а вместо них с противоположной стороны появляются нули (при сдвиге на 41 или больше разрядов любое число превращается в нуль). Сдвинутое число засылается по III адресу.

Остальные три логические операции выполняются поразрядно над всеми 41 разрядами кодов $'a_1$ и $'a_2$, так что n-й разряд результата зависит лишь от n-х разрядов кодов $'a_1$ и $'a_2$. Результаты, как и в предыдущих случаях, засылаются по III адресу.

Операция *логического сложения* (дизъюнкция) обозначается через $\vee$ и задается соотношениями $0 \vee 0 = 0$, $0 \vee 1 = 1$, $1 \vee 1 = 1$, $1 \vee 0 = 1$.

Операция *логического умножения* (конъюнкция) имеет символ $\wedge$ и соотношения $0 \wedge 0 = 0$, $0 \wedge 1 = 0$, $1 \wedge 0 = 0$, $1 \wedge 1 = 1$, а операция неравнозначности — символ $\cong$ и соотношения $0 \cong 0 = 0$, $1 \cong 0 = 1$, $0 \cong 1 = 1$, $1 \cong 1 = 0$.

В машине «Киев» насчитывается семь операций передачи управления: обычное сравнение *Ср1* (восьмеричный код операции 04), сравнение модулей *Ср2* (код 05), точное сравнение *Ср3* (код 16), условный переход по знаку числа *УПЧ* (код 31), условный переход на подпрограмму *УПП* (код 30), переход по регистру возврата *ПРВ* (код 32), безусловный останов машины *Ост* (код 33). Во всех командах передачи управления, кроме перехода по регистру возврата и останова, все адреса допускают модификацию.

Операция *обычного сравнения (Ср1)* заключается в том, что сравниваются числа $'a_1$ и $'a_2$. Если $'a_1 \leqslant 'a_2$, то управление пе-

редается команде по адресу a_3; если же $'a_1 > 'a_2$, то выполняется следующая по порядку команда.

Операция *сравнения модулей (Ср2)* отличается от операции обычного сравнения тем, что в ней сравниваются не сами числа $'a_1$ и $'a_2$, а их абсолютные величины (модули).

Операция *точного сравнения (Ср3)* означает, что управление передается команде по адресу a_3, если числа $'a_1$ и $'a_2$ равны между собой по величине и по знаку. В случае же неравенства чисел $'a_1$ и $'a_2$ выполняется следующая по порядку команда.

Условный переход по знаку числа (УПЧ) означает передачу управления команде по адресу a_2, если знак числа $'a_1$ положительный, и по адресу a_3 — в противном случае. Нуль в машине может быть как положительным, так и отрицательным (нуль, получаемый как разность двух равных чисел, имеет отрицательный знак).

Условный переход на подпрограмму (УПП) осуществляет следующие операции: если число $'a_1$ положительно (или равно нулю), то выполняется следующая по порядку команда; если же число $'a_1$ отрицательно, то управление передается команде по адресу a_3. Одновременно число a_2 (не $'a_2$!) заносится в регистр возврата (предыдущее содержимое этого регистра вытесняется). Каждая *УПП* применяется обычно при переходе на стандартную подпрограмму. Тогда число a_2 означает адрес команды, к которой надо перейти после выполнения подпрограммы. Стандартная подпрограмма должна оканчиваться специальной командой «переход по регистру возврата».

Переход по регистру возврата (ПРВ) означает безусловную передачу управления команде, адрес которой указан в регистре возврата. Адресная часть самой команды *ПРВ* при этом не используется, поэтому она может быть произвольной.

Операция *останов (Ост)* означает безусловный останов машины. Адресная часть команды, как и в предыдущем случае, не используется.

В машине «Киев», кроме программного, возможны еще два контрольных останова. Один из них предусматривает останов машины в случае выполнения машиной команды, адрес которой набирается на специальном тумблерном регистре центрального пульта управления. Второй контрольный останов происходит в случае совпадения третьего адреса выполняемой машиной команды с адресом, набранным тумблерами на пульте управления.

Переходим к описанию операций с регистром модификации адресов, являющихся наиболее сложными операциями машины «Киев». Эта группа состоит всего из трех операций: заполнение *А*-регистра по фиксатору (вызов по фиксатору) *Ф* (восьмеричный код 34), начало групповой операции *НГО* (код

26) и окончание групповой операции *ОГО* (код 27). При выполнении операций этой группы имеет место лишь та модификация адресов, которая специально оговаривается при описании каждой из этих операций.

В случае операции *Ф* часть кода $'a_1$, расположенная от 13-го по 24-й разряд включительно (иначе говоря, его II адрес), передается на *А*-регистр (предшествующее содержимое этого регистра вытесняется). Кроме этого, осуществляется пересылка числа $'m$, где m — число, переданное на *А*-регистр, в ячейку с адресом a_3. II адрес при операции *Ф* не используется. Передача управления осуществляется естественным образом, т. е. после операции *Ф* выполняется следующая по порядку команда. Операция *Ф* позволяет кодировать в постоянной памяти циклические процессы произвольной глубины.

Заметим, что операция *Ф* может быть использована одновременно для пересылки кода, заданного адресом его адреса (адресом второго ранга), в любую ячейку ОЗУ.

Взяв в качестве a_3 адрес 0000, получим возможность использовать операцию *Ф* только для заполнения *А*-регистра. При этом в любом случае в последующих командах программы можно пользоваться естественной модификацией адресов.

Другие две операции рассматриваемой группы *(НГО* и *ОГО)* употребляются при программировании циклов и носят специальное название *групповых* операций.

При операции *НГО* на регистр циклов засылается число a_1 (не $'a_1$!), т. е. содержимое I адреса команды *НГО*, а на регистр модификации адресов — число a_2 (не $'a_2$!), т. е. содержимое II адреса команды *НГО*. Если после этого содержимые регистра циклов и регистра модификации адресов оказываются равными, то управление передается команде по адресу a_3, в противном случае — следующей по порядку команде.

При операции *ОГО* содержимое регистра модификации адресов увеличивается на содержимое I адреса команды *ОГО* (число a_1, а не $'a_1$!). Если после этого содержимые регистра циклов и регистра модификации адресов окажутся равными, то управление передается команде по адресу a_3, в противном же случае — команде по адресу a_2.

Для того чтобы произвести одинаковый набор операций над группой чисел, расположенных в ячейках, адреса которых составляют арифметические прогрессии с одной и той же разностью p, можно с помощью операции *НГО* заслать в регистр циклов число, равное содержимому регистра модификации адресов, которое потребуется для выполнения указанного выше набора операций над последними числами группы. Тогда участок программы, реализующий данный набор операций над группой чисел, можно начать командой *НГО* и закончить командой *ОГО*. Между этими командами должен быть помещен заданный набор команд,

у которых в 12-х разрядах необходимо поставить единицы — признаки модифицируемости адресов. Команда *НГО* должна выполняться только один раз, а команда *ОГО* — повторяться циклически до тех пор, пока не будут выполнены операции над всеми числами группы. По второму адресу команды *ОГО* должен быть задан адрес начальной команды цикла.

Последнюю группу операций машины «Киев» составляют шесть операций с внешними устройствами. Три из них являются операциями с магнитными барабанами: подготовительная для обращения к магнитному барабану *МБП* (восьмеричный код операции 25), исполнительная при записи кодов *МБЗ* (код 23) и исполнительная при считывании кодов *МБЧ* (код 24). Другие три операции осуществляют ввод и вывод данных: ввод чисел *В1* (код 20), ввод команд *В2* (код 21) и вывод *Печ* (код 22).

Все операции с внешними устройствами являются групповыми. Они разрушают содержимое регистра модификации адресов.

Операция *МБП* является *подготовительной операцией для обмена кодами с магнитными барабанами*. Содержимое II адреса команды a_2 (а не $'a_2$!) указывает номер магнитного барабана и номер тракта, с которым будет производиться обмен кодами при выполнении следующей операции (*МБЗ* и *МБЧ*). Содержимое III адреса команды a_3 (а не $'a_3$!) указывает номер числа в выбранном тракте, с которого начинается обмен кодами. Содержимое I адреса команды *МБП* может принимать только два значения: нуль — если подготавливается чтение на магнитный барабан, единица — если подготавливается запись с него. После выполнения операции *МБП* управление передается следующей по порядку команде, которой должна быть обязательно *МБЗ* или *МБЧ*.

Необходимо отметить особенность заполнения II адреса в команде *МБП*. Для кодирования трех номеров барабанов достаточно двух двоичных разрядов. В действительности же, чтобы обеспечить удобство записи команды в восьмеричной системе счисления, для кодирования номера барабана отводится третий восьмеричный разряд II адреса.

Операция *МБЗ* осуществляет *запись кодов* из ОЗУ или ПЗУ *на магнитный барабан*; числа из ячеек с адресами от a_1 до a_2 включительно записываются на тот из трактов магнитных барабанов, который был подготовлен предыдущей операцией *МБП*. После выполнения операции *МБЗ* управление передается команде по адресу a_3.

Операция *МБЧ* означает *чтение с магнитного барабана* группы кодов и запись их в ОЗУ. Коды барабана, подготовленного предыдущей операцией *МБП*, записываются в ячейки ОЗУ с адресами от a_1 до a_2 включительно. После выполнения операций *МБЗ* и *МБЧ* управление передается по адресу a_3.

При выполнении обеих операций (*МБЗ* и *МБЧ*) происходит автоматическое переключение на следующий по порядку тракт

того же самого магнитного барабана, если передаваемые числа не помещаются на одном тракте. С последнего (24-го) тракта каждого барабана переключение не производится.

Операция *ввод чисел* означает, что с перфоленты или перфокарт вводится группа чисел в ячейки ОЗУ, имеющие адреса от a_1 до a_2 включительно. Числа при этом автоматически переводятся из десятичной системы в двоичную. При вводе с перфокарт в III адресе команды указывается номер зоны; начальное из вводимых чисел определяется началом данной зоны, конечное — количеством вводимых чисел.

Операция *ввод команд* отличается от ввода чисел лишь отсутствием перевода из десятичной системы в двоичную.

Операция *вывод* означает, что выводится группа кодов из ячеек ОЗУ, имеющих адреса от a_1 до a_2 включительно. После выполнения операции управление передается команде по адресу a_3. Коды могут выводиться как на печать, так и на перфорацию переключением специального тумблера. Аналогичным образом с помощью переключения дополнительного тумблера может быть выбран вид печати: десятичный (для чисел) или восьмеричный (для команд). Напоминаем, что при выводе на десятичную печать выводимый массив чисел с помощью специальной программы должен быть предварительно преобразован в двоично-десятичную систему счисления.

Начальный запуск машины может осуществляться двумя различными способами. Во-первых, после предварительной установки машины в начальное положение можно тумблерами набрать на пульте управления команду ввода (команд или чисел). Запуском машины для работы в режиме чтения команды с пульта осуществляется ввод начальной информации в машину. После этого необходимо набрать на пульте управления адрес первой команды программы и запустить машину для решения задачи.

Второй способ заключается в том, что начальная команда ввода набирается на одном из тумблерных регистров ПЗУ, а адрес этой команды — на пульте управления. После нажатия ключа сначала осуществляется ввод, а затем машина автоматически переходит к выполнению введенной программы.

ПРЕДСТАВЛЕНИЕ КОДОВ В МАШИНЕ

Представление чисел

Машина «Киев» оперирует двоичными числами, представленными в системе фиксированной запятой. 40 разрядов ячейки используются для записи мантиссы числа и один разряд (41-й) — для знака числа (см. рис. 1): 0 соответствует знаку плюс, а 1 — знаку минус. Таким образом, машина оперирует числами

в диапазоне от $-1+2^{-40}$ до $1-2^{-40}$. Число, меньшее по модулю, чем 2^{-40}, записывается в ячейку как машинный нуль. При этом $-0<+0$.

Для сокращения записи двоичных чисел принята следующая система. Двоичное число разбивается на группы, из которых первая содержит два двоичных разряда, включая знаковый, а последующие состоят из трех разрядов каждая (всего 14 групп). Затем группы кодируются соответствующими восьмеричными цифрами. Так, отрицательное число с кодом 1·1 110 010 011 010 100 001... записывается в виде 3 623 241..., а положительное число с кодом 0,1 110 010... — в виде 1 62...

Для перевода чисел из десятичной системы в принятую на машине запись удобна следующая методика: n-значное число умножается на 2; $(n+1)$-й разряд положительного числа есть первая цифра искомой записи; $(n+1)$-й разряд отрицательного числа, увеличенный на два, есть первая цифра искомой записи. Оставшиеся n разрядов результата (остаток) умножаются на 8. $(n+1)$-й разряд есть вторая цифра искомой записи. Все последующие остатки аналогично умножаются на 8. Процесс продолжается до тех пор, пока в остатке не появится 0, или до получения необходимой точности. Например, перевод числа 0,625

$$\begin{array}{ll} 0 & 625\times 2 \\ 1 & 250\times 8 \\ 2 & 000 \end{array}$$

дает результат 12 0000 0000 0000. Перевод числа —0,625 даст запись 32 0000 0000 0000.

Принятая запись двоичных чисел делает единообразной запись чисел и команд, в чем и заключается ее удобство.

Десятичные числа вводятся в машину в двоично-десятичном коде. Для записи одного десятичного разряда при этом используется четыре двоичных разряда. Таким образом, максимальное число десятичных разрядов, которое может быть введено в одну ячейку машины или выведено на печать, равно десяти.

Представление приказов (команд)

Числа и команды в машине могут кодироваться в одних и тех же ячейках ЗУ. Код машины «Киев» содержит четыре группы цифр: код операции (0-й адрес) и три адреса (I, II, III). Код операции, содержащийся в нулевом адресе, определяет смысл адресов. Под код операции отводится пять старших разрядов, под каждый из адресов — по 12 разрядов. Старший разряд каждого из адресов используется в качестве признака модифицируемости адреса (признак групповой операции).

При записи на бланках в восьмеричных числах адрес кодируется вместе с признаком модифицируемости. Таким образом, в восьмеричном коде признаком модифицируемости адреса является соотношение $a > 3777$. Для сообщения адресу признака модифицируемости необходимо к нему прибавить число 4000.

Команда, код которой состоит из нулей, машиной не выполняется. После такта работы машины вхолостую управление передается следующей по номеру команде.

ГЛАВА II

ПРОБЛЕМА ОБОСНОВАНИЯ ВЫБОРА ОСНОВНЫХ ХАРАКТЕРИСТИК

УНИВЕРСАЛЬНЫЙ ЭКОНОМИЧЕСКИЙ КРИТЕРИЙ ЭФФЕКТИВНОСТИ АВТОМАТИЧЕСКОЙ ЦИФРОВОЙ МАШИНЫ

Приступая к работе по созданию электронной машины «Киев», авторский коллектив не располагал сколько-нибудь обоснованными критериями для определения степени эффективности будущей машины. Поэтому выбор основных параметров машины «Киев» был осуществлен на основе интуиции и обобщения накопленного к тому времени опыта. Только на втором этапе работы (1958 г.) В. М. Глушковым [2] был предложен универсальный экономический критерий, позволяющий сравнивать эффективность вычислительных машин самых различных типов. Этот критерий получил название *критерия цены эффективного быстродействия.*

В основу критерия цены эффективного быстродействия положены две идеи. Первая из них заключается в экономическом подходе к оценке производительности машины, который легко уяснить себе на следующем примере. Пусть для решения какого-нибудь класса задач разработаны вычислительные машины A и B, причем машина A обладает в два раза бóльшим быстродействием, чем машина B. Можно ли утверждать, что машина A более эффективна, чем машина B? Разумеется, нет. Ведь могло случиться, что машина A оказалась гораздо дороже машины B и требует больших эксплуатационных расходов и т. д. и т. п. Может оказаться поэтому, что общая сумма затрат при решении какого-либо класса задач на машине A, например, в шесть раз превосходит аналогичные затраты для машины B. Поэтому при одной и той же сумме затрат можно иметь вместо одной машины A шесть машин B, которые смогут за одно и то же время выполнить в три раза больший объем работы, чем машина A. Таким образом, *в экономическом плане* менее быстродействующая машина может оказаться более эффективной, чем машина с бóльшим быстродействием. Благодаря именно этому обстоятельству при решении многих задач оказывается выгодным применять настольные клавишные вычислительные машины, быстродействие которых в десятки тысяч раз меньше, чем быстродействие электронных цифровых машин. Определяющим является не быстродействие само по себе, а цена быстродействия.

Следует, разумеется, иметь в виду, что учет экономических факторов, по крайней мере в том виде, как это было только что изложено, ни в коем случае нельзя абсолютизировать. Действительно, может, например, оказаться настолько важным решить задачу в возможно более короткое время, что следует стремиться к максимальному увеличению быстродействия, не считаясь ни с какими затратами. То же самое может иметь место и в отношении некоторых других факторов (габарит машины, потребление энергии и др.). Впрочем экономический критерий можно так перефразировать, чтобы и эти исключительные случаи были им охвачены.

Вторая идея, положенная в основу критерия цены эффективного быстродействия, заключается в том, что скорость выполнения машиной тех или иных элементарных операций (сложение, умножение и т. п.) еще не определяет быстродействия машины, даже для автоматических вычислительных машин с программным управлением. В самом деле, хорошо известно, что скорость работы машины может быть существенно повышена за счет рационального выбора операций управления и программных регистров (так называемой логики машины). Существенное влияние на скорость решения задач оказывают также такие параметры машины, как объем оперативного запоминающего устройства, скорость обмена с внешней памятью, скорость ввода и вывода. Наконец, нельзя забывать, что реальное быстродействие машины резко снижается при малой ее надежности, поскольку при этом приходится тратить много времени не только на устранение неисправностей в машине, но и на многократные контрольные просчеты, дублирование вариантов и т. п.

Говоря о цене быстродействия, надо иметь в виду не обычное номинальное быстродействие, т. е. указываемую в характеристике машины скорость выполнения основных арифметических операций, а эффективное быстродействие, учитывающее снижение скорости за счет операций с внешними устройствами, дублирования расчетов для получения достаточной надежности результата и другие подобные факторы.

Перейдем теперь к более точной формулировке критерия цены эффективного быстродействия. Пусть фиксирован некоторый класс K задач, которые предполагается решать на вычислительной машине A. Будем считать, что задана статистика класса K, если для каждой задачи k из K задана относительная частота $f(k)$, с которой эта задача будет встречаться при решении задач класса K в течение неограниченно большого промежутка времени.

Фиксируем далее некоторый *типичный набор* O операций универсальной цифровой машины и предположим, что для каждой задачи k из K известно количество $\varphi(k)$ операций из O, необходимых для ее решения. Тогда любую работу по решению задач класса K, проделанную машиной A, можно характеризовать числом

операций типичного набора, необходимых для выполнения той же работы. Фиксируем теперь некоторый достаточно большой промежуток времени T (целесообразно в качестве T выбирать промежуток времени порядка 10 лет) и подсчитаем объем работы, которую машина A выполнит за этот период с учетом потерь времени на профилактику машины, дублирование расчетов, операции внешних устройств и т. п. Пусть этот объем выражается числом $N = N(T)$ операций типичного набора. Деля N на количество секунд в T, получим величину $n_э$, называемую *эффективным быстродействием* машины.

Подсчитаем также суммарные затраты $Z(T)$ на изготовление, амортизацию и эксплуатацию машины за время T. В суммарные затраты войдут, в частности, затраты на эксплуатационный персонал и расходы на электроэнергию. Вместе с тем затраты на программирование задач нецелесообразно включать в $Z(T)$, ибо эти затраты зависят не столько от конструкции машины, сколько от имеющегося задела в программировании (в первую очередь от объема библиотеки стандартных подпрограмм). Теперь цена эффективного быстродействия q подсчитывается по формуле

$$q = \frac{Z(T)}{N(T)} = \frac{Z(T)}{Tn_э}. \qquad (1)$$

Нетрудно понять, что одна из основных задач проектировщика вычислительной машины должна состоять в максимально возможном снижении цены эффективного быстродействия.

Для того чтобы исключить влияние неопределенности в выборе промежутка времени T, можно было бы в формуле (1) перейти к пределу

$$q = \lim_{T \to \infty} \frac{Z(T)}{N(T)}.$$

Однако на практике этот предел с достаточной степенью точности достигается при значениях T порядка 10 лет и выше. Кроме того, не следует забывать, что при $T \to \infty$ исключается учет морального старения и необходимость замены устаревшего оборудования. Поэтому сейчас, по-видимому, наиболее рационально выбирать значение T, равное 10 годам.

Заметим, что если величину $Z(T)$ подсчитать относительно нетрудно, то величина $N(T)$ существенно зависит от статистики класса K решаемых задач, которая, как правило, неизвестна. Впрочем и при известной статистике класса K точный подсчет величины $N(T)$ весьма затруднителен. Поэтому в ряде случаев наиболее целесообразным методом является ориентировочный подсчет коэффициента p, равного отношению эффективного быстродействия машины к ее номинальному быстродействию. Для под-

счета коэффициента p можно пользоваться приближенной формулой

$$p = p_0 p_п p_м p_в p_к. \qquad (2)$$

Здесь p_0 — отношение номинального быстродействия машины, выраженного в операциях типичного набора, к номинальному быстродействию, выраженному в истинных операциях машины. Коэффициент p_0 определяется набором операций машины и статистикой решаемых задач. Он может быть и больше и меньше единицы. Через $p_п$ обозначен коэффициент использования времени в машине (равный части времени, затрачиваемой на решение задач), он всегда меньше единицы. Обычно значение $p_п$ лежит в пределах 0,7—0,9.

Через $p_м$ и $p_в$ обозначены коэффициенты, показывающие снижение быстродействия за счет операций с магнитными барабанами и магнитными лентами ($p_м$) и за счет операций с устройствами ввода-вывода ($p_в$). Эти коэффициенты определяются объемами оперативной памяти, скоростью обмена информацией с внешними устройствами и статистикой класса решаемых задач. Их величина всегда меньше единицы.

Наконец, $p_к$ — коэффициент, показывающий снижение быстродействия машины за счет программного контроля, дублирования расчетов и других средств контроля правильности результатов решения задачи. Этот коэффициент также всегда меньше единицы. Его значение определяется прежде всего динамической надежностью машины (вероятностью сбоев) и зависит также от статистики класса решаемых задач.

Машине «Киев» приходится решать достаточно сложные задачи, поэтому не будет большой ошибки, если примем значение коэффициента p для нее равным 0,1. Такое же значение коэффициента p можно, по-видимому, взять и для других машин с объемом оперативной памяти порядка 1000—4000 чисел.

Обозначая через n номинальное быстродействие машины (среднее количество операций в секунду, выполняемое машиной при работе с оперативной памятью) и учитывая, что в 10 годах насчитывается около $3 \cdot 10^8$ *сек*, получим формулу для цены эффективного быстродействия

$$q = \frac{Z(T)}{3 \cdot 10^8 pn}. \qquad (3)$$

Отметим, что даже при выражении величины $Z(T)$ в копейках q будет значительно меньше единицы. Поэтому целесообразно выражать q в копейках за каждую тысячу операций, или в рублях за миллион операций.

Для машины «Киев» при T, равном 10 годам, величина $Z(T)$ ориентировочно равна 1,2 млн. руб. (0,4 млн. руб. — первоначальная стоимость, 0,5 млн. руб. — амортизация, 0,1 млн. руб. —

электроэнергия, 0,2 млн. руб. — зарплата эксплуатационного персонала, включая перфораторщиц). Поэтому цена эффективного быстродействия для машины «Киев» выражается ориентировочно величиной

$$\frac{1{,}2 \cdot 10^6 \cdot 100}{3 \cdot 10^8 \cdot 0{,}1 \cdot 10\,000} = 0{,}0004 \text{ коп/опер}$$

или 0,4 коп. за тысячу операций, или 4 руб. за миллион операций.

Для машины «Урал» первых выпусков величина $Z(T)$ снижается примерно до 0,4 млн. руб. Вместе с тем эффективное быстродействие (величина pn) для «Урала» вряд ли превышает 15 *опер/сек*. Поэтому цена эффективного быстродействия для машины «Урал» выражается величиной порядка

$$\frac{4 \cdot 10^6 \cdot 100}{3 \cdot 10^8 \cdot 15} = 0{,}01 \text{ коп/опер},$$

или 10 коп. за тысячу операций, или 100 руб. за миллион операций.

Таким образом, при заданных условиях машина «Киев» оказывается примерно в 25 раз более эффективной вычислительной машиной, чем «Урал». Следует, разумеется, иметь в виду, что такая оценка справедлива лишь при условии, что на обеих машинах решается один и тот же класс задач, причем задач относительно сложных, характеризующихся достаточно большим отношением числа собственно машинного времени к времени ввода и вывода. С нарушением этого условия картина резко меняется, поэтому при решении коротких задач с большим вводом преимущество может оказаться на стороне машины «Урал».

В заключение отметим, что учет всех факторов, влияющих на эффективное быстродействие вычислительной машины, часто оказывается весьма затруднительным. Очень трудно, например, вычислить сколько-нибудь точно величину коэффициента $p_к$. Поэтому в приложениях целесообразно использовать упрощенные расчеты, основанные на том, что один или несколько из коэффициентов p_0, $p_п$, $p_м$, $p_в$, $p_к$ принимаются равными единице. Такое упрощение оказывается вполне допустимым в том случае, если критерий цены эффективного быстродействия употребляется для определения оптимального значения параметра, мало влияющего на коэффициенты, приравненные к единице. Можно также вместо минимизации цены эффективного быстродействия стремиться к максимальному увеличению самого эффективного быстродействия безотносительно к цене.

Необходимо также отметить, что при современном состоянии теории электронных вычислительных машин, как правило, точная оценка эффективного быстродействия оказывается возможной лишь апостериори, поскольку не существует сколько-нибудь удовлетво-

рительных методов априорной оценки динамической надежности машины (вероятности сбоев). Кроме того, статистика класса решаемых задач, как правило, заранее неизвестна.

АДРЕСНОСТЬ И РАЗРЯДНОСТЬ

Вопрос о целесообразном количестве адресов в универсальной электронной цифровой машине можно решить на основании критерия цены эффективного быстродействия. Основную роль при этом будет играть статистика класса решаемых задач. Поскольку для машины «Киев» такая статистика не производилась, вопрос об ее адресности мог решаться лишь на основании простейших качественных соображений. Легко понять, что критерий цены эффективного быстродействия зависит от эффективности использования оборудования и, в частности, от наличия или отсутствия лишних передач информации внутри машины. С этой точки зрения в машинах с параллельной передачей кодов наиболее эффективной является одноадресная система команд.

Снижение эффективности использования оборудования, возникающее при использовании многоадресных команд, легко продемонстрировать на примере подсчета суммы трех и более слагаемых. Действительно, при подсчете суммы $x + y + z$ в случае использования трехадресной системы команд результат первого сложения $x + y$ засылается в какую-нибудь ячейку запоминающего устройства только для того, чтобы следующей командой снова извлечь его для засылки в сумматор.

При использовании же одноадресных команд обычно возникает проблема несоответствия длин кодов чисел и команд, приводящая либо к снижению эффективности использования памяти машины, либо к усложнению ее коммутации. С учетом этих дополнительных соображений трехадресную систему команд можно считать примерно равноценной с одноадресной системой. Решающим фактором, определившим выбор для машины «Киев» именно трехадресной системы, послужил опыт работы с трехадресными машинами, накопленный коллективом Вычислительного центра АН УССР.

Статистика класса решаемых задач оказывает определяющее влияние на выбор счисления и системы представления чисел (фиксированная или плавающая запятая). Из двух систем счисления — десятичной и двоичной, получивших распространение в практике мирового математического машиностроения, двоичная система обеспечивает бо́льшую эффективность использования оборудования с точки зрения внутренних операций машины. Вместе с тем при вводе и выводе информации сказываются определенные преимущества десятичной системы, не нуждающейся в дополнительных операциях преобразования чисел из одной системы счисления в другую.

Таким образом, выбор той или иной системы счисления определяется прежде всего характером задач, решаемых на машине. Для задач с относительно малым объемом вводимой и выводимой информации по сравнению с числом операций, производимым над этой информацией внутри машины, более целесообразно использовать двоичную систему счисления. Такое положение имеет место при решении большинства сложных научных задач, на которые рассчитана в основном машина «Киев». Поэтому для нее в качестве внутренней системы счисления и избрана двоичная система.

Выбор системы представления чисел (плавающая или фиксированная запятая) определяется прежде всего характером решаемых на машине задач. При преимущественном решении неарифметических задач и таких арифметических задач, в которых порядки чисел имеют примерно одинаковую величину, фиксированная запятая обеспечивает меньшую цену эффективного быстродействия и является, следовательно, более выгодной. В тех же задачах, в которых необходимо часто изменять масштабы чисел в ходе вычислений, плавающая запятая обеспечивает большие преимущества и создает большие удобства при программировании задач.

Следует, однако, заметить, что и применение плавающей запятой часто не исключает необходимости заранее оценивать порядки чисел, возникающих в процессе решения задачи. В противном случае можно столкнуться с непредвиденной потерей точности при вычислении разности двух близких по величине чисел. Особенно опасным является то обстоятельство, что при применении плавающей запятой такая потеря точности происходит незаметно, поскольку мантисса всегда сохраняет строго определенное число значащих цифр. Например, при вычислении по формуле $u = (x - y)z$ для $x = 0{,}1278$; $y = 0{,}1273$; $z = 0{,}9812$ полученный результат $u = 10^{-3} \cdot 0{,}4906$ будет давать совершенно неправильное представление о числе верных в нем знаков, тогда как при использовании фиксированной запятой полученный результат $u = 0{,}0005$ будет сразу свидетельствовать о происшедшей потере точности.

Переходим, наконец, к рассмотрению последнего вопроса, связанного с представлением чисел в машине, — к выбору их разрядности. Фактически этот вопрос в машине «Киев» был определен из условия равенства для кодов чисел и команд, выполнение которого имеет существенные преимущества с точки зрения эффективности использования памяти. Что же касается длины кода команды, то она определилась из необходимости кодирования 29 операций и трех адресов. Поскольку объем памяти (с учетом резерва) был установлен в $2048 = 2^{11}$ чисел, а еще один двоичный разряд в каждом адресе отведен на признак модификации, то длина кода команды составляет $5 + 3(11 + 1) =$

=41 двоичный разряд. Такой же выбрана и длина кода числа (со знаковым разрядом).

Для решения вопроса о том, достаточна ли такая длина для кода числа, необходимо обратиться к характеру решаемых на машине задач. Основное значение здесь приобретают два показателя: требуемая максимальная точность результатов и максимальная длина цепочек последовательных вычислений, в которых происходит накопление ошибок округления.

Исследуем более подробно влияние последнего фактора. Для простоты рассуждений предположим, что ошибки округления при выполнении каждой отдельной операции последовательной цепочки операций являются независимыми случайными величинами, а суммарная ошибка округления всей цепочки равна сумме ошибок округления на каждом отдельном элементарном этапе вычислений. Ошибка округления на произвольном (i-м) этапе вычислений распределена равномерно в пределах от минус половины до плюс половины единицы младшего значащего разряда.

Суммарная среднеквадратичная ошибка округления σ_n (в единицах младшего значащего разряда) за n этапов определится тогда по формуле

$$\sigma_n = \sqrt{\int_{-\frac{1}{2}}^{\frac{1}{2}} \int_{-\frac{1}{2}}^{\frac{1}{2}} \dots \int_{-\frac{1}{2}}^{\frac{1}{2}} (x_1 + x_2 + \dots + x_n)^2 \, dx_1 dx_2 \dots dx_n}. \quad (4)$$

Поскольку интегралы от произведений $x_i x_j$ обращаются в нуль, то формула (4) примет вид

$$\sigma_n = \sqrt{n \int_{-\frac{1}{2}}^{\frac{1}{2}} x^2 dx} = \sqrt{\frac{n}{12}}. \quad (5)$$

Хорошо известно, что распределение сумм независимых случайных величин рассматриваемого нами вида уже для относительно небольших значений n (порядка нескольких десятков) с достаточной точностью описывается нормальным законом. Поэтому с вероятностью 0,9 суммарная ошибка округления α_n не будет превосходить удвоенной среднеквадратичной ошибки

$$|\alpha_n| \leqslant 2\sqrt{\frac{n}{12}} = \sqrt{\frac{n}{3}}. \quad (6)$$

Предположим теперь, что оперируем системой счисления с основанием a и располагаем m дополнительными разрядами для компенсации ошибок округления. В этом случае единица младшего значащего разряда в окончательном результате составит,

очевидно, a^m единиц младшего значащего разряда, сохраняемого в процессе вычислений.

Следовательно, для сохранения точности в одну единицу младшего значащего разряда в окончательном результате с вероятностью 0,9 должно выполняться неравенство

$$2\sigma = \sqrt{\frac{n}{3}} \leqslant a^m. \tag{7}$$

Поскольку a всегда больше, чем единица, то

$$m \geqslant \frac{1}{2} \log_a \frac{n}{3}. \tag{8}$$

Формула (8) и получающаяся в результате ее обращения формула

$$n \leqslant 3a^{2m} \tag{9}$$

дают возможность оценивать необходимое количество дополнительных разрядов m для компенсации ошибок округления (с вероятностью 0,9) в последовательных цепочках вычислений длины n.

В частности, полагая $m = 1$ и $a = 10$, получаем, что известное правило А. Н. Крылова о необходимости удержания одного лишнего десятичного разряда в промежуточных вычислениях дает удовлетворительные результаты, когда длина цепочек последовательных вычислений не превышает $3 \cdot 10^2 = 300$. Этот случай имеет обычно место при ручном счете, однако при применении автоматических вычислительных машин максимальная длина цепочек последовательных вычислений резко повышается. Предположим, что необходимо обеспечить точность вычислений при длине цепочек порядка трех миллионов. Тогда из формулы (8) получаем, что для компенсации ошибок округления необходимо сохранять в процессе вычислений дополнительно $\frac{1}{2} \lg 10^6 = 3$ десятичных разряда, или $\frac{1}{2} \log_2 10^6 \approx 10$ двоичных разрядов.

В машине «Киев», как известно, имеется 40 двоичных разрядов для кодирования чисел. Отбрасывая 10 разрядов, идущих на компенсацию ошибок округления, получим 30 двоичных (или около 9 десятичных) разрядов, дающих представление о возможной максимальной точности результатов. В действительности точность результатов снижается еще за счет неполного использования располагаемого числа разрядов при вычислениях с фиксированной запятой. Однако точный учет величин этого снижения возможен лишь в каждой конкретной задаче. Заметим, кроме того, что снижение точности за счет неполного использования разрядов числа может быть в значительной мере устранено при тщательном масштабировании чисел.

Программисту, ставившему на машине сколько-нибудь сложные задачи, хорошо известно, какие неудобства представляет относительно малый объем ОЗУ современных вычислительных машин. Поэтому при прочих равных условиях целесообразно стремиться к тому, чтобы сделать объем ОЗУ возможно бо́льшим. Однако не следует забывать, что увеличение объема ОЗУ приводит к усложнению коммутации, увеличивающей время переходных процессов в целях управления и снижающей быстродействие. Поэтому если руководствоваться лишь критерием максимального эффективного быстродействия, то необходимо решать задачу о нахождении оптимального объема ОЗУ.

Приведем одну из возможных методик для решения подобной задачи. Следует сразу оговориться, что любая методика неизбежно сталкивается с большим количеством неизвестных факторов, которые в настоящее время можно оценивать лишь приблизительно. Поэтому значение приводимой методики не следует преувеличивать: она может дать лишь весьма грубую оценку для оптимального объема ОЗУ.

Условимся, что будем иметь дело только с ОЗУ определенного типа, а именно с кубом на ферритовых сердечниках. Если обозначить через x емкость (количество ячеек) такого куба, то естественно предположить, что время обращения к одной ячейке выражается формулой

$$t_{\text{обр}} = a + b\sqrt{x}, \tag{10}$$

где a — константа, представляющая собой время, затрачиваемое на перемагничивание ферритов; b — константа, характеризующая время, затрачиваемое на переходные процессы в цепях управления.

Далее необходимо задаться некоторой функцией $f(x)$, выражающей среднее число операций, выполняемое с ОЗУ емкостью в x ячеек без обращения к внешним запоминающим устройствам ВЗУ. Разумеется, эта функция зависит от класса решаемых задач и поэтому не может быть определена раз навсегда. По-видимому, достаточно правильно отражает существо дела в случае универсальной машины соотношение

$$f(x) = x^2. \tag{11}$$

Необходимо еще задаться по крайней мере двумя параметрами, характеризующими ВЗУ, а именно: средним временем ожидания c и временем выборки и записи одного кода d. Кроме того, целесообразно ввести еще функцию $\varphi(x)$, равную средней относительной доле тех задач, которые могут быть решены с использованием ОЗУ емкостью в x ячеек без обращения к ВЗУ.

Что же касается самого обращения к ВЗУ, то для простоты рассуждений будем принимать, что это обращение заключается в полном обновлении содержимого ОЗУ с предварительным выводом старого его содержимого на одно из ВЗУ*. Для простоты будем также предполагать, что обращение происходит лишь к одному (начальному) месту ВЗУ при записи и еще к одному — при считывании.

Если обозначить теперь через τ среднее время выполнения одной операции в арифметическом устройстве машины, а через p — число адресов в команде, то среднее время t выполнения одной команды в машине (с учетом обращения к ВЗУ) выразится, очевидно, формулой

$$t = \{[(p+1)(a+b\sqrt{x})+\tau]f(x)+(1-\varphi(x))(2c+2dx)\}:f(x) =$$
$$= (p+1)(a+b\sqrt{x})+\tau+2\frac{1-\varphi(x)}{f(x)}(c+dx).$$

Отбрасывая константы, придем к задаче минимизации выражения

$$v = (p+1)\,b\sqrt{x}+2\frac{1-\varphi(x)}{f(x)}(c+dx). \qquad (12)$$

Решим эту задачу для трехадресной машины и примем, что

$$f(x)=x^2,\ \varphi(x)=0,$$

тогда

$$v = 4b\sqrt{x}+\frac{2c}{x^2}+\frac{2d}{x}.$$

После дифференцирования по x и приравнивания полученного выражения к нулю, придем к уравнению

$$bx^2\sqrt{x}-dx\ -2c=0. \qquad (13)$$

Выражая время в микросекундах и принимая, что время переходных процессов в цепях управления кубом на 1000 чисел равно 1 *мксек*, из формулы (10) получим, что $b\approx\frac{1}{30}$. Нетрудно понять, что уравнение (13) даст тем меньший оптимальный объем оперативной памяти, чем меньше будут коэффициенты d и c, т. е. чем более быстрым будет ВЗУ.

С целью снижения объема ОЗУ в машине «Киев» применена внешняя память на магнитных барабанах, обеспечивающая бо́льшую скорость выборки, чем память на магнитных лентах. Имеется три магнитных барабана общей емкостью в 9 000 кодов. Скорость вращения барабана 15000 *об/мин*, что соответствует среднему времени ожидания 20 *мксек*. Учитывая еще время, требующееся для переключения реле, получим среднее время

* Другое предположение о характере обмена кодами внешней памяти с ОЗУ развито в работе [1].

ожидания около 25 *мсек*, или 25 000 *мксек*. Время выборки одного кода 60 *мксек*. Подставляя эти данные в уравнение (13), придем к уравнению

$$\frac{1}{30}x^2\sqrt{x} - 60 - 50\,000 = 0. \qquad (14)$$

Последнее уравнение имеет единственный положительный корень, приближенно равный 340. Следует отметить, однако, что в приведенном расчете не учитывалась необходимость переключения трактов магнитных барабанов при бо́льших выборках, равно как и возможность малых выборок из разных мест одного и того же тракта. Учет этих обстоятельств приведет к росту оптимальной емкости ОЗУ. Наконец, если даже отбросить эти трудно учитываемые факторы и принять, что функция fx равна не x^2, а $x^{\frac{3}{2}}$, то вместо уравнения (4) придем к уравнению

$$2bx^2 - dx - 3c = 0, \qquad (15)$$

имеющему при тех же, что и раньше, значениях параметров b, c и d положительный корень, приближенно равный 1600.

Таким образом, объем оперативной памяти в 1024 числа, принятый в машине «Киев», имеет по крайней мере тот же порядок, что и рассчитанный оптимальный объем ОЗУ.

НАБОР ОПЕРАЦИЙ

При определении набора операций для цифровой вычислительной машины проектировщик должен, руководствуясь спецификой задач, которые будут решаться на машине, обеспечить максимально возможное снижение цены эффективного быстродействия при одновременном обеспечении достаточных удобств для программирования.

Поскольку априорные оценки критерия цены эффективного быстродействия оказываются, как правило, невозможными, приходится прибегать к различного рода упрощениям. Одно из самых распространенных упрощений состоит в том, что, варьируя в разумных пределах набор операций машины, стараются по возможности увеличить значение коэффициента p_0 в формуле (2). Иначе говоря, набор операций подбирается так, чтобы при решении наиболее типичных для данной машины задач обеспечивалось возможно более экономное употребление этих операций. В качестве эталона служат обычно операции некоторого фиксируемого заранее стандартного набора операций O.

Для универсальной цифровой машины, кроме того, необходимо обеспечить *универсальность набора* ее *операций*. Поскольку в понятии «универсальная машина» до сих пор наблюдается значительная путаница, прежде всего дадим его более четкое определе-

ние. С этой целью следует познакомиться в общих чертах с понятием машины Тьюринга, предложенным английским математиком Тьюрингом в 1936 г.

Машиной Тьюринга называется автомат Q, способный принимать некоторое конечное количество различных состояний q_1, q_2, ..., q_n и двигаться вправо и влево вдоль бесконечной в обе стороны ленты, разделенной на отдельные ячейки. В каждой ячейке ленты может быть записан один из конечного множества символов s_1, s_2, ..., s_m. Автомат Q снабжен считывающим устройством, способным воспринимать эти символы, и записывающим устройством, способным записать любой из этих символов в обозреваемую ячейку.

Автомат Q строго детерминирован: его поведение в каждый данный момент определяется парой символов (q_i, m_j), а именно: состоянием автомата q_i в данный момент и считываемым символом m_j (автомат способен в каждый момент обозревать лишь одну ячейку ленты). Для любой такой пары (q_i, m_j) определены следующие действия автомата Q: он записывает в обозреваемую ячейку ленты новый символ m_{j_i} (вытесняя прежний символ m_j), сдвигается вдоль ленты вправо или влево на одну ячейку и переходит в новое состояние q_{ij}. После выполнения всех указанных действий процесс начинается сначала. Обозначая через d_1 сдвиг вправо, а через d_2 — сдвиг влево, заметим, что поведение машины Тьюринга полностью определяется *начальным заполнением ленты* (начальной информацией) и *программой* машины, состоящей из конечного числа упорядоченных пятерок символов $q_i m_j m_{j_i} d_{k_{ij}} q_{ij}$, где q_i и m_i пробегают независимо друг от друга множество всех состояний автомата и соответственно множество всех символов.

В математической логике формулируется постулат, утверждающий, что любая переработка буквенной (или числовой) информации, совершаемая в соответствии с конечной системой точных правил (любой природы), может быть осуществлена некоторой, специально для этой цели построенной машиной Тьюринга.

Пусть теперь имеем автоматический преобразователь информации с программным управлением (например, автоматическую цифровую машину). Естественно называть такой преобразователь *универсальным*, если с помощью изменения программы его работы можно имитировать на нем работу любой машины Тьюринга. В силу сказанного выше на таком универсальном преобразователе можно осуществить переработку буквенной (числовой) информации в соответствии с любой системой точно сформулированных правил (алгоритмов).

Оказывается, что достаточно ввести в набор операций автоматической цифровой машины весьма небольшое количество *основных* операций, чтобы превратить ее в универсальный преобразователь информации, или, как обычно принято говорить,

в универсальную вычислительную машину. Это операции с внешними устройствами (включая внешнюю память), а также операции условного перехода, переадресации (сложения команд) и пересылки из любой ячейки памяти в любую другую ячейку (последняя операция в трехадресных машинах обычно реализуется как частный случай операции сложения, а именно сложения с нулем).

Нужно сразу оговориться, что, строго говоря, для универсальности машины необходим бесконечный объем памяти. В то же время хорошо известно, что объем оперативной памяти в современных вычислительных машинах не только не бесконечен, но весьма ограничен. Положение спасает то обстоятельство, что внешняя память в современных вычислительных машинах практически неограничена и может быть в любой момент использована для восполнения ограниченности оперативной памяти.

Условимся называть набор операций автоматической цифровой машины *алгоритмически полным*, если он обеспечивает моделирование любой машины Тьюринга. Вычислительную машину A будем называть *универсальной*, если, во-первых, набор ее операций алгоритмически полон, а, во-вторых, объем ее оперативной памяти достаточен для размещения *универсальной имитирующей программы*, позволяющей нашей машине имитировать работу любой машины Тьюринга, коль скоро ее программа будет записана в оперативной или внешней памяти машины A.

Предположим для простоты, что наша машина имеет три внешних запоминающих устройства (ВЗУ-1, ВЗУ-2, ВЗУ-3), в которых соответственно разместим программу работы имитируемой машины Тьюринга (в виде совокупности пятерок символов, подобно тому, как указывалось выше), состояние этой машины в данный момент и содержимое ее ленты.

Опишем теперь в общих чертах универсальную имитирующую программу. Основная трудность заключается в том, что как количество состояний имитирующей машины Тьюринга, так и количество различных символов на ее ленте могут быть настолько большими, что для их представления не хватит разрядности чисел, которыми оперирует моделирующая вычислительная машина A. В этом случае каждое состояние q_i машины Тьюринга будет представляться рядом чисел. Целесообразно размещать эти числа в последовательных четных ячейках ВЗУ, а нечетные ячейки использовать для размещения вспомогательных индексов, отмечающих основные числа тем или иным способом. Кроме того, необходимо использовать различные числа для обозначения состояний (q_i), символов (m_j) и характера движения ленты (d_k), причем использовать также особые числа для обозначения начала и конца последовательностей чисел, изображающих q_i и m_j.

Тогда универсальная имитирующая программа должна содержать следующие основные блоки. Первый блок, управляю-

щий последовательной (с шагом 2) выборкой чисел из ВЗУ-1, осуществляет сравнение этих чисел и переходит в случае несовпадения снова к началу ВЗУ-2 и к началу следующей последовательности q_i в ВЗУ-1. Второй блок программы начинает работать после того, как будет установлено полное совпадение последовательности q, записанной в ВЗУ-2 с одной из последовательностей q_i в ВЗУ-1. Он осуществляет выборку и сравнение последовательностей чисел с ВЗУ-3 (обозреваемый символ m) и ВЗУ-1 (символ m_j программы машины Тьюринга, следующий за выбранным состоянием q_i).

При несовпадении последовательностей управление снова передается первому блоку, а при совпадении — вступают в работу следующие блоки, осуществляющие запись в ВЗУ-3 нового символа m_{ji} из выбранной пятерки символов программы в ВЗУ-1, запись нового состояния q_{ij} (из той же пятерки) в ВЗУ-2 и переадресующие команды, относящиеся к ВЗУ-3, к началу следующей или предыдущей последовательности.

Легко видеть, что все эти блоки могут быть реализованы в командах машины «Киев» *МБП*, *МБЗ*, *МБЧ*, сложения команд и точного сравнения (а в случае использования в качестве одного из ВЗУ перфоленты — также в командах «ввод команд» и «вывод»). Вся универсальная имитирующая программа будет состоять из нескольких десятков команд и, во всяком случае, может быть размещена в ОЗУ машины «Киев».

Таким образом, согласно принятому выше определению «Киев» можно считать универсальной вычислительной машиной. Алгоритмическую полноту системы операций в ней обеспечивают операции с внешними устройствами (включая внешнюю память), сложение команд и точное сравнение. В случае размещения части программы имитируемой машины Тьюринга в ОЗУ к этим операциям необходимо прибавить еще операцию пересылки чисел из одной ячейки в любую другую (в машине «Киев» такая операция может быть осуществлена либо с помощью операции *Ф*, либо сложением с нулем). В случае размещения универсальной имитирующей программы в ПЗУ к числу операций, обеспечивающих алгоритмическую полноту набора операций, необходимо отнести еще операцию *Ф*. Все остальные операции машины «Киев» являются со строго алгоритмической точки зрения лишними. Их назначение состоит главным образом в облегчении программирования и увеличения эффективного быстродействия машины.

Заметим, что включение в число основных операций, обеспечивающих алгоритмическую полноту набора, операции точного сравнения необязательно; эту операцию можно заменить, например, *операцией обычного сравнения* (Ср.1), *применяя* ее к сравниваемой паре чисел два раза (в прямом и обратном порядке).

Дальнейшее обоснование выбора системы операций в машине «Киев» можно провести лишь в форме иллюстраций, демонстри-

рующих увеличение эффективного быстродействия машины, вызываемое наличием той или иной операции в часто встречающихся стандартных подпрограммах. Такого рода рассмотрения проводились на протяжении всего времени работы над машиной «Киев». В результате производились многократные изменения набора операций с целью его улучшения, так что окончательный набор, описанный ниже, был принят только в 1959 г. К сожалению, в процессе такого улучшения приходилось считаться с тем, что основные устройства машины были уже смонтированы, и поэтому идти только на такие улучшения, которые не сопровождались бы значительными схемными переделками. В силу этой причины окончательный набор операций не вполне удовлетворяет математиков, работавших над машиной. Большинство из выявившихся к настоящему времени пожеланий по улучшению набора операций предполагается осуществить при последней модернизации машины.

Пример. 1. Вычисление скалярного произведения двух векторов

$$a_1b_1 + a_2b_2 + \ \ldots \ + a_nb_n.$$

Для решения этой задачи в машине «Киев» достаточно четырех команд: *НГО*, $\times$, $+$ и *ОГО*. При этом команда *НГО* будет выполняться лишь один раз, а остальные — по n раз каждая Таким образом, общее число операций, необходимых для вычисления скалярного произведения двух n-мерных векторов в машине «Киев», равно $3n + 1$. При отсутствии операций *НГО* и *ОГО* потребовалось бы $4n$ операций (кроме умножения и сложения в каждом цикле вычислений необходимо было бы использовать еще операцию сложения команд и сравнения).

Пример 2. Пусть требуется выполнить некоторое множество операций O над группой чисел, начальный адрес которой заранее неизвестен и получается лишь в ходе решения задачи. С помощью набора операций машины «Киев» подобная задача легко решается благодаря возможности заполнения регистра модификации адресов по фиксатору (возможно, еще придется применить логический сдвиг, если начальный адрес группы возникает не в тех разрядах, которые переносятся на регистр модификации операцией $Ф$). При отсутствии операции $Ф$ решение этой задачи потребовало бы формирования команды *НГО* с помощью операций сложения команд и сдвига, а при отсутствии регистра модификации оказалось бы необходимым в каждом цикле переадресовывать все команды множества O.

ГЛАВА III

АДРЕСНОЕ ПРОГРАММИРОВАНИЕ И ЭЛЕКТРОННАЯ ВЫЧИСЛИТЕЛЬНАЯ МАШИНА «КИЕВ»

АДРЕСНЫЙ ЯЗЫК

В дальнейшем изложении книги будем пользоваться алгоритмическим языком, введенным для формального описания алгоритмов и получившим название адресного языка.

Адресный язык является универсальным алгоритмическим языком, весьма близким, с одной стороны, к общепринятому языку формул и, с другой стороны, — к машинным языкам в целом, в силу чего алгоритмы в этом языке могут рассматриваться как программы для универсальных машин. Близость адресного языка к языкам машинных кодов объясняется тем обстоятельством, что в нем нашли свое отражение основные алгоритмические принципы, реализованные в современных универсальных вычислительных машинах: принцип программного управления и принцип адресности.

Перевод алгоритмов с общего формального языка на язык конкретной машины можно автоматизировать (на той же машине) путем создания соответствующих программирующих программ ПП.

Идея создания подобного языка возникла в результате работы семинара по теории алгоритмов (1957—1958 гг), работавшего под руководством В. М. Глушкова, Л. А. Калужнина, В. С. Королюка и Е. Л. Ющенко.

Собственно работу по созданию языка вначале вели совместно В. С. Королюк и Е. Л. Ющенко [10]. В дальнейшем Е. Л. Ющенко [15] в язык были внесены существенные усовершенствования, часть из которых нашла известное освещение в совместной работе Б. В. Гнеденко, В. С. Королюк, Е. Л. Ющенко [5]. В настоящей книге излагается с незначительными изменениями вариант языка в таком виде, в каком он был предложен Е. Л. Ющенко в конце 1959 г.

Работа по созданию адресного языка оказала свое влияние на выбор операций для машины «Киев». Прежде всего следует назвать включенную по предложению Е. Л. Ющенко операцию *Ф*, а также в известной мере операции *НГО* и *ОГО*, в разработке которых, помимо Е. Л. Ющенко, принял участие В. С. Королюк [9]. Техническая реализация всех этих операций была осуществлена

под руководством Л. Н. Дашевского и Е. А. Шкабары. Далее Е. Л. Ющенко предложила осуществить условные передачи управления по адресу высшего ранга и, наконец, относительные передачи управления, позволяющие помещать подпрограммы на произвольных участках памяти без каких-либо изменений в их записях. Техническая реализация последнего предложения будет осуществлена в ближайшем будущем.

В адресном языке приняты все символы, которыми в математике обозначают величины, векторы, функции, множества и знаки математических операций. Однако в систему понятий (и символов) адресного языка вводятся специальные понятия (и соответствующие символы), благодаря которым он становится более приспособленным для описания алгоритмов. Адресный язык — это обычный символический язык математики (алфавитом для которого служит конечное число знаков), дополненный специальными понятиями и символами,

Строки. Адресный алгоритм состоит из строк. В каждой строке записывается одно или несколько алгоритмических действий. Запись каждого действия называется формулой; все допустимые формулы описаны дальше. Формулы в строке (если их несколько) отделяются знаком «;» (точка с запятой), либо знаком «,» (запятая). Первый из них означает недопустимость перестановки формул в строке, второй — допустимость. Строки отделяются друг от друга тем, что пишутся одна под одной (при кодировании для машины в качестве разделительного знака может ставиться «.» (точка).

Штрих-операция и адресное отображение. Штрих-операция определяет некоторую функцию одного переменного. Ее символ (штрих) ставится вверху слева от аргумента

$$'a = b,$$

где a — аргумент; b — результат операции.

Читаем: штрих a равен b (или b — содержимое a). Аргумент a называется адресом, значение функции b — содержимым адреса.

Функция $'$ (штрих) определяет некоторое отображение множества адресов A на множество содержимых B, которое будем называть адресным отображением.

Отображение множества A на множество B должно быть однозначным (одному адресу соответствует только одно содержимое). Обратное отображение может быть неоднозначным. Никакого более точного смысла в штрих-операцию не вкладывается.

В абстрактных рассмотрениях на множества A и B никаких ограничений не накладывается, т. е. штрих-функция может быть определена на произвольном множестве аргументов A и иметь значения в произвольном конечном множестве B. Однако в приложениях оба множества ограничены и являются множествами некоторых кодов.

Для обозначения элементов множества адресов используются буквы некоторого исходного (конечного) алфавита с индексами или без них, натуральные числа, а также слова, составленные из букв исходного алфавита и цифр. В качестве такого исходного алфавита примем объединение латинского, русского и греческого алфавитов из строчных и прописных букв. Тогда для обозначения адресов могут быть использованы, например, записи: a_2, 1012, $\alpha 3$, сумма, сумма 2, ω.

Если $x \overline{\in} A$, то выражение $'x$ не имеет смысла. Будем предполагать, что штрих-операция применима к тем элементам, которые используются в качестве ее аргументов.

Повторное применение штрих-операции приводит к понятию адреса второго ранга. Так, если $'a_1 = a_2$ и $a_2 \in A$, т. е. a_2 в свою очередь является адресом, то имеется такое d, при котором $'a_2 = d$. Эта зависимость символически записывается так:

$$^2a_1 = '('a_1) = ''a_1 = 'a_2 = d.$$

В этом случае говорят: a_1 — адрес второго ранга или фиксатор для d.

Аналогично вводится понятие адресов высшего ранга. Так, запись

$$^4a = d$$

означает, что имеются элементы x_1; x_2; $x_3 \in A$ такие, что $'a = x_1$; $'x_1 = x_2$; $'x_2 = x_3$; $'x_3 = d$, т. е.

$$^4a = {}^3x_1 = {}^2x_2 = 'x_3 = d.$$

Условимся d называть адресом нулевого ранга величины d.

Адресные функции. Общепринятое понятие функции и соответствующего ей выражения, построенного из символов одноместных и двухместных операций, скобок и символов величин, расширим, включив в качестве одной из допустимых элементарных операций штрих-функцию. Такие функции будем называть адресными.

При записи адресных функций предполагается, что все символы применяются правильно. Таким образом, любая функция, выражение которой не содержит знака штрих-функции, является частным примером адресной функции.

Выражения для адресных функций в языке используются для указания самостоятельных алгоритмических действий (формулы или метки безусловного перехода), а также для построения записей других алгоритмических действий (формулы засылки, обмена, предикатные, вхождения, относительного перехода, циклирования). Если задано отображение, определяющее штрих-операцию, то для любой адресной функции можно найти ее значение.

Однако адресная функция может быть написана формально, т. е. без предварительного задания отображения множества A на множество B, определяющего смысл штрих-операции. В этом случае адресная функция символизирует операции над пока еще неизвестным содержимым выражений, входящих в нее под символом $'$. Если теперь задать отображение, определяющее штрих-операцию (причем в множество A входят все символы, стоящие в функции под этим знаком), то функция приобретает некоторое определенное значение.

Операция засылки. Для определения и изменения адресного отображения применяется операция засылки, символом которой является $\Longrightarrow$ (стрелка, соединяющая два элемента). Запись операции $d \Longrightarrow a$ (d заслать по адресу a) означает, что:

1) элемент a включается в множество адресов A;

2) элемент d включается в множество содержимых B;

3) устанавливается соответствие $'a = d$;

4) все ранее установленные соответствия вида $'x = y$, где $x \neq d$, остаются неизменными.

Таким образом, *после* засылки $d \Longrightarrow a$ изменяется значение всех адресных функций, в которые входят выражения $'a$, ^2a, ^3a и т. д., а все адресные функции, в которые не входят эти выражения, остаются неизменными.

Метки. Меткой может быть цифра, буква, слово из цифр и букв с индексами или без них.

Метками могут быть значения адресных функций. Поэтому метки рассматриваются как частный случай адресной функции, значение которой постоянно и является меткой.

Меченые строки. Для указания порядка применения строк алгоритма те или иные из них могут отмечаться одной или несколькими (различными) метками. В этом случае метка с последующим знаком «...» (троеточие) ставится слева от отмечаемой строки. Троеточие в записи алгоритма всегда означает, что слева от него стоит метка, отмечающая данную строку. Соответствующие строки алгоритма называются меченными.

Переходим к перечислению допустимых алгоритмических действий и описанию соответствующих им формул языка.

Формулы вычисляемого перехода. Запись адресной функции, значениями которой могут быть метки (частный случай — просто метка), может обозначать отдельное алгоритмическое действие, называемое формулой вычисляемого перехода. Такое алгоритмическое действие состоит в переходе к выполнению строки алгоритма, помеченной меткой, равной значению этой адресной функции, вычисленному при данном адресном отображении.

В частном случае, когда адресная функция является просто меткой, формула называется *меткой безусловного перехода.* Таким образом, метка (без последующего за ней троеточия) может со-

ставлять отдельную строку алгоритма и в этом случае обозначаемое ею действие состоит в переходе к строке, помеченной такой же меткой.

Выполнение формулы вычисляемого перехода (или метки безусловного перехода) не изменяет адресного отображения.

Формулы останова. Употребляется два специальных символа Я и !, называемые соответственно формулами относительного и безусловного останова. Употребление символа Я связано с формулами вхождения; символ ! означает действие — конец алгоритма. Символ Я также может означать конец алгоритма, если ему не предшествовала соответствующая формула вхождения.

Формулы останова либо составляют отдельные строки алгоритма, либо входят как составные части в другие формулы адресного языка (см. формулы предикатные, вхождения, циклирования, замены).

Формула относительного перехода. Выражения вида

$$\updownarrow N$$

могут составлять отдельные строки алгоритма и называются формулами относительного перехода. Здесь N — *адресная функция*, значения которой — положительные или отрицательные целые числа или просто целое отличное от нуля число; $\updownarrow$ — специальный символ.

Формула относительного перехода является записью следующего алгоритмического действия: совершается переход к строке, расположенной выше или ниже данной на число строк, равное значению N, вычисленному при заданном адресном отображении: ниже — если оно положительно, выше — в противном случае.

Формулы засылки. Выражения вида

$$f_1 \Rightarrow f_2$$

могут составлять отдельные строки алгоритма и называются формулами засылки. Здесь f_1 и f_2 — адресные функции.

Формула засылки символизирует следующую алгоритмическую операцию: значение адресной функции f_1 засылается по адресу, равному значению функции f_2. Таким образом, выполнение формулы засылки изменяет адресное отображение и тем самым значение ряда адресных функций. При этом понимается, что все сказанное при определении операции засылки применяется к значениям функций f_1 и f_2. Необходимо отметить, что определение формулы засылки следует понимать в том смысле, что $'f_2$ после выполнения формулы засылки равно значению f_1 до засылки. Это важно, если в выражение f_1 входит $'f_2$, $^2f_2$ и т. д.

Допустима также запись адресных формул в виде

$$\varnothing \Rightarrow a,$$

где $\varnothing$ означает символ «пусто» (место для помещения фактических параметров). При этом предполагается, что в алгоритме к

моменту обращения к адресной формуле с «пустой» левой частью будет определено, какой символ должен стоять на ее месте.

Формула обмена. Выражения вида

$$f_1 \langle = \rangle f_2$$

могут составлять отдельные строки алгоритма и называются формулами обмена. Здесь f_1 и f_2 — адресные функции. Формула обмена является записью следующей алгоритмической операции: при заданном адресном отображении вычисляются значения адресных функций f_1 и f_2, которые воспринимаются как адреса. Действие состоит в обмене содержимыми этих адресов (последние к моменту выполнения действия должны быть определены). Содержимые остальных адресов остаются неизменными. Так, формула

$$a \langle = \rangle c$$

(где a и c — адреса) означает действие, равносильное действиям, записанным с помощью последовательности трех формул засылки:

$$'a = \rangle r$$
$$'c = \rangle a$$
$$'r = \rangle c$$

В общем случае, когда f_1 и f_2 — произвольные адресные функции, это действие равносильно следующим четырем последовательно выполняемым формулам засылки:

$$a = \rangle r$$
$$'c = \rangle r_1$$
$$'a = \rangle c$$
$$'r_1 = \rangle r$$

Здесь r и r_1 — свободные относительно данного алгоритма (рабочие) адреса.

Список формул. В одной строке алгоритма может записываться несколько формул засылки или обмена. В этом случае между ними ставится один из знаков: «;» (точка с запятой) или «,» (запятая). Первый из этих знаков означает недопустимость перестановки порядка выполнения разделенных им действий, второй — возможность перестановки или одновременного выполнения этих действий. Подобная строка называется списком формул и означает выполнение всех перечисленных в ней действий с указанными замечаниями относительно порядка их выполнения.

Список формул с переходом. В конце списка формул засылки или обмена, приведенного в одной строке (или после одной такой формулы), после знака «;» может ставиться одна из формул вычисляемого или относительного перехода. Подобная строка называется списком формул с переходом и, как и в предыдущем

случае, означает выполнение всех перечисленных в ней действий, последним из которых является переход.

Формулы вхождения и подпрограммы. Формулы вхождения применяются для записи того или иного преобразования, которое по тем или иным причинам не выписываем в данном месте. Такое преобразование называется подпрограммой, и предполагается, что оно описано некоторым образом (не обязательно в адресном языке). Требуется, однако, чтобы в описании подпрограммы были указаны:

1) упорядоченный список входных и выходных параметров;

2) момент окончания вычислений;

3) метка, присвоенная данной подпрограмме (название).

К числу таких параметров относятся, помимо данных об аргументах, размерностях их массивов и т. д., метки тех подпрограмм, использование которых определяется основной программой. Так, например, если подпрограммой является алгоритм вычисления определенного интеграла по некоторой схеме, то, помимо границ интегрирования, шага и т. д., к входным данным будет относиться указание метки подпрограммы вычисления значений подынтегральной функции.

Допускается запись подпрограмм в адресном языке. В этом случае требования к их записям состоят в следующем:

1) начальная (входная) строка подпрограммы метится меткой, служащей ее названием;

2) начальная строка подпрограммы состоит из адресных формул с «пустыми» левыми частями (вместо них в ходе выполнения подпрограммы ставятся элементы списка по формуле вхождения). Порядок выполнения этих формул не играет роли, но их запись упорядочена и согласована со списком формулы вхождения;

3) в подпрограмме используется по крайней мере один символ Я.

Формула вхождения подпрограммы в алгоритм

$$П\alpha\,(a_1,\ \ldots,\ a_n)\,\beta,$$

где $П$ — символ формулы вхождения; α — формула (или метка) вычисляемого перехода; β — формула (или метка) вычисляемого или относительного перехода или одна из формул останова; $a_1, \ldots, a_n$ — список адресных функций.

Формула означает:

а) перейти к подпрограмме с меткой α (или равной значению α);

б) первые выражения из списка $a_1, \ldots, a_n$ считать упорядоченным списком аргументов подпрограммы;

в) выполнив подпрограмму, переслать упорядоченный список результатов в остальные адреса списка $a_1, \ldots, a_n$ формулы $П$. Таким образом, число членов этого списка должно быть равно сумме чисел входных и выходных параметров данной подпрограммы;

г) перейти к строке алгоритма с меткой β (или равной значению β).

Для подпрограммы, заданной в адресном языке, формула вхождения означает переход к строке с меткой α, включение (с сохранением порядка) списка адресных выражений $a_1, \ldots, a_n$ на место знаков $\varnothing$ левых частей формул первой строки подпрограммы и замену символа Я меткой β (все замены — не в записи алгоритма, а лишь в ходе его выполнения).

Соответствие между формулами вхождения и формулами останова устанавливается, как и соответствие между открывающими и закрывающими скобками в алгебраических формулах.

Подпрограмма называется областью действия формулы вхождения.

Предикатные формулы. Пусть L обозначает некоторое высказывание. Выражения вида

$$P\{L\}\,\alpha\downarrow\beta$$

могут составлять отдельные строки алгоритма и называются предикатными формулами, где P — символ предикатной формулы; $\downarrow$ — разделительный знак; α и β — верхнее и нижнее значения предикатной формулы и каждое из них может быть одной из описанных ранее строк.

Соответствующее предикатной формуле алгоритмическое действие состоит в выполнении строки, представляющей ее верхнее значение, если соответствующее высказывание истинно; нижнее значение — в противном случае.

Если значение L всегда истина, то вместо предикатной формулы можно записать ее верхнее значение, а если оно всегда ложь — ее нижнее значение. В этом смысле любая из приведенных ранее строк является частным случаем предикатной формулы.

Если одним из значений предикатной формулы является метка следующей за ней строки, то последняя может опускаться (для нижнего значения — вместе с символом $\downarrow$).

Заметим, что с помощью только формул останова !, засылки и частного случая предикатных формул, значения которых — метки, можно записать любой алгоритм. Все остальные формулы вводятся для удобства.

Формулы замены. Выражения вида

$$З\{a_1 \to c_1, \ \ldots, \ a_n \to c_n\}\,\alpha,\ \beta$$

называются формулами замены, где $З$ — символ формулы; α, β — формулы вычисляемого или относительного перехода, или просто метки; $a_1 \to c_1, \ \ldots, \ a_n \to c_n$ — список замен.

Алгоритмическое действие, обозначенное формулой замены, состоит в следующем:

а) при заданном адресном отображении определяются метки — значения формул α и β; строки алгоритма между этими метками определяют область действия формулы замены;

б) выполняются строки алгоритма, составляющие область дей-

ствия формулы замены с заменой в них в процессе выполнения символов c_i соответствующими символами a_i;

в) возможен выход из области действия формулы замены в результате ее исчерпания (с переходом к следующей строке алгоритма) или вследствие действия какой-либо формулы перехода; как в том, так и в другом случае при выходе из области действия восстанавливается в ней первоначальная запись алгоритма.

Если α — метка следующей за формулой замены строки, то она в записи формулы замены может опускаться. Формула в этом случае приобретает вид

$$З\{a_1 \to c_1, \ \ldots, \ a_n \to c_n\}, \beta.$$

Если область действия формулы замены состоит из одной следующей за ней строки, то в записи формулы обе формулы α и β опускаются.

Формулы замены позволяют ввести в употребление еще одну удобную запись для подпрограмм. При этом вместо строки формул засылки с «пустой» левой частью (см. формулы вхождения) подпрограмма начинается формулой замены, в списке замен которой вместо левых частей стоит тот же символ $\varnothing$.

Подпрограммы с первой строкой из адресных формул будем называть подпрограммами с засылкой. Их запись остается неизменной даже в процессе работы подпрограммы. Подпрограммы с первой строкой, содержащей формулу замены, будем называть подпрограммами с заменой (с переадресацией). Их запись в процессе работы изменяется, ибо в этом случае одни символы заменяются другими. После окончания работы запись восстанавливается. Допускаются смешанные подпрограммы.

Формула циклирования. В конечных упорядоченных множествах вводится понятие операции следования. Символ C означает «следующий». В каждом конкретном случае операция следования задается некоторым образом. Не накладывая на операцию следования никаких ограничений, потребуем, чтобы применение ее к любому элементу множества давало бы (и притом однозначно) следующий элемент.

Элемент информации называется обозреваемым алгоритмом, если запись алгоритма содержит его адрес некоторого ранга. Схемами обозревания элементов информации называются алгоритмы, включающие в своей записи адреса некоторого ранга этих элементов.

Естественно, что операция следования в общем случае не может быть установлена непосредственно в множестве элементов исходной информации. Операция следования для элементов исходной информации может устанавливаться с помощью штрих-операции по операции следования в множестве адресов. Выбор множества адресов будем считать находящимся в нашей воле. Задача программирования состоит в построении схем обозревания

массивов с помощью операции следования в массивах их адресов.

Упорядоченное множество запишем в виде

$$\{a,\ C\varnothing, P\{L\}\},$$

где a — первый элемент; C — операция следования (на место символа $\varnothing$ может ставиться любой элемент множества); L — условие принадлежности элемента множеству.

Предположим, что в некотором алгоритме требуется выполнить операцию Φ последовательно над всеми элементами множества $\{a, C\varnothing, P\{L\}\}$ и перейти к строке с меткой l. Адресная программа такого алгоритма может быть записана в виде

$$\begin{array}{ll} & a \Longrightarrow \pi \\ K \ \ldots & P\{L\} \downarrow l \\ & \Phi('\pi) \\ & C'\pi \Longrightarrow \pi \\ & K \end{array}$$

Для сокращения записи подобных схем вводится понятие формул циклирования по адресу π

$$Ц\{a,\ C\varnothing,\ P\{L\} \Longrightarrow \pi\}\,\alpha,\ l \quad \Phi('\pi)$$

или формулы циклирования по параметру π

$$Ц\{a,\ C\varnothing,\ P\{L\} \to \pi\}\,\alpha,\ l \quad \Phi(\pi),$$

где $Ц$ — символ формулы, α — метка последней строки преобразования Φ.

Вводится понятие области действия формулы циклирования. К области действия конкретной формулы циклирования относятся строки алгоритма, следующие за формулой вплоть до строки с меткой α. Если в область действия формулы циклирования входит другая формула циклирования, замены или вхождения, то область действия последней входит в область действия первой.

Строка с формулой циклирования может быть помечена и к ней могут быть переходы от других строк алгоритма. Устанавливается, что если к формуле циклирования совершен переход не из области ее действия, то перебор элементов начинается с первого; если же переход осуществлен из области действия, то перебор продолжается.

Для множеств с операцией следования, определяемой соотношением $C\varnothing = \varnothing + b$, употребляется запись

$$\{a\,(b)\,P\,(L)\} \quad \text{или} \quad \{a\,(b)\,c\},$$

где c — последний элемент множества.

Формулы циклирования в этом случае имеют вид

$$Ц\{a(b)P(L) \Longrightarrow \pi\}$$

или

$$Ц\{a(b)c \rightarrow \pi\} \text{ и т. д.}$$

Если область действия формулы циклирования состоит из одной строки или совпадает с областью действия записанной в следующей строке формулы циклирования, то метка α опускается

$$Ц\{\ldots\},\ \beta.$$

Если β — метка строки, непосредственно следующей за областью действия этой формулы, то она также может опускаться

$$Ц\{\ldots\}\alpha \quad \text{или} \quad Ц\{\ldots\}.$$

В формуле циклирования может быть предусмотрен одновременный перебор нескольких множеств, например,

$$Ц\{a_1(b_1)c_1 \rightarrow \pi_1;\ a_2(b_2)c_2 \Longrightarrow \pi_2\}\alpha,\ \beta.$$

В этом случае перебор продолжается, пока верны все указанные условия принадлежности, а в формуле может быть указано только одно из них.

Допускается также циклирование по одному адресу в множестве, где для начальной группы элементов задана одна операция следования, для следующей — другая и т. д. Например,

$$Ц\{a_1(b_1)c_1;\ a_2(b_2)c_2 \Longrightarrow \pi\}\alpha,\ \beta.$$

Формальное определение адресного алгоритма. Адресный алгоритм состоит из исходного адресного отображения и конечного числа строк, записанных одна под другой, и указания множества адресов результативного отображения. В каждой строке может стоять одна из формул: вычисляемого перехода (или метка безусловного перехода); останова; относительного перехода (или метка относительного перехода); засылки; обмена; вхождения; предикатная; замены; циклирования, а также списков формул, или формул с переходом. Любая строка алгоритма может быть помечена.

Начальное адресное отображение задается в виде последовательностей равенств

$$'a = b,$$

где a — адрес; b — содержимое.

Результативное адресное отображение задается перечислением адресов, содержимые которых после окончания работы алгоритма представляют решение задачи.

Порядок выполнения алгоритма определяется следующими правилами:

1. Первой выполняется строка, помеченная специально указан-

ной начальной меткой. Если такого указания нет, то первой выполняется строка, стоящая первой в записи.

2. Порядок выполнения строк, содержащих списки формул, указан при их описании.

3. Формулы вычисляемого и относительного переходов, списки формул с переходом, формулы вхождения, замены и циклирования, а также предикатные формулы, значение которых содержит формулы перехода, указывают себе строку-преемницу.

4. После выполнения строки, не указывающей своей преемницы, переходим к следующей в записи строке.

5. Формула безусловного останова! или формула относительного останова Я, которая не соответствует никакой формуле вхождения, означает конец работы алгоритма.

Содержательно адресный алгоритм представляет собой некоторую переработку информации о задаче, в результате которой получается ее решение. Эта переработка имеет форму установления и изменения некоторого адресного отображения.

Информация о задаче задается некоторым адресным отображением. Поэтому во многих случаях алгоритм начинается установлением некоторого исходного адресного отображения посредством ряда засылок или же таблицей такого отображения.

Результатом работы алгоритма является содержимое адресов, указанных в качестве результативного множества.

УРОВНИ И СТИЛИ АДРЕСНОГО ЯЗЫКА

Адресный язык, основанный на принципе адресности, не предъявляет, однако, никаких требований к упорядоченности множества адресов. Адресное отображение может задаваться на произвольном множестве адресов. Для задач, по существу своему связанных с упорядоченностью элементов, могут вводиться операции следования, алгоритмически даже не описанные. Уровень упорядоченности адресов, алгоритмизации введенных в них операций следования, составляет в известном смысле уровень языка.

Необходимо отметить три основных уровня адресного языка [15, 18]:

1. Общеалгоритмический уровень, на котором принимается наиболее естественное для данной конкретной задачи множество адресов и операции следования, описываются общематематическими средствами, например с помощью индексов. На этом уровне адресный язык наиболее близок к Алголу [13].

2. Уровень условных (или символических) адресов, требующий упорядочения адресов элементов, составляющих массивы, обрабатываемые алгоритмом. На этом уровне учитывается линейность и ограниченность множества адресов, согласующихся с техническими возможностями машин; обычно, если это не оговаривается, речь идет о языке уровня условных адресов.

3. Уровень конкретных адресов, предполагающий полностью упорядоченное множество адресов в связи с конкретной машиной с учетом множества истинных адресов ячеек и их глубины.

В самом адресном языке заложена возможность перехода от уровня к уровню, от самого абстрактного алгоритмического языка до уровня полного распределения адресов для данной машины. Этим определяется особенная пригодность адресного языка в качестве входного языка для программирующих программ. Отдельные уровни могут служить в качестве промежуточных внутренних языков программирующих программ [ПП], осуществляющих перевод записи алгоритма в машинную программу постепенно, через промежуточные записи.

Свободное использование изобразительных средств языка позволяет записывать один и тот же алгоритм различными способами. Для облегчения перевода со свободного адресного языка на язык конкретной машины могут быть определены стили языка.

Стиль языка предполагает такие ограничения в применении его выразительных средств, которые делают перевод на язык данной машины более простым. Перевод с общего языка в определенный его стиль возможен на любом уровне. Задачей ПП является перевод с общего адресного языка. Однако этот перевод может быть разбит на этапы введением ряда стилей, все более и более приближающихся к машинному языку. Таким образом, работа ПП сводится в основном к формальным преобразованиям внутри адресного языка, к переводу из стиля в стиль.

Широта и гибкость адресного языка позволяют в зависимости от того, насколько сложную ПП мы в состоянии осуществить, выбрать точку, до которой программы задач необходимо доводить вручную, с тем чтобы дальнейшую их обработку проводила ПП. Благодаря этому уже сейчас практически возможно осуществить автоматизацию программирования на некоторых его участках. Так, внедрение ПП с входным адресным языком на машине «Киев»— ПП-АК (см. далее), как и ПП с тем же входным языком на других машинах («Урал», УМШН — управляющей машины широкого назначения и др.), повысило производительность программистов в несколько раз. Наличие этих ПП благодаря общности входных языков позволяет при необходимости без особых затруднений переносить решение задач с одной машины на другую и расширяет круг лиц, составляющих программы (в адресном языке) для машин из числа людей, занимающихся непосредственной алгоритмизацией тех или иных процессов переработки информации и даже вовсе незнакомых с машинными кодами. Последнее обстоятельство следует считать особо важным ввиду возрастающих потребностей в кадрах, вызванных ростом числа используемых машин и их быстродействия.

Таким образом, работу, которую обычно выполняет квалифицированный программист, удается разделить на две существенно

различные части. Труд по уточнению и составлению алгоритма, требующий высокой математической культуры, выполняется инженером-программистом. Результат этого труда — адресная программа. Работа по переводу последней в машинный код осуществляется программирующими программами. Кодирование адресных программ (а иногда и перевод в необходимый стиль), как и работа на машинах, выполняется операторами со средней квалификацией. Перспективы внедрения читающих автоматов обещают свести этот труд до минимума.

ПРИМЕР СОСТАВЛЕНИЯ АДРЕСНОЙ ПРОГРАММЫ

Составим алгоритм решения системы линейных алгебраических уравнений с симметричной матрицей коэффициентов усовершенствованным методом Гаусса [14].

Пусть $A = \{a_{ij}\}$ $(i, j = 1, 2, \ldots, n; a_{ij} = a_{ji})$ — исходная матрица и $F\{a_{i, n+1}\}$ — вектор правых частей.

Согласно избранному методу алгоритм распадается на два этапа — прямой ход и обратный ход.

Прямой и обратный ходы

Прямой ход. Элементы a_{ij} преобразуются последовательно по строкам по рекурентным формулам:

$$\bar{a}_{1j} = a_{1j} \; (1 \leqslant j \leqslant n+1);$$

$$\bar{a}_{ij} = a_{ij} - \sum_{k=1}^{i-1} \frac{\bar{a}_{ki}\,\bar{a}_{kj}}{\bar{a}_{kk}} \; (2 \leqslant i \leqslant n; \; i \leqslant j \leqslant n+1).$$

Обратный ход — решение системы уравнений с треугольной матрицей $\{\bar{a}_{ij}\}$ $(i = 1, 2, \ldots, n; \; i \leqslant j \leqslant n+1)$, полученной в результате выполнения прямого хода.

Расчетные формулы имеют вид

$$x_i = \frac{\bar{a}_{i, n-1} - \sum_{k=i+1}^{n} a_{ik}x_k}{\bar{a}_{ii}} \; (i = n, \; n-1, \; \ldots, \; 1).$$

1. Составление программ на общеалгоритмическом уровне. Примем в качестве исходного отображения

$$'a_{ij} = a_{ij} (i = 1, \; 2, \; \ldots, \; n; \; 1 \leqslant j \leqslant n+1).$$

Пусть s_i — адреса, отведенные для компонент вектора решений $x_i (i = 1, \; 2, \; \ldots, \; n)$. Пусть элементы $\bar{a}_{ij} (1 \leqslant i \leqslant n; \; i \leqslant j \leqslant n+1)$ вычисляются по строкам слева направо. Тогда нетрудно заметить, что вновь получаемым элементам $\bar{a}_{ij}$ могут быть поставлены в соответствие те же адреса α_{ij}.

Алгоритм прямого хода

Исходное адресное отображение

$'\alpha_{ij} = a_{ij}$
$(i = 1, 2, \ldots, n;$
$j = 1, 2, \ldots, n+1)$

Адресный алгоритм 1

ПРЯМОЙ ХОД ...
$Ц\{2(1)n \to i\}, Я$
$Ц\{i(1)n+1 \to j\}$
$Ц\{1(1)i-1 \to k\}$
$'\alpha_{ij} - \frac{'\alpha_{ki} \cdot '\alpha_{kj}}{'\alpha_{kk}} \Longrightarrow \alpha_{ij}$

Результативное множество адресов совпадает с исходным.

Алгоритм обратного хода

Исходное адресное отображение

$'\alpha_{ij} = \bar{a}_{ij}$
$(i = 1, 2, \ldots, n;$
$j = 1, 2, \ldots, n+1)$

Адресный алгоритм 2

ОБРАТНЫЙ ХОД ...
$Ц\{n(-1)1 \to i\} M, Я$
$\alpha_{i,\,n+1} \Longrightarrow s_i$
$Ц\{i+1(1)n \to k\}$
$'s_i - '\alpha_{ik} \cdot s_k \Longrightarrow s_i$
$M \ldots 's_i : '\alpha_{ii} \Longrightarrow s_i$

Результативное множество адресов

$$s(i = 1, 2, \ldots, n).$$

2. Для перехода на уровень условных адресов необходимо алгоритмически описать операции следования в множестве адресов, описанные в программах прямого и обратного хода с помощью индексов. В нашем случае это сводится к построению алгоритма определения в линейно упорядоченном множестве адресов адреса элемента матрицы по его заданным индексам.

Алгоритм нахождения адреса элемента по его индексам будет определяться способом кодирования исходной информации.

В связи с наличной схемой обработки информации и симметрией исходной матрицы напрашивается идея кодирования в последовательность по столбцам либо по строкам лишь тех элеменнов матрицы, для которых $j \geqslant i$. Для определенности будем считать, что имеет место кодировка по столбцам. Например, для $n = 4$ мыслится следующая схема размещения исходной информации по адресам:

$'(\alpha+1) = a_{11}$	$'(\alpha+2) = a_{12}$ $'(\alpha+3) = a_{22}$	$'(\alpha+4) = a_{13}$ $'(\alpha+5) = a_{23}$ $'(\alpha+6) = a_{33}$	$'(\alpha+7) = a_{14}$ $'(\alpha+8) = a_{24}$ $'(\alpha+9) = a_{34}$ $'(\alpha+10) = a_{44}$	$'(\alpha+11) = a_{15}$ $'(\alpha+12) = a_{25}$ $'(\alpha+13) = a_{35}$ $'(\alpha+14) = a_{45}$

Порядковый номер элемента информации a_{ij} в последовательности

$$\alpha + 1,\ \alpha + 2,\ \ldots,\ \alpha + \frac{n^2+n}{2} + n$$

будем называть его приведенным индексом $\pi(i,\ j)$.

Нетрудно убедиться в том, что

$$\pi(i,\ j) = i + \sum_{s=1}^{j-1} s = i + \frac{j}{2}(j-1).$$

Чтобы сделать программы не зависимыми от параметров n (размерности матрицы) и α (начала массива, в котором размещены элементы), примем еще

$$'\alpha = n;\ '\varphi = \alpha.$$

Программа прямого хода

Исходное адресное отображение

$'\alpha = n$
$'\varphi = \alpha$
$'(\alpha + i + \frac{j}{2}(j-1))' = a_{ij}$
$(1 \leqslant i \leqslant n;\ i \leqslant j \leqslant n+1)$

Адресный алгоритм 3

ПРЯМОЙ ХОД ...
$Ц\{2\,(1)''\varphi \to i\}$, Я
$Ц\{i\,(1)''\varphi + 1 \to j\}$
$Ц\{1\,(1)\,i - 1 \to k\}$

$$'('\varphi + i + \frac{j}{2}(j-1)) - \frac{'\left('\varphi + k + \frac{i}{2}(i-1)\right) \times \left('\varphi + k + \frac{j}{2}(i-1)\right)}{'\left('\varphi + \frac{k}{2}(k+1)\right)} \Longrightarrow$$

$$\Longrightarrow '\varphi + i + \frac{j}{2}(j-1)$$

Результативное множество адресов совпадает с исходным.

Для возможности получения результата в программе обратного хода — вектора решений — в произвольном массиве адресов введем адрес ψ

$$'\psi = s.$$

Программа обратного хода

Исходное адресное отображение	Результативное множество адресов
$'\varphi = \alpha$ $'\alpha = n$ $'(\alpha + i + \frac{j}{2}(j-1)) = a_{ij}$ $(1 \leqslant i \leqslant n;\ i \leqslant j \leqslant n+1)$	$s + i\,(i = 0, 1, 2, \ldots, n)$

Адресный алгоритм

$$\text{ОБРАТНЫЙ ХОД} \ldots$$
$$Ц\,\{''\varphi(-1)\,1 \to i\}\,M,\ Я$$
$$'\left('\varphi + i + \frac{''\varphi + 1}{2}\,''\varphi\right) \Longrightarrow '\psi + i$$
$$Ц\,\{i + 1\,(1)''\varphi \to k\}$$
$$'('\psi + i) - '\left('\varphi + i + \frac{k}{2}(k-1)\right)'('\psi + k) \Longrightarrow '\psi + i$$
$$M \ldots \frac{'('\psi + i)}{'\left('\psi + \frac{i(i+1)}{2}\right)} \Longrightarrow '\psi + i$$

Каждый из приведенных алгоритмов можно оформить в виде подпрограммы. Это имеет смысл, поскольку, например, программу обратного хода можно использовать при решении системы линейных уравнений в общем случае по методу исключений и т. д.

Для получения подпрограмм при принятом способе кодирования информации к задаче остается только к указанным программам прибавить вначале соответственно строки, к которым перенесены начальные метки:

$$\text{ПРЯМОЙ ХОД} \ldots \varnothing \Longrightarrow \varphi$$
$$\text{ОБРАТНЫЙ ХОД} \ldots \varnothing \Longrightarrow \varphi,\ \varnothing \Longrightarrow \psi$$

Формулы вхождения на эти подпрограммы будут иметь вид

$$П\quad \text{ПРЯМОЙ ХОД}\,\{\alpha\}$$
$$П\quad \text{ОБРАТНЫЙ ХОД}\,\{\alpha, s\}$$

Повторяем, как запись формул вхождения, так и запись алгоритмов согласуются со способом кодирования информации.

АДРЕСНЫЕ АЛГОРИТМЫ И МАТЕМАТИЧЕСКИЕ МАШИНЫ

Современные универсальные цифровые машины с математической точки зрения основаны на двух принципах: адресности и программного управления.

Принцип адресности состоит в том, что запись алгоритмических действий в машинном коде осуществляется с помощью указания адресов, в которых хранятся элементы преобразуемой информации. Согласно принципу программного управления при решении каждой конкретной задачи управление работой машины осуществляется непосредственно алгоритмом ее решения, представленным на языке данной машины в виде специальной программы.

В адресной записи алгоритмов отражаются эти общие для современных вычислительных машин принципы адресности и программного управления. С другой стороны, любой алгоритм описывается конечным числом операций. Следовательно, в принципе может быть построена машина ИАВМ (идеальная адресная вычислительная машина), для которой данный адресный алгоритм, как и все алгоритмы, составленные из тех же действий и использующие те же адреса, является программой [17]. С этой целью в набор элементарных операций машины надо включить все встречающиеся в алгоритме действия и обеспечить кодировку команд, соответствующую адресной записи.

Реальные универсальные вычислительные машины отличаются от ИАВМ, помимо ограниченности памяти, тем, что набор их элементарных операций ограничен небольшим числом, хотя и является полным, т. е. позволяющим (путем их последовательного применения) построение любой алгоритмической операции.

Ради простоты записи в качестве адресов разных машин используется начальный отрезок чисел натурального ряда, длина которого характеризует объем оперативного запоминающего устройства машины. Однако в множестве ячеек, используемых для хранения информации в машине, существенное место занимают программные регистры. При описании операций, связанных с этими регистрами, последним во избежание смешения следует присваивать наименования-адреса, отличные от числовых адресов ячеек ЗУ. Заметим, что в машинных кодах указания об использовании программных регистров либо включаются в код операции, либо в сочетании с ним кодируются на специально отведенных разрядах.

Моделирование работы любой реальной вычислительной машины на ИАВМ сводится к выбору из ее элементарных операций только тех, которые входят в набор данной реальной машины. Порождаемый при этом язык представляет собой некоторый стиль адресного языка и адресная программа, записанная при соблюдении требований данного стиля, для превращения в программу реальной машины нуждается только в перекодировке. В этом смысле язык любой конкретной машины является адресным.

С другой стороны, любую элементарную операцию ИАВМ можно воспроизвести некоторой последовательностью операций — программой любой реальной универсальной вычислительной машины. Разумеется, при переходе от адресной программы к програм-

ме конкретной машины необходимо учитывать объем машинной памяти (и ее линейность), а также наличие тех или иных программных регистров.

Анализ адресных программ показывает, что адресное программирование сводится к построению тех или иных схем обозревания информации [10]. Использование понятия адресов высших рангов и схем обозревания информации по ним значительно упрощает задачу программирования и позволяет записывать алгоритмы (их адресные программы) в не изменяемом в процессе их выполнения виде и не зависящем от параметров частных задач (в отличие от записей программ с переадресацией команд).

Произвольный адресный алгоритм можно представить в эквивалентной ему (в смысле результативности) форме, в которой все входящие в запись алгоритма *адресные функции имеют ранг не выше второго* [16]. Однако алгоритмы, описываемые с помощью адресных функций только первого ранга и не включающие формулы замены (переадресации), представляют собой весьма узкий класс алгоритмов [11]. Таким образом, включение в набор для алгоритмической полноты операции обращения по адресу второго ранга или замены обязательно.

Необходимо отметить, что первая из этих операций обладает известными удобствами по сравнению со второй в связи с тем, что пользоваться программами, не изменяющимися в процессе своей работы, практически удобнее. Кроме того, такие программы легко осуществимы схемно в виде блоков или укрупненных элементарных операций, например для матричных операций и вообще для решения задач линейной алгебры, для реализации различных методов решения дифференциальных уравнений и пр.

Передача информации на многоместные операторы при схемной их реализации или на подпрограммы может показаться невозможной без фиксации параметров (размерностей векторов, матриц, порядков систем и пр.); однако, приняв стандартные способы задания информации и используя операцию обращения по адресу второго ранга, получаем возможность передавать всю необходимую информацию о массивах сложной структуры независимо от значений этих параметров с помощью одного *ведущего* адреса. Более подробно об этом будет сказано в следующей главе.

Разработка математических требований к машине «Киев» шла параллельно с работой по созданию и развитию адресного языка. Идея построения сменно-спаянной памяти (выдвинутая Е. Л. Ющенко), в которой в виде спаянных блоков могли бы храниться стандартные программы для реализации тех или иных алгоритмов, была первоначально осуществлена с помощью групповых операций *НГО* и *ОГО*. Эта же идея послужила толчком к выработке понятия адреса высшего (второго) ранга и вместе с ним первоначального понятия адресного алгоритма, впервые сформу-

лированного в [10]. Анализ ряда адресных алгоритмов, в том числе алгоритмов программирующих программ, привел к идее групповой операции по адресу второго ранга *Ф*, которая и была включена в набор операций и в дальнейшем получила широкую популярность среди программистов. Этому способствовало также дальнейшее развитие адресного языка, при практическом применении которого на машине «Киев» существенная роль принадлежит именно операции *Ф*.

Предложения, вытекающие из анализа адресных алгоритмов, были высказаны тогда, когда схемы машины были полностью разработаны и уже частично осуществлялся ее монтаж. В частности, это касается предложенных Е. Л. Ющенко операции *Ф*, операции передачи управления по содержимому адреса, относительных передач управления и др. Стремление к простоте технических решений и наличие уже разработанных схем позволили реализовать пока только некоторые из них, хотя схемные решения получены полностью.

АДРЕСНОЕ ОПИСАНИЕ МАШИНЫ «КИЕВ»

Приведем описание набора операций машины «Киев» в адресном языке.

Введем обозначения для программных регистров и тумблеров машины «Киев»:

С — счетчик команд — устройство ввода команд (УВК), 11-разрядный;

К — регистр команд (РК), 41-разрядный;

Р — регистр возврата (РВ), 11-разрядный;

Ц — регистр циклов (РЦ), 10-разрядный;

А — регистр модификации адресов — счетчик адреса (СчА), 10-разрядный;

Тр — регистр-триггер аварийного останова (1-разрядный);

Тб — тумблер блокировки аварийного останова ($'Тб = 1$, если тумблер включен, и $'Тб = 0$ — в противном случае).

При описании операций, включающих работу этих регистров, ограничимся их буквенными обозначениями — адресами, которые будем предполагать отличными от кодов адресов внутреннего запоминающего устройства (ВЗУ).

Согласно принятому принципу программного управления отдельный цикл работы машины состоит в выполнении команды $'К$, код которой хранится к данному моменту на регистре команд *К*, и в засылке на этот регистр кода очередной команды, т. е. команды, которую надлежит выполнить в следующем цикле.

Для хранения номера (адреса) очередной команды в машине «Киев» используется счетчик команд *С*. Выполнение команды $'К$ зависит от ее содержания (кода).

Для удобства описания набора элементарных операций, выполняемых отдельными командами машины «Киев», выделим следующие части регистра команд K, соответственно обозначив их:

A_0 — регистр адреса кода операции, 41-й—37-й разряды;

E_1 — разряд признака модифицируемости I адреса, 36-й разряд;

A_1 — регистр I адреса, 35-й—25-й разряды;

E_2 — разряд признака модифицируемости II адреса, 24-й разряд;

A_2 — регистр II адреса, 23-й—13-й разряды;

E_3 — разряд признака модифицируемости III адреса, 12-й разряд;

A_3 — регистр III адреса, 11-й—1-й разряды.

В связи с числом отведенных разрядов на указанных регистрах могут храниться соответственно восьмеричные коды:

$$'A_0 = 00,\ 01,\ \ldots,\ 37;$$
$$'E_1,\ 'E_2,\ 'E_3 = 0,1;$$
$$'A_1,\ 'A_2,\ 'A_3 = 0000,\ 0001,\ \ldots,\ 3777.$$

В зависимости от $'A_0$ (кода операции) машина выполняет следующие операции:

Основные арифметические операции

1) $'A_0 = 01$. Операция сложения $+$. Реализуется следующая адресная программа:

$$a\ \ldots\ '('A_1 + 'E_1'A) + '('A_2 + 'E_2'A) \Longrightarrow 'A_3 + 'E_3'A$$
$$P\{'|'A_3 + 'E_3'A| \geqslant 1\} \downarrow 'C + 1 \Longrightarrow C,\ б$$
$$P\{'T_б = 1\}'C + 2 \Longrightarrow C \downarrow !$$
$$б\ \ldots\ {}^2C \Longrightarrow K$$

Строка с меткой a означает, что при $'E_i = 1$ соответствующий адрес модифицируется, т. е. увеличивается на величину содержимого регистра модификации адресов A, и при $'E_i = 0$ операция выполняется непосредственно по адресу. Следующая строка означает проверку выхода результата из разрядной сетки и переход к следующей по номеру команде ($'C + 1 \Longrightarrow C$), если выход не имеет места. При выходе результата из разрядной сетки согласно второй предикатной формуле производится проверка состояния тумблера аварийного останова и при включенном тумблере — остановка машины, а при выключенном — пропускается одна команда ($'C + 2 \Longrightarrow C$). Последнее используется для автоматического вмешательства в вычислительный процесс (перемасштабирование и др.). Эти особенности выполнения команд относятся ко всем операциям данной группы.

Аналогично выполняются следующие две операции:

2) $'A_0 = 02$. Вычитание —.

3) $'A_0 = 12$. Деление :. Операция деления на целые степени 2 равносильна арифметическому сдвигу, так как деление выполняется без округления.

4) $'A_0 = 10$. Умножение без округления $\times$. Реализуется адресная программа

$$'('A_1 + 'E_1'A) \times '('A_2 + 'E_2'A) \Longrightarrow 'A_3 + 'E_3'A$$
$$'C + 1 \Longrightarrow C$$
$$^2C \Longrightarrow K$$

5) $'A_0 = 11$. Умножение с округлением $\boxed{\times}$ выполняется аналогично предыдущей операции.

Вспомогательные арифметические операции

6) $'A_0 = 03$. Сложение команд *СлК*.

$$|'('A_1 + 'E_1'A) + |'('A_2 + 'E_2'A)| \vee \operatorname{sign}'('A_2 + 'E_2'A) \Longrightarrow 'A_3 + 'E_3'A$$
$$'C + 1 \Longrightarrow C$$
$$^2C \Longrightarrow K$$

Операция *СлК* отличается от операции сложения тем, что здесь к модулю содержимого адреса $'A_2 + 'E_2'A$ (кода команды) прибавляется содержимое адреса $'A_1 + 'E_1'A$ (константа переадресации) и результату присваивается знак $\operatorname{sign}'('A_2 + 'E_2'A)$ («знак» команды).

7) $'A_0 = 06$. Операция вычитания модулей «$|-|$».

8) $'A_0 = 07$. Циклическое сложение $Ц +$ — сложение кодов с гашением выхода из старшего разряда. Как и в предыдущих операциях, допускается возможность модификации адресов.

Логические операции

После каждой из операций этой группы совершается переход к следующей по номеру команде.

9) $'A_0 = 13$. Сдвиг логический $Л \rightarrow$. Код $'('A_1 + 'E_1'A)$ (включая знаковый разряд) сдвигается на число разрядов, указанное в III адресе кода, имеющего своим адресом $('A_2 + 'E_2'A)$, вправо, если $'('A_2 + 'E_2'A) < 0$, или влево — в противном случае; результат помещается по адресу $'A_3 + 'E_3'A$.

10) $'A_0 = 35$. Нормализация H. Число $'('A_1 + 'E_1'A)$ нормализуется, порядок числа помещается в младшие шесть разрядов по адресу $('A_2 + 'E_2'A)$, мантисса — по адресу $'A_3 + 'E_3'A$. Кроме того,

$$'C + 1 \Longrightarrow C$$
$$^2C \Longrightarrow K$$

11) $'A_0 = 14$. Операция *поразрядного* логического сложения $\vee$

$$'('A_1 + 'E'A) \vee '('A_2 + 'E_2'A) \Rightarrow 'A_3 + 'E_3'A$$
$$'C + 1 \Rightarrow C$$
$$^2C \Rightarrow K$$

Аналогично выполняются следующие две операции:

12) $'A_0 = 15$. Операция поразрядного логического умножения $\wedge$.

13) $'A_0 = 17$. Поразрядная логическая операция неравнозначности $\cong$.

Операции передачи управления

Все операции передачи управления не изменяют содержимого ВЗУ; результат их действия касается лишь некоторых специальных регистров.

14) $'A_0 = 16$. Операция условной передачи управления по равенству *Ср3*. Реализуется программа

$$P\{'('A_1 + 'E_1'A) = '('A_2 + 'E_2'A)\}'A_3 + 'E_3'A \Rightarrow C \downarrow 'C + 1 \Rightarrow C$$
$$^2C \Rightarrow K$$

15) $'A_0 = 04$. Условная передача управления по соотношению «меньше или равно» *Ср1*

$$P\{'('A_1 + 'E_1'A) \leqslant '('A_2 + 'E_2'A)\}'A_3 + 'E_3'A \Rightarrow C \downarrow 'C + 1 \Rightarrow C$$
$$^2C \Rightarrow K$$

16) $'A_0 = 05$. Условная передача управления по соотношению «меньше или равно без учета знаков» *Ср2*

$$a \ldots P\{|'('A_1 + 'E_1'A)| \leqslant |'('A_2 + 'E_2'A)|\}'A_3 +$$
$$+ 'E_3'A \Rightarrow C \downarrow 'C + 1 \Rightarrow C$$
$$C \Rightarrow K$$

17) $'A_0 = 31$. Условная передача управления по знаку числа *УПЧ*

$$P\{'('A_1 + 'E_1'A) \leqslant -0\}'A_3 + 'E_3'A \Rightarrow C \downarrow 'A_2 + 'E_2'A \Rightarrow C$$
$$^2C \Rightarrow K$$

18) $'A_0 = 30$. Условный переход на подпрограмму *УПП*

$$P\{'('A_1 + 'E_1'A) \leqslant -0\}'A_3 + 'E_3'A \Rightarrow C;$$
$$'A_2 + 'E_2'A \Rightarrow P \downarrow C + 1 \Rightarrow C$$
$$^2C \Rightarrow K$$

Таким образом, если указанное условие выполнено, производится замена содержимого регистра возврата P на величину $'A_2 + 'E_2A$ и переход к выполнению приказа, номер которого

указан в III адресе. При этом $'A_3 + 'E_3'A$ является адресом начального приказа подпрограммы, а $'P$ — номером приказа, которому должно быть передано управление после выполнения подпрограммы. При невыполнении этого условия производится переход к следующему по номеру приказу программы.

Соответствующая подпрограмма в этом случае заканчивается специальным приказом *ПРВ*.

19) $'A_0 = 32$. Переход по регистру возврата *ПРВ*. По этой команде выполняются операции:

$$'P \Longrightarrow C$$
$$0 \Longrightarrow P$$
$$^2C \Longrightarrow K$$

Содержимое регистров A_1, A_2, A_3 на результат операции не влияет.

Операции обращения к внешним устройствам

Все операции данной группы являются групповыми, т. е. относятся к последовательностям кодов.

20) $'A_0 = 20$. Ввод чисел *B1*. В ячейки оперативного запоминающего устройства (ОЗУ) с адресами

$$'A_1,\ 'A_1 + 1,\ \ldots,\ 'A_2$$

вводятся коды, предварительно переведенные из десятично-двоичной системы в двоичную. Условно запишем эту групповую операцию в виде

$$'(ПЛ)_{\text{переведен в двоичный код}} \Longrightarrow \begin{pmatrix} 'A_1 \\ 'A_2 \end{pmatrix}$$

Кроме того, выполняются засылки:

$$'C + 1 \Longrightarrow C$$
$$^2C \Longrightarrow K$$

21) $'A_0 = 21$. Ввод команд *B2* выполняется, как и операция *B1*, только передача кодов производится без преобразования:

$$'(ПЛ) \Longrightarrow \begin{pmatrix} 'A_1 \\ 'A_2 \end{pmatrix}$$
$$'C + 1 \Longrightarrow C$$
$$^2C \Longrightarrow K$$

22) $'A_0 = 22$. Вывод кодов *Печ*. Обратная операция: выводится содержимое ячеек $'A_1,\ 'A_1 + 1,\ \ldots,\ 'A_2$. Кроме того,

$$'A_3 \Longrightarrow C$$
$$^2C \Longrightarrow K$$

23) $'A_0 = 23$. Обмен кодами ОЗУ с внешним ЗУ (МБ) в режиме запись — *МБЗ*. Коды, содержащиеся в последовательности ячеек ОЗУ

$$'A_1,\ 'A_1 + 1,\ \ldots,\ 'A_2,$$

переносятся на МБ:

$$\begin{pmatrix} 'A_1 \\ 'A_2 \end{pmatrix} \Longrightarrow МБ;$$

кроме того,

$$'A_3 \Longrightarrow C$$
$$^2C \Longrightarrow K$$

24) $'A_0 = 24$. Обмен кодами ОЗУ с МБ в режиме чтение—*МБЧ*. Коды с МБ передаются в последовательность ячеек ОЗУ $'A_1$, $A_1 + 1$, ..., $'A_2$, т. е.

$$'(МБ) \Longrightarrow \begin{pmatrix} 'A_1 \\ 'A_2 \end{pmatrix};$$

кроме того,

$$'A_3 \Longrightarrow C$$
$$C \Longrightarrow K$$

25) $'A_0 = 25$. Подготовительная операция для операций *МБЗ* и *МБЧ*, обеспечивающая надлежащую подводку магнитного барабана *МБП*. Здесь $'A_1$ равно 1 для операций 23 и равно 0 — для операции 24; $'A_2 = n \cdot 2^{-22}$; $n = 1,\ 2,\ 3$ — номер МБ, с которым производится обмен кодами; $'A_3$ — номер числа на МБ, с которого необходимо начать соответствующую операцию. Кроме того, выполняются действия

$$'C + 1 \Longrightarrow C$$
$$^2C \Longrightarrow K$$

26) $'A_0 = 33$. Останов.

Операции модификации адресов

Операции модификации адресов являются групповыми в том смысле, что формируемое ими содержимое регистра модификации может быть использовано группой команд.

27) $'A_0 = 26$. Начало групповой операции (*НГО*). По операции *НГО*:

а) на регистр циклов *Ц* засылается число, характеризующее число циклов в циклическом процессе,

$$'A_1 \Longrightarrow Ц;$$

б) на регистр модификации адресов A засылается константа переадресации

$$'A_2 \Longrightarrow A;$$

в) реализуется предикатная формула

$$P\{'Ц = 'A\}\,'A_3 \Longrightarrow C \downarrow 'C + 1 \Longrightarrow C;$$

г) $^2C \Longrightarrow K$.

Таким образом, команда *НГО* подготавливает содержимое регистра модификации A и тем самым обеспечивает надлежащую модификацию переменных адресов.

Если заранее известно число циклов N и шаг переадресации p, то полагаем

$$'A_1 = 'A_2 + Np.$$

При каждом повторении цикла в этом случае содержимое регистра A увеличивается на величину p. Пока $'Ц \neq 'A$, продолжаются вычисления по циклу; при $'Ц = 'A$ совершается выход с цикла на команду с номером $'A_3$.

28) $'A_0 = 27$. Конец групповой операции *ОГО*. По этой команде:

а) содержимое регистра A увеличивается на шаг переадресации $'A_1 + 'A \Longrightarrow A$;

б) реализуется предикатная формула

$$P\{'Ц = 'A\}\,'A_3 \Longrightarrow C \downarrow 'A_2 \Longrightarrow C;$$

в) $^2C \Longrightarrow K$.

Здесь $'A_3$ — номер команды, которой передается управление по окончанию цикла; $'A_2$ — номер команды, которой передается управление при продолжении вычислений по циклу. (Заметим, что $'A_2$ не является номером команды *НГО*, так как последняя теперь не повторяется при переходе от цикла к циклу, поскольку ее повторение привело бы к восстановлению начального заполнения регистра A).

29) $'A_0 = 34$. Операция обращения по адресу второго ранга (вызов по фиксатору) *Ф*.

Помимо операции *НГО*, обеспечивающей заполнение регистра модификации адресов, в машине реализована операция *Ф*, которая также выполняет эту функцию. Однако, в то время как в команде *НГО* величина, засылаемая на регистр A, задавалась в явном виде (как $'A_2$), в команде *Ф* эта величина задается лишь своим адресом. По команде *Ф* выполняются операции:

$$'('A_1 + 'E_1'A)_{II} \Longrightarrow A$$
$$'('('A_1 + 'E_1'A)_{II}) \Longrightarrow 'A_3 + 'E_3'A$$
$$'C + 1 \Longrightarrow C$$
$$^2C \Longrightarrow K$$

($'A_2$ при выполнении команды не используется).

Здесь обозначено через $'\alpha_{II}$ содержимое II адреса ячейки α; при $'A_1 = \alpha$; $'\alpha_{II} = \beta$ по операции Φ имеем:

$$\beta \Longrightarrow A$$
$$'\beta \Longrightarrow 'A_3$$
$$'C + 1 \Longrightarrow C$$
$$^2C \Longrightarrow K$$

Наличие операции Φ позволяет для произвольных циклических параметрических процессов составлять программы, не изменяющиеся в процессе своей работы (без команд переадресации).

ГРУППОВЫЕ ОПЕРАЦИИ МАШИНЫ «КИЕВ»

Вопросы контроля правильности программ при решении задач на ЦАМ играют первостепенную роль. Если для решения одной и той же задачи в соответствии с выбранным методом могут быть предложены две различные программы, то с точки зрения удобства их ввода в машину и скорости отладки преимущество должно быть отдано той из программ, которая занимает меньшее количество ячеек памяти, т. е. содержит информацию о задачах в более компактном виде. Предпочтение такой программе может быть отдано даже в том случае, если непосредственный счет по ней требует несколько большей затраты времени. Более выгодно пользоваться программой с бо́льшим числом ячеек памяти, если ее команды могут быть размещены в пассивной памяти как более удобной с точки зрения контроля и ввода.

Экономия в командах программ часто может быть достигнута за счет включения в набор элементарных операций машины команды переадресации и упорядоченного размещения величин в последовательности ячеек памяти.

В общем случае программы включают схемы обозревания ряда последовательностей или таблиц, т. е. являются сложными циклическими процессами, зависящими от параметров.

Известны некоторые приемы (заçылка в стандартные адреса, перенос из стандартных адресов в последовательность и др.), позволяющие иногда сокращать программы. Можно ставить вопрос о сокращении циклических программ, но не за счет удачного программирования, а за счет схемной реализации в машине некоторых специальных команд.

Специальные команды, при которых информация о выполнении циклических программ задается в компактном виде с помощью сокращенного числа кодов, называют *групповыми*. Введением групповых операций может преследоваться также цель использования пассивной памяти машины.

Остановимся на более полном описании групповых операций машины «Киев».

I. Пусть цикличная адресная программа содержит группу адресов, меняющихся от цикла к циклу по закону

$$\alpha_0 + \alpha + ip, \qquad (16)$$

где i — номер цикла; α_0 — исходное значение переменного адреса; α — величина начального (первого) сдвига адреса; p — шаг переадресации.

Для обозревания информации, размещенной на различных участках памяти, возможность выполнения *начального сдвига* на величину α, отличную от щага переадресации, представляет существенное удобство. Таким образом, необходимой информацией для цикла, зависящего от параметра, будет:

а) набор вычислительных операций цикла в исходном виде;

б) указания на переменные адреса;

в) пара чисел (α, p), определяющих параметр цикла.

Информация а) может быть задана с помощью соответствующих вычислительных команд. Для распознавания переменных адресов, т. е. для выяснения необходимости образования адресов по правилу (16), в каждом из адресов машины «Киев» используется дополнительный (12-й) разряд. С этой целью введены две операции: *НГО* — начало групповой операции и *ОГО* — конец групповой операции. Первая из них может употребляться самостоятельно, другая — при использовании операции *Ф* или *НГО*.

Операция *НГО* кодируется следующим образом:

НГО	*M*	α	*k*

Здесь *НГО* — код операции; *M* — число, равное $\alpha + Np$, где N — общее количество повторений цикла (если для окончания цикла используется какое-либо другое условие, то I адрес заполняется кодом 11...1); α — величина начального сдвига имеет указанное выше значение, k — номер некоторой команды, которой передается управление после окончания цикла (для подпрограмм может указываться адрес постоянно-спаянной ячейки 3146, содержащей код команды перехода по регистру возврата).

Функциональное назначение операции *НГО*:

1) константа *M*, характеризующая общее число циклов, засылается на регистр циклов *Ц*;

2) константа α — величина начального сдвига переменных адресов — засылается на регистр адресов *А*;

3) содержимые регистра адресов и регистра циклов сравниваются, и при их совпадении управление передается команде, адрес которой k находится в III адресе команды *НГО*, т. е. $'A_3 \Longrightarrow C$; в противном случае управление передается следующей по номеру команде, т. е. $'C + 1 \Longrightarrow C$.

Изменение шага переадресации — содержимого регистра адреса — может быть реализовано двумя способами:

а) путем переадресации команды *НГО* за счет соответствующего изменения ее II адреса. В этом случае программа будет содержать переменную команду *НГО*; на каждом из циклов предусматривается повторное выполнение команды *НГО* и тем самым готовится новое значение константы модификации переменных адресов. Этот способ удобен для кодирования сложных циклических процессов;

б) путем использования команды *ОГО*, которая кодируется в виде

ОГО	$4000 + p$	k_1	k_2

Здесь *ОГО* — код операции; p — шаг переадресации (в I адресе указывается шаг p, увеличенный на 4000); k_1, k_2 — номера некоторых команд.

По команде *ОГО*:

1) шаг переадресации p прибавляется к содержимому регистра адреса: $p + {}'A \Longrightarrow {}'A$ (т. е. ${}'A_1 + {}'A \Longrightarrow A$);

2) содержимое регистра адресов и регистра циклов сравниваются и при полном их совпадении управление передается команде, адрес которой k_2 указан в III адресе команды *ОГО* (т. е. ${}'A_2 \Longrightarrow C$), в противном случае — команде, адрес которой k_1 указан во II адресе этой команды (т. е. ${}'A_3 \Longrightarrow C$).

Важно заметить, что при использовании команды *ОГО* программой предусматривается выполнение команды *НГО* на первом цикле; при повторном выполнении цикла команда *НГО* не повторяется и остается неизменной.

Таким образом, использование операции *ОГО*, наряду с операцией *НГО*, позволяет кодировать циклические программы, включающие параметры, в постоянной памяти, что особенно удобно для циклических процессов с восстановлением, а также в случае наперед известного количества циклов. Эти операции широко применяются в стандартных подпрограммах.

Для схемной реализации описанных процессов в устройстве управления предусмотрены: регистр циклов, регистр адреса, сумматор адреса и устройство совпадения.

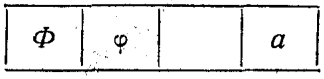
При помощи сумматора производится сложение адресов, указанных в команде, с содержимым регистра адреса, а во время выполнения команды *ОГО* — прибавление шага переадресации к содержимому регистра адреса.

II. Легко заметить, что при кодировании сложных циклических процессов наличие одного регистра адреса позволяет при помощи операции *ОГО* кодировать только внутренние циклы, которые в свою очередь не включают операций *Ф* или *ОГО*.

Внешние циклы необходимо кодировать при помощи переадресации команды *НГО*.

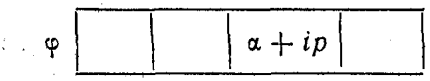
Для широкого использования сменно-спаянной памяти, реализованной в виде блоков, которые по необходимости могут включаться (или выключаться) во внутреннюю пассивную память машины, в машине «Киев» предусмотрена операция группового характера *Ф*, которая позволяет вводить любую программу в пассивную память.

Операция *Ф* кодируется в виде



Здесь *Ф* — код операции; φ, a — адреса некоторых ячеек.

Операция *Ф*, как и операция *НГО*, подготавливает регистр адреса *А*, который далее используется для формирования переменных адресов. Однако в отличие от операции *НГО*, в которой константа переадресации кодировалась в явном виде, величина $a + ip$ теперь кодируется во II адресе ячейки, номер которой φ указан в I адресе команды *Ф*



$('\varphi_{II} = a + ip)$,

т. е. $'\varphi_{II} \Longrightarrow A$.

Кроме того, при выполнении операции *Ф* в адрес a поступает код, содержащийся в ячейке $a + ip$, номер которой находится во II адресе ячейки φ, т. е. $''\varphi_{II} \Longrightarrow {}'A_3$. Поскольку теперь величина групповой константы переадресации явно не входит в программу, последняя не меняет своего вида. Изменение константы переадресации сводится теперь к изменению содержимого некоторого адреса φ. II адрес команды *Ф* не используется.

Таким образом, внешние циклы в сложных циклических процессах можно кодировать как с помощью операции *НГО* с последующей переадресацией последней, так и в постоянной памяти с помощью операции *Ф*. В связи с тем что код φ в I адресе команды *Ф* можно рассматривать как адрес, который «фиксирует» величину сдвига по последовательности некоторых адресов, операцию *Ф* принято называть операцией обращения по адресу второго ранга.

Первая из функций операции *ОГО* — изменение содержимого регистра адреса — может быть использована также в связи с операцией *Ф*.

Операция *Ф* реализуется следующим образом: адрес второго ранга φ передается на регистр адреса в оперативную память; содержимое этого адреса $a + ip = '\varphi_{II}$ (адрес первого ранга $'\varphi_{II}$) из оперативной памяти передается снова на регистр команд устройства управления (в его свободный II адрес); далее с реги-

стра команд код $'\varphi_{II}$ снова поступает на регистр адреса ОЗУ. Содержимое этого адреса $'('\varphi_{II})$ (содержимое по адресу второго ранга) выходит из ОЗУ и задерживается там на регистре числа. Код из III адреса команды Φ поступает на ОЗУ, и по этому адресу происходит запись содержимого регистра числа $'('\varphi_{II})$.

Пример 1. Главный элемент n-квадратной матрицы, элементы которой расположены (по строкам или столбцам) по адресам $\alpha+1$, $\alpha+2$, ..., $\alpha+n^2$, поместить в ячейку с номером γ.

Программа для машины «Киев» будет иметь вид

$N+1$	$+$			γ
$N+2$	*НГО*	M	α	
$N+3$	$\|\leqslant\|$	4001	γ	$N+5$
$N+4$	$+$	4001		γ
$N+5$	*ОГО*	4001	$N+3$	N_1

Здесь $M=\alpha+n^2$; знаком 4000 в командах $N+3$ и $N+4$ отмечено наличие единицы в 12-м разряде адреса (признак групповой операции); N_1 — адрес команды, которой передается управление после окончания работы данной программы.

Пример 2. Вычислить сумму

$$S=\sum_{j=1}^{m}\sum_{i=1}^{n}\frac{a_ix_j+b_iy_j}{c_ix_j+d_iy_j}.$$

Пусть заданные величины помещены по следующим адресам:

$\alpha+1$	$\alpha+2$	...	$\alpha+n$	$\beta+1$	...	$\beta+n$	$\gamma+1$	...	$\gamma+n$
a_1	a_2	...	a_n	b_1	...	b_n	c_1	...	c_n

$\delta+1$	...	$\delta+n$	$\varphi+1$	...	$\varphi+m$	$\varphi+m+1$	...	$\varphi+2m$
d_1	...	d_n	x_1	...	x_m	y_1	...	y_m

Программа будет иметь вид

$N+1$	$+$			γ
$N+2$	$НГО$	$\varphi + m$	φ	k
$N+3$	$СлК$	δ	$N+2$	$N+2$
$N+4$	$+$	4001		ω_1
$N+5$	$+$	$4001 + m$		ω_2
$N+6$	$НГО$	n		
$N+7$	$\times$	$4001 + \alpha$	ω_1	τ_1
$N+10$	$\times$	$4001 + \beta$	ω_2	τ_2

$N+11$	$+$	τ_1	τ_2	τ_1
$N+12$	$\times$	$4001 + \delta$	ω_2	ρ_1
$N+13$	$\times$	$4001 + \gamma$	ω_1	ρ_2
$N+14$	$+$	ρ_1	ρ_2	ρ_1
$N+15$	$:$	τ_1	ρ_1	τ_1
$N+16$	$+$	τ_1	γ	γ
$N+17$	$ОГО$	4001	$N+7$	$N+2$

Здесь k — адрес команды, которой передается управление после окончания работы программы; в ячейке с адресом δ помещен код

		0001	

Искомую сумму получаем по адресу γ.

Пример 3. Последовательность адресов

$$'b + 0, \quad 'b + 1, \quad \ldots, \quad 'b + 35$$

содержит коэффициенты семи полиномов, из которых: первые полиномы — первой степени, вторые — второй и т. д. Коэффициенты многочленов размещены в порядке убывания их степеней. Значения всех полиномов вычисляются по схеме Горнера при фиксированном значении переменной, содержащейся по адресу ξ, и помещаются в последовательность адресов

$$'B + 0, \quad 'B + 1, \quad \ldots, \quad 'B + 6,$$

содержимые которых и составляют результат работы алгоритма.

Адресную программу можно записать в следующем виде:

$$0 \Longrightarrow \delta B, \quad 0 \Longrightarrow {}'b; \quad 1 \Longrightarrow \Delta 1; \quad 2 \Longrightarrow \Delta 0$$

$$M \ldots 0 \Longrightarrow 'B + '\delta B$$

$$M1 \ldots {}' ({}'B + {}'\delta B) \times {}'\xi + {}' ({}'b + {}'\delta b) \Longrightarrow {}'B + {}'\delta B$$
$${}'\delta b + 1 \Longrightarrow \delta b$$
$$P \{{}'\delta b \leqslant {}'\Delta 1\} M1$$
$${}'\Delta 0 + 1 \Longrightarrow \Delta 0; \quad {}'\Delta 1 + {}'\Delta 0 \Longrightarrow \Delta 1, \quad {}'\delta B + 1 \Longrightarrow \delta B$$
$$P \{{}'\delta B \leqslant 6\} M\downarrow !$$

Остановимся на программировании с помощью операции Φ второй и третьей строк, содержащих адресные функции второго ранга. Всего под знаком вторичной штриховки имеется четыре адреса B, δB, b, δb. Поскольку эти адреса встречаются в алгоритме лишь в сочетании ${}'B + {}'\delta B$ и ${}'b + {}'\delta b$, то указанные две строки удобно заменить следующими строками:

$$M \ldots {}'B + {}'\delta B \Longrightarrow r_1$$
$$0 \Longrightarrow {}'r_1$$
$$M1 \ldots {}'b + {}'\delta b \Longrightarrow r_2$$
$${}''r_2 \Longrightarrow r_2$$
$${}''r_1 \times {}'\xi + {}'r_2 \Longrightarrow {}'r_1$$

или в кодах машины «Киев»

$M \ldots$	+	B	δB	r_1
	Φ	r_1		
	+			4000
$M1 \ldots$	+	b	δb	r_2
	Φ	r_2		r_2
	Φ	r_1		
	$\times$	4000	ξ	r_3
	+	r_3	r_2	4000

Здесь r_1, r_2, r_3 — рабочие ячейки.

ПРЕДСТАВЛЕНИЕ АДРЕСНЫХ ФУНКЦИЙ НА МАШИНЕ «КИЕВ»

Введем понятие ранга R адресной функции $f : R(f)$.

1. Адресная функция a, не содержащая штрих-операции, имеет ранг R, равный нулю:

$$R(a) = 0.$$

2. Пусть ранг функции f равен s. Тогда ранг функции $'f$ равен $R(f)+1=s+1$, т. е.

$$R('f)=R(f)+1.$$

3. Любая функция $O_1(x)$ от одного переменного x, отличная от штрих-операции, имеет ранг, равный рангу ее аргумента:

$$R(O_1(x))=R(x).$$

4. Пусть $R(a)=s_1$, $R(b)=s_2$. Ранг функции $a\theta b$, где θ — любая арифметическая или логическая операция от двух аргументов, равен максимальному из рангов s_1 и s_2:

$$R(a\theta b)=\max(R(a), R(b)).$$

Произвольный адресный алгоритм можно представить в эквивалентной ему (в содержательном смысле) форме, в которой все входящие в его запись адресные функции имеют ранг не выше второго.

Например, адресные формулы

$$^{n}\alpha \Longrightarrow \beta \quad (n \geqslant 3)$$

и

$$a \Longrightarrow {}^{m}\beta \quad (m \geqslant 2)$$

можно записать соответственно в виде

$$\left.\begin{array}{l} {}^{2}\alpha \Longrightarrow \beta \\ {}^{2}\beta \Longrightarrow \beta \\ \vdots \\ {}^{2}\beta \Longrightarrow \beta \\ {}^{2}\beta \Longrightarrow \beta \end{array}\right\} n-3 \text{ раза} \qquad \left.\begin{array}{l} {}^{2}\beta \Longrightarrow \omega \\ {}^{2}\omega \Longrightarrow \omega \\ \vdots \\ {}^{2}\omega \Longrightarrow \omega \\ a \Longrightarrow {}'\omega \end{array}\right\} m-2 \text{ раза}$$

Здесь и далее ω — некоторый рабочий адрес, содержимое которого не влияет на работу алгоритма.

Заметим, кроме того, что любую адресную формулу

$$a \Longrightarrow b,$$

где a, b — адресные функции соответственно рангов n и m, можно заменить эквивалентной ей программой из двух строк

$$a \Longrightarrow \omega$$
$${}'\omega \Longrightarrow b$$

Отличительная особенность представления адресных функций на машине «Киев» состоит в следующем способе выполнения операций по адресам второго, высших и нулевого рангов:

1) компоненты любой (арифметической или логической) операции, представляющие собой адресные функции второго и высших рангов, формируются лишь с помощью A-регистра;

Таблица 1

Представление адресных функций в кодах машины «Киев»

Адресная программа	Программа в кодах «Киева»			
$'(a+\delta_1'd)\,\theta$ $'(b+\delta_2'd) \Longrightarrow$ $\Longrightarrow c+\delta_3'd$	34	d		—
	θ	$\delta_1 \cdot 4000+a$	$\delta_2 \cdot 4000+b$	$\delta_3 \cdot 4000+c$
$'(a+\delta_1'd)\,\theta$ $'(b+\delta_2'd) \Longrightarrow$ $\Longrightarrow c+\delta_3'd$ $''d \Longrightarrow l$	34	d		l
	θ	$\delta_1 \cdot 4000+a$	$\delta_2 \cdot 4000+b$	$\delta_3 \cdot 4000+c$
$'(a+\delta_1 d)\,\theta$ $'(b+\delta_2 d) \Longrightarrow$ $\Longrightarrow c+\delta_3 d$	*НГО*	—	d	—
	θ	$\delta_1 \cdot 4000+a$	$\delta_2 \cdot 4000+b$	$\delta_3 \cdot 4000+c$
$d \Longrightarrow A$	*НГО*	—	d	—
$'A+p \Longrightarrow A$	*ОГО*	$4000+p$	—	—
$^2a\theta^2b \Longrightarrow c$	34	a		r
	34	b		—
	θ	r	4000	c
	или			
	34	a		r_1
	34	b		r_2
	θ	r_1	r_2	c

Адресная программа	Программа в кодах «Киева»				
$'('a + b)\,\theta$ $'('c + d) \Longrightarrow b$	34	a		—	
	01	$4000 + b$		r_1	
	34	c		—	
	01	$4000 + d$		r_2	
	θ	r_1	r_2	b	
Формирование содержимого A-регистра по формуле $b + 'a + ('c + d) \Longrightarrow A$	34	c		—	
	01	$4000 + d$		r	
	34	a		—	
	27	$4000 + b$	—	—	
	34	$4000 + r$		—	
$^n a \Longrightarrow c$	34	a		c	
	34	c		c	$n-2$ раза
	⋮	⋮		⋮	
	34	c		c	
$'('('a + a) + b) \Longrightarrow c$	34	a		—	
	01	$4000 + a$		c	
	34	c		—	
	01	$4000 + b$		c	

Адресная программа	Программа в кодах «Киева»			
$'('\alpha + '\beta) \Longrightarrow c$	01	α	β	c
	34	c		c
$'('\alpha + '\beta + a) \Longrightarrow c$	01	α	β	c
	34	c		—
	01	$4000 + a$		c
Формирование содержимого A-регистра по формуле $c + 'a'b$ ($'3010 = 2^{-16}$) ($'3040 = 2^{-12}$)	12	a	3010	r
	12	b	3040	r_1
	10	r	r_1	r
	34	r		—
	27	$4000 + c$	—	—
$'a\theta'\beta \Longrightarrow '('a + b) + c$	34	a		—
	34	$4000 + b$		—
	θ	α	β	$4000 + c$
$'a\theta'\beta \Longrightarrow 'a\theta_1'b + c$	θ_1	a	b	r
	34	r		—
	θ	α	β	$4000 + c$

Продолжение табл. 1

Адресная программа	Программа в кодах «Киева»			
$'\alpha\theta'\beta \Longrightarrow {}^{n}a$	34	a		r
	34	r		r
	⋮			
	34	r		r
	θ	α	β	4000

$n-2$

2) адреса нулевого ранга могут быть только компонентами операции сложения с содержимым A-регистра или с нулем, результат которой помещается по A-регистру.

Основные соответствия между адресными функциями и программами в кодах машины «Киев» приведены в табл. 1, где обозначены через a, b, c целые неотрицательные восьмеричные числа $\leqslant 3777$ (адреса нулевого ранга), а через δ_1, δ_2, δ_3 — числа, равные нулю или единице; многоточие в столбце команд означает, что в этом месте может стоять любое число команд, не изменяющих содержимого A-регистра; прочерк в адресе команды означает, что для данной операции содержимое этого адреса не используется; θ — символ прозвольной (арифметической или логической) элементарной операции; A — регистр модификации (A-регистр).

АЛГОРИТМ ФОРМАЛЬНОГО ПЕРЕВОДА АДРЕСНЫХ ФУНКЦИЙ В КОДЫ МАШИНЫ «КИЕВ»

Специфика алгоритмических языков состоит в том, что в них каждый из синтаксических знаков (или их совокупность) в отличие от живых языков несет свою собственную информацию о смысле, не зависимую от их сочетаний с другими знаками. В применении к адресному языку эта особенность позволяет изложить метод программирования в виде нового принципа работы программирующих программ.

Общий принцип работы известных до сих пор программирующих программ (см., например, [7]), в том числе приведенных далее ПП-2 и ПП-АК для машины «Киев», состоит в поиске в за-

писи алгоритма *представимой совокупности символов*, по которым можно сформировать одну или несколько машинных команд. При скобочной записи формул такой представимой совокупностью будет, например, *выполнимая операция* — знак двухместной операции, соединяющий два соседних с ним символа величин, или знак одноместной операции, за которым непосредственно следует величина. В отведенном рабочем массиве программирующая программа по найденному представимому сочетанию кодов записывает надлежащую команду или несколько команд (в зависимости от адресности машины и пр.), выбирая при необходимости рабочую ячейку для результата, и заменяет в исходной информации это сочетание символов номером рабочей ячейки. Далее в той или иной форме осуществляется сжатие исходной информации, поскольку количество ее элементов уменьшается при такой обработке, и процесс повторяется вплоть до исчерпания всей записи. Реализация тех или иных усовершенствований (например, экономия рабочих ячеек) не вызывает принципиальных затруднений.

Можно предложить следующий принцип поэлементного перевода адресных формул в машинный код:

1) для построения программы, соответствующей одной адресной формуле, отводится рабочее поле, достаточное по объему;

2) исходная информация просматривается поэлементно в естественном порядке и только один раз и в зависимости от просматриваемого в данный момент элемента информации производятся записи в рабочем поле.

Принцип однократного поэлементного просмотра информации был предложен Е. Л. Ющенко [22] для реализации алгоритма формальной проверки правильности бесскобочной и скобочной записей формул. Позднее М. М. Бушко-Жук предложил использовать этот принцип применительно к работе программирующих программ.

Рассмотрим алгоритм подобного рода для перевода адресных функций в код машины «Киев». Для простоты предположим, что в запись адресной функции могут входить: адреса произвольного ранга, за исключением адресов нулевого ранга; произвольные машинные двухместные операции (арифметические или логические); одноместные операции с входным адресом 0002 и выходным 0003, кодируемые адресами начальных команд подпрограмм, реализующих эти операции.

Примем строго скобочную запись формул, в которой порядок выполнения операций определяется только скобками (старшинство операций не учитывается), лишних скобок нет.

Описание адресной функции в строго скобочной записи дадим с помощью металингвистических формул [13]. Последовательности знаков, заключенных в скобки $< >$, представляют собой металингвистические переменные, значениями которых являются последовательности символов; знаки $::=$ и $|$ являются мета-

лингвистическими связками: знак $::=$ означает «равно по определению», знак | означает «или». Знак в формуле, не являющийся переменной или связкой, обозначает самого себя или подобный ему класс знаков. Соединение знаков или переменных означает соединение обозначаемых последовательностей.

Согласно принятому условию основными символами являются:

(— открывающая скобка;

) — закрывающая скобка;

O_1 — одноместные операции, к которым отнесем также штрих-операцию;

O_2 — двухместные операции;

a — адреса.

Таким образом,

$$< \text{основной символ} > ::= (\mid) \mid O_1 \mid O_2 \mid a.$$

Вначале введем понятие *компоненты*:

$$< \text{компонента} > ::= a \mid O_1 < \text{компонента} > \mid (< \text{компонента} > O_2 < \text{компонента} >).$$

Теперь строго скобочная запись адресной функции может быть определена следующим образом:

$$< \text{строго скобочная запись функции} > ::= a \mid O_1 < \text{компонента} > \mid < \text{компонента} > O_2 < \text{компонента} >.$$

Формула дает рекурсивное правило для образования адресных функций в строго скобочной записи.

Рабочий массив, отведенный для составления программы, заполняется снизу вверх. Компоненты команд устанавливаются по мере *просмотра закодированной информации* (в процессе работы некоторые команды могут сформироваться полностью, в то время как другие, быть может и те, которые будут выполняться раньше, еще вовсе не сформированы или сформированы частично). Экономия рабочих ячеек производится попутно.

Введем обозначения:

φ_0 — фиксатор начального элемента информации;

φ — фиксатор элементов входной информации; вначале $'\varphi = C^{-1}(\alpha)$;

C — операция следования в последовательности адресов элементов исходной информации, первый из которых обозначим через α (в одном машинном адресе может находиться несколько элементов);

ψ — фиксатор рабочего массива, отведенного для записи команд рабочей программы; адрес последней ячейки этого массива есть s;

$\varkappa$ — фиксатор массива рабочих ячеек рабочей программы (РП);

r — первая рабочая ячейка этого массива;

Σ, μ, λ — рабочие фиксаторы.

Фиксатор Σ используется при обработке скобок. С помощью

этого фиксатора в некотором массиве рабочих ячеек программирующей программы (ПП), содержащем k адресов ($\gamma+1$, $\gamma+2$, ..., $\gamma+k$), при обработке символов открывающих скобок фиксируются надлежащие адреса команд РП, к формированию которых совершается переход при обработке соответствующего (парного) символа закрывающей скобки. Размер массива k определяет допустимую максимальную глубину скобок в выражениях адресных функций.

Фиксатор μ используется для хранения информации о том, что в данный момент программируется одноместная операция — обращение к соответствующей подпрограмме (при этом в μ засылается единица).

Фиксатор λ используется для запоминания адреса команды РП, к программированию которого необходимо перейти после обработки знаков штрих-рангов выше первого или знака одноместной операции.

Условимся, кроме того, считать, что *пр f* означает признак элемента *f*. Среди признаков различаем:

ω — признак конца записи функции;
'адр — признак адреса первого ранга;
(— открывающая скобка;
) — закрывающая скобка;
'не-адр — признак наличия штрих-операции, не относящейся непосредственно к адресу;
O_1 — признак одноместной операции;
O_2 — признак двухместной операции;
раб — признак рабочей ячейки.

В алгоритме признаки элементов ω, 'адр, (,), *'не-адр*, O_1, O_2 используются в качестве меток.

Через $'\psi_0$, $'\psi_I$, $'\psi_{II}$, $'\psi_{III}$ обозначены соответственно содержимые 0-го, I, II, III адресов ячейки ψ. Коды машины «Киев», используемые в адресной программе, взяты в прямоугольные рамки.

Особо отметим роль адресной формулы безусловного перехода *пр* $^2\varphi$ (третья строка алгоритма).

Фиксатор φ обозревает элементы входной информации (адресного алгоритма), т. е. в каждый данный момент работы алгоритма $^2\varphi$ является обрабатываемым элементом. В зависимости от признака элемента по формуле

$$пр\,^2\varphi$$

совершается переход на строку алгоритма, помеченную соответствующей меткой.

Функция *пр* $^2\varphi$ может принимать одно из семи значений: ω, *'адр*, *'не-адр*, (,), O_1, O_2; таким образом, эта формула означает переход на одну из семи ветвей алгоритма, каждая из которых помечена соответствующей меткой.

Алгоритм поэлементного перевода адресных функций в коды машины «Киев»

Исходное адресное отображение

$$
\begin{array}{ll}
{}'\varphi_0 = \alpha & \\
{}'(\alpha+1) & \left.\begin{array}{l}\\\\\end{array}\right\} \text{поэлементно} \\
{}'(\alpha+2) & \left.\begin{array}{l}\\\\\end{array}\right\} \text{закодированная} \\
\vdots & \\
{}'(\alpha+N) & \left.\begin{array}{l}\\\\\end{array}\right\} \text{адресная функция}
\end{array}
$$

Результативное адресное отображение

$$
\left.\begin{array}{l}
\vdots \\
{}'(s-2) \\
{}'(s-1) \\
{}'s
\end{array}\right\} \quad \text{программа для вычисления значения адресной функции в кодах машины «Киев»}
$$

Адресный алгоритм

```
B ... 0 ⟹ μ;  0 ⟹ λ;  r ⟹ ϰ;  s ⟹ ψ;  γ + 1 ⟹ Σ
H ... C('φ) ⟹ φ
      np²φ
( ... 0 ⟹ μ
      P{'λ = 0}'ψ ⟹ 'Σ↓'λ ⟹ 'Σ;  0 ⟹ λ
      'Σ + 1 ⟹ Σ
      P{'('ψI) ≠ 0}(b
      'ϰ ⟹ 'ψI
      'ψ − 1 ⟹ ψ
(a ... 'ϰ ⟹ 'ψIII
      H
(b ... P{np'('ψI) = раб}'ϰ + 1 ⟹ ϰ
      'ϰ ⟹ 'ψII
      Ц{'ψ(−1)P{²ψ ≠ 0} ⟹ ψ}
L ...
        (a
) ... 'Σ − 1 ⟹ Σ
      0 ⟹ λ
      "Σ ⟹ ψ
      H
O2 ... P{'λ ≠ 0}'λ ⟹ ψ,  0 ⟹ λ
      ²φ ⟹ 'ψ0
      0 ⟹ μ
      H
```

ШТРИХ a ... $P\{'('\psi_I) = 0\}^2\varphi \Longrightarrow' \psi_I \downarrow^2\varphi \Longrightarrow' \psi_{II}$

Н

ШТРИХ ... $P\{'('\psi_I) \neq 0\}$ Шб

$P\{'\mu = 0\}$ Ша

$0 \Longrightarrow \mu$; $|\overline{\Phi}| \Longrightarrow' \psi_0$

Н

Ша ... $'x \Longrightarrow' \psi_I$

$P\{'\lambda = 0\}' \psi \Longrightarrow \lambda$

$'\psi - 1 \Longrightarrow \psi$

Шв

Шб ... $P\{np'('\psi_I) = раб\}' x + 1 \Longrightarrow x$

$'x \Longrightarrow' \psi_{II}$

$Ц\{'\psi(-1) P\{^2\psi \neq 0\} \Longrightarrow \psi\}$

Z ...

Шв ...

Φ			$'x$

$\Longrightarrow' \psi$

Н

O_1 ... $P\{'('\psi_I) \neq 0\}$ $O_1 b$

$|\overline{0003}| \Longrightarrow' \psi_I$

$P\{'\lambda = 0\}' \psi \Longrightarrow \lambda$

$O_1 a$ ... $'\psi - 1 \Longrightarrow \psi$

УПП	3026	$'\psi + 1$	$^2\varphi$

$\Longrightarrow' \psi$

$'\psi - 1 \Longrightarrow \psi$

+			0002

$\Longrightarrow' \psi$

$1 \Longrightarrow \mu$

Н

$O_1 b$ ... $P\{('('\psi_I) = 0003) \vee np'('\psi_I) = раб\}' x + 1 \Longrightarrow x$

$'x \Longrightarrow' \psi_{II}$

$Ц\{'\psi(-1) P\{^2\psi \neq 0\} \Longrightarrow \psi\}$

Y ...

+	0003		$'x$

$\Longrightarrow' \psi$

$O_1 a$

ω ... Я

Рассмотрим работу алгоритма на примере программирования адресной функции

$$(''a - \sin\cos'('a + 'b)) \times \sin'' c\omega,$$

где ω — признак конца.

Будем отмечать изменения адресного отображения на каждом шаге работы алгоритма.

Вначале $'\mu = 0$; $'\lambda = 0$; $'\varkappa = r$; $'\psi = s$; $'\Sigma = \gamma + 1$; $'(s - i) = 0$, где $i = 0, 1, 2, \ldots$

1. Обрабатывается код открывающей скобки; $пр^2\varphi = (:$

$'(\gamma + 1) = s$

$'\Sigma = \gamma + 2$

$'s =$

	r		

$'\psi = s - 1$

$'(s - 1) =$

			r

2. Обрабатывается код « ' »; $пр^2\varphi =$ ШТРИХ:

$'(s - 1) =$

	r		r

$'\lambda = s - 1$

$'\psi = s - 2$

$'(s - 2) =$

Φ			r

3. Обрабатывается код $'a$; $пр^2\varphi =$ ШТРИХ a:

$'(s - 2) =$

Φ	a		r

4. Обрабатывается код « — »; $пр^2\varphi = O_2$:

$'\psi = s - 1$

$'\lambda = 0$

$'(s - 1) =$

—	r		r

5. Обрабатывается код sin; $пр^2\varphi = O_1$:

$'\varkappa = r + 1$

$'(s - 1) =$

—	r	$r + 1$	r

$'\psi = s - 3$

$'(s-3) =$

+	0003		$r+1$

$'\psi = s - 4$

$'(s-4) =$

УПП	3026	$s-3$	sin

$'\psi = s - 5$

$'(s-5) =$

+			0002

$'\mu = 1$

6. Обрабатывается код cos; $np^2\varphi = O_1$:

$'(s-5) =$

+	0003		0002

$'\lambda = s - 5$

$'\psi = s - 6$

$'(s-6) =$

УПП	3026	$s-5'$	cos

$'\psi = s - 7$

$'(s-7) =$

+			0002

$'\mu = 1$

7. Обрабатывается код « ' »; $np^2\varphi =$ ШТРИХ:

$'\mu = 0$

$'(s-7) =$

Ф			0002

8. Обрабатывается код « (»; $np^2\varphi =$ (:

$'(\gamma + 2) = s - 5$

$'\lambda = 0$

$'\Sigma = \gamma + 3$

$'(s-7) =$

Ф	$r+1$		0002

$'\psi = s - 8$

$'(s-8) =$

			$r+1$

9. Обрабатывается код « $'a$ »; $пр^2\varphi =$ ШТРИХ a:

$$'(s-8) = \boxed{\ \ |\ a\ |\ \ |\ r+1\ }$$

10. Обрабатывается код «+»; $пр^2\varphi = O_2$

$$'(s-8) = \boxed{\ +\ |\ a\ |\ \ |\ r+1\ }$$

11. Обрабатывается код «$'b$»; $пр^2\varphi =$ ШТРИХ a:

$$'(s-8) = \boxed{\ +\ |\ a\ |\ b\ |\ r+1\ }$$

12. Обрабатывается код «) »; $пр^2\varphi =$):

$$'\Sigma = \gamma + 2$$
$$'\psi = s - 5$$

13. Обрабатывается код «) »; $пр^2\varphi =$):

$$'\Sigma = \gamma + 1$$
$$'\psi = s$$

14. Обрабатывается код « $\times$ »; $пр^2\varphi = O_2$:

$$'s = \boxed{\ \times\ |\ r\ |\ \ |\ \ }$$

15. Обрабатывается код sin; $пр^2\varphi = O_1$:

$$'\varkappa = r + 2$$

$$'s = \boxed{\ \times\ |\ r\ |\ r+2\ |\ \ }$$

$$'\psi = s - 9$$

$$'(s-9) = \boxed{\ +\ |\ 0003\ |\ \ |\ r+2\ }$$

$$'\psi = s - 10$$

$$'(s-10) = \boxed{\ УПП\ |\ 3026\ |\ s-9\ |\ \sin\ }$$

$$'\psi = s - 11$$

$$'(s-11) = \boxed{\ +\ |\ \ |\ \ |\ 0002\ }$$

$$'\mu = 1$$

16. Обрабатывается код «′»; $пр^2\varphi$ = ШТРИХ:

$'\mu = 0$

$'(s-11) =$

Ф			0002

17. Обрабатывается код «′*c*»; $пр^2\varphi$ = ШТРИХ *a*:

$'(s-11) =$

Ф	*c*		0002

Вся программа в кодах машины «Киев» выглядит так:

s — 11	*Ф*	*c*		0002
s — 10	*УПП*	3026	s — 9	sin
s — 9	+	0003		*r* + 2
s — 8	+	*a*	*b*	*r* + 1
s — 7	*Ф*	*r* + 1		0002
s — 6	*УПП*	3026	s — 5	cos
s — 5	+	0003		0002
s — 4	*УПП*	3026	s — 3	sin
s — 3	+	0003		*r* + 1
s — 2	*Ф*	*a*		*r*
s — 1	—	*r*	*r* + 1	*r*
s	×	*r*	*r* + 2	

ГЛАВА IV

БИБЛИОТЕКА ПОДПРОГРАММ МАШИНЫ «КИЕВ»

ОБЩИЙ ПРИНЦИП ПОСТРОЕНИЯ СТАНДАРТНЫХ МАССИВОВ И МЕТОД ПОДПРОГРАММ

Анализ адресных программ показывает, что наличие в ЦАМ алгоритмических операций обращения по адресу второго ранга упрощает алгоритмический язык и тем самым значительно расширяет возможности машины. Необходимо отметить, что использование таких операций упрощает программы, а для машин, имеющих постоянную память, дает возможность размещать в ней произвольные программы и облегчает задачу создания библиотеки стандартных подпрограмм и использования их в процессе решения задач.

Известно, что при наличии богатой библиотеки можно строить сложные программы почти целиком из библиотечных подпрограмм с помощью так называемых компилирующих или интерпретирующих систем [12].

Создание библиотеки подпрограмм для машины «Киев» предусматривает широкий набор стандартных подпрограмм, например для вычисления скалярных и векторных функций от скалярных, векторных и матричных аргументов. Наличие библиотеки упрощает работу по автоматизации программирования, контролю, вводу и отладке отдельных частей программы и тем самым снижает вероятность ошибок программирования и дает значительный выигрыш во времени.

Библиотека подпрограмм машины «Киев» включает набор программ, размещаемых в пассивном запоминающем устройстве (ПЗУ), в оперативной памяти и на магнитных барабанах. Ее отличительной особенностью является наличие сменно-спаянных блоков, которые в зависимости от необходимости могут включаться в пассивную память машины.

Одна часть стандартных подпрограмм, как например, $\sin x$, $\ln x$, $\operatorname{arc\,tg} x$, e^x и т. д., а также некоторые константы впаяны в постоянные блоки ПЗУ. В сменно-спаянную память входят универсальные программирующие программы; подпрограммы для задач линейной алгебры; подпрограммы, реализующие режим плавающей запятой, и др. Кроме того, сюда входят программы для вычисления одной и той же функции с различной точностью вычисления, а также различным объемом памяти и временем

Таблица 2

Постоянно-спаянные константы

Восьмеричный номер ячейки	Содержимое в 8-й системе	Примечания
3010	00 0001 0000 0000	2^{-16}
3011	00 0000 0001 0000	2^{-28}
3012	00 0000 0000 0001	2^{-40}
3013	00 0001 0001 0001	
3014	00 0001 0001 0000	
3015	00 0000 0001 0001	
3016	00 0001 0000 0001	
3017	00 0002 0000 0000	2^{-15}
3020	00 0000 0002 0000	2^{-27}
3021	00 0000 0000 0002	2^{-39}
3022	00 7777 0000 0000	
3023	00 0000 7777 0000	
3024	00 0000 0000 7777	
3025	37 0000 0000 0000	$-\frac{15}{16}$
3026	20 0000 0000 0000	-0
3027	37 0000 7777 7777	
3030	00 7777 7777 7777	
3031	37 7777 0000 7777	
3032	37 7777 7777 0000	
3033	00 7777 0000 7777	
3034	00 0000 7777 7777	
3035	17 7777 7777 7777	Машинная единица $1-2^{-40}$
3036	37 7777 7777 7777	$-(1-2^{-40})$
3037	04 6420 2324 1220	$\lg 2$
3040	00 0020 0000 0000	2^{-12}
3041	00 0000 0020 0000	2^{-24}
3042	10 0000 0000 0000	2^{-1}
3043	04 0000 0000 0000	2^{-2}
3044	02 0000 0000 0000	2^{-3}
3045	01 0000 0000 0000	2^{-4}
3046	00 2000 0000 0000	2^{-6}
3047	00 0000 0000 0020	2^{-36}
3050	12 0000 0000 0000	$\frac{10}{16}$
3051	13 0562 0577 3722	$\ln 2$
3052	01 4631 4631 4632	0,1
3053	00 1217 2702 4366	0,01
3054	00 0101 4223 3514	0,001
3055	03 1463 1463 1463	$0,2=\frac{1}{5}$
3056	04 6314 6314 6315	0,3
3057	14 6314 6314 6315	$0,8=\frac{4}{5}$
3060	00 0203 0446 7230	$0,002=\frac{1}{500}$
3061	02 4365 6050 7534	$0,16=\frac{4}{25}$
3062	05 0753 4121 7270	0,32

Продолжение табл. 2

Восьмеричный номер ячейки	Содержимое в 8-й системе	Примечания
3063	05 0574 6033 3447	$\frac{1}{\pi}$
3064	11 0156 5650 1025	$\frac{1}{\sqrt{\pi}}$
3065	05 7055 2615 4264	$\frac{1}{e}$
3066	14 4417 6652 1042	$\frac{\pi}{4}$
3067	12 1371 4066 7116	$\frac{2}{\pi}$
3070	13 2404 7463 1772	$\frac{\sqrt{2}}{2}$
3071	11 1715 1642 6202	$\frac{1}{\sqrt{3}}$
3072	16 3765 6134 5604	$\frac{e}{3}$
3073	06 3041 0514 7521	$\frac{1}{\sqrt{2\pi}}$
3074	12 6770 2505 4243	$\frac{e}{4}$
3075	05 2525 2525 2525	$\frac{1}{3}$
3076	01 4760 1366 1043	$\frac{1}{\pi^2}$
3077	06 7455 7305 2237	$M = \log_{10} e$

работы. Последнее дает возможность оптимально использовать машину для решения конкретных задач.

Использование стандартных подпрограмм сопряжено с рядом специфических вопросов.

В зависимости от того, насколько велико разнообразие задач, решаемых с помощью стандартной подпрограммы, стандартной подпрограмме, кроме непосредственно исходных данных, задается еще то или иное число различных вспомогательных параметров (размерности векторов, матриц, таблиц, порядок интегрируемой системы дифференциальных уравнений и т. д.). Обычно вся исходная информация для стандартной подпрограммы размещается на фиксированных местах памяти, что зачастую сопряжено со значительными затратами оперативной памяти (в случае многоместных программ) и времени на пересылку данных из рабочих в стандартные ячейки подпрограмм и, наоборот, из стандартных ячеек в рабочие.

Кроме того, в число различных вспомогательных параметров может включаться информация о размещении исходных данных

в памяти машины, что связывается, как правило, с перестройкой программ для заданных параметров.

Использование операций по адресу второго ранга в машине «Киев» позволяет освободиться от перестройки стандартных программ для новых параметров и, в частности, дает возможность помещать любую стандартную программу в постоянной памяти, а кроме того, позволяет свести к минимуму требования относительно стандартности размещения исходной информации в оперативной памяти машины. Для большинства стандартных подпрограмм (а принципиально допустимо для всех) достаточно одного-двух фиксированных адресов, в которых хранится информация о размещении исходных числовых массивов. В простейшем случае при вычислении одно- и двухместных функций в этих адресах (обозначим их φ_0, φ_1) содержатся их аргументы или значения функций.

Для задания более сложных форм информации уславливаемся о следующем:

А. Пусть исходными данными для подпрограммы служит вектор; тогда компоненты его $a_1, \ldots, a_n$ размещаются в последовательность адресов

$$C\alpha,\ (C^2)\,\alpha,\ \ldots,\ (C^n)\,\alpha, \tag{17}$$

где C — некоторая операция следования в множестве адресов (машинному адресу при этом может соответствовать несколько адресов или наоборот). Для наиболее часто употребляемой операции следования, определяемой соотношением $C\alpha = \alpha + 1$, вместо (17) имеем последовательность

$$\alpha + 1,\ \alpha + 2,\ \ldots,\ \alpha + n. \tag{18}$$

Кроме этого, по адресу α, называемому ведущим адресом последовательности, всегда помещается информация о векторе в виде его размерности n (для машины «Киев» величины такого рода удобно помещать во вторые адреса ячеек).

Последовательность адресов

$$\alpha,\ C\alpha,\ \ldots,\ (C'^{\alpha})\,\alpha$$

будем называть α-последовательностью с операцией следования C (типа вектора).

Стандартной программе достаточно задать на фиксированном месте φ_0 адрес, который будем называть *ведущим адресом* последовательности (17):

$$'\varphi_0 = \alpha \qquad ('' \varphi_0 = '\alpha = n).$$

Таким образом, информация о векторе задается по содержимому одного адреса φ_0.

Б. Информация о квадратной матрице; компоненты матрицы a_{ij} размещаются в последовательности адресов

$$C\alpha,\ (C^2)\,\alpha,\ \ldots,\ (C^{(i-1)\,n+j})\,\alpha$$

по строкам или столбцам в зависимости от требований задачи; информация о матрице задается в виде одного числа (содержимого ячейки α) — размерности матрицы n.

Последовательность

$$\alpha,\ C\alpha,\ \ldots,\ (C^{('\alpha)^2})\,\alpha$$

будем называть также α-последовательностью (типа квадратной матрицы).

Стандартной программе задается лишь информация в виде содержимого адреса φ_0, равного адресу ведущего элемента последовательности:

$$'\varphi_0 = \alpha;$$
$$^2\varphi_0 = n.$$

В. Информация о прямоугольной матрице; по адресам α и $C\alpha$ задаются размерности матрицы m и n, а по адресам $(C^2)\,\alpha$, $(C^3)\,\alpha$, ..., $(C^{mn+1})\,\alpha$ размещаются элементы матрицы по строкам или столбцам в зависимости от требований стандартной программы.

Последовательность такого рода будем называть α-последовательностью (типа прямоугольной матрицы). Как и ранее, информация стандартной программе о последовательности задается в виде содержимого одного адреса — адреса ведущего элемента последовательности:

$$'\varphi_0 = \alpha;$$
$$^2\varphi_0 = m;$$
$$'('\varphi_0 + 1) = n.$$

Г. Исходной информацией для стандартной программы служит квадратная матрица, элементы которой являются векторами различной размерности n_{ij}. Компоненты векторов — элементов матрицы размещаются в последовательности типа A. Пусть ведущими адресами этих последовательностей будут

$$\alpha_{ij} \qquad ('\alpha_{ij} = n_{ij}).$$

Величины α_{ij} размещаются в свою очередь в последовательность типа Б

$$\alpha,\ C\alpha,\ \ldots,\ (C^{('\alpha)^2})\,\alpha.$$

Д. Исходной информацией для подпрограммы служит тре-

угольная (симметричная) матрица n-го порядка. Информация задается по строкам (или столбцам) в виде последовательности

$$\alpha,\ C\alpha,\ \ldots,\ (C^{\frac{('\alpha)^2+'\alpha}{2}})\alpha,$$

где по адресу α содержится порядок матрицы n.

Как и в предыдущих случаях, информация о массиве задается в виде содержимого одного адреса φ_0, по которому помещается адрес ведущего элемента последовательностей. Из изложенного ясен общий принцип построения стандартных массивов информации и метод свертывания информации о них.

Каждая стандартная подпрограмма, как правило, заканчивается командой *ПРВ*, функциональное назначение которой состоит в передаче управления команде, адрес которой содержится в регистре возврата. Содержимое регистра возврата соответственно подготавливается при обращении к подпрограмме. Однако такой способ перехода на подпрограммы на машине «Киев» ввиду наличия одного регистра возврата возможен только для программ, не содержащих обращения к подпрограммам. В остальных случаях возврат от подпрограмм к программам реализуется с помощью заранее подготавливаемого приказа безусловного перехода. В зависимости от глубины перехода на подпрограммы принято пользоваться фиксированными адресами.

ПОДПРОГРАММЫ ПОСТОЯННО-СПАЯННОЙ ПАМЯТИ (ПСП)

Для всех программ одноместных элементарных функций входным адресом, т. е. ячейкой, в которую до обращения к подпрограмме должен быть помещен аргумент x, является 0002; выходным, т. е. ячейкой, в которой подпрограмма выдает значение искомой функции, является 0003. Исключение составляет программа $\ln x$, которая не является одноместной; для вычисления $\ln x$ аргумент x предварительно засылается в ячейку 0002, а результат выдается в двух ячейках: 0003 — значение функции, 0004 — масштабный множитель.

Все программы ПСП заканчиваются приказом перехода по регистру возврата (*ПРВ*), что должно учитываться при программировании.

Часто употребляемые константы включены в ПСП. Их список приведен в табл. 2. Список подпрограмм ПСП приведен в табл. 3.

Программа 1

[2 → 10; перевод из двоичной системы счисления в десятичную]

```
2 → 10...3100 01 0000 0000 0003      3105 01 0004 0003 0003
         1 26 0012 0000 0000            6 12 0004 3047 0004
         2 11 0002 3050 0002            7 02 0002 0004 0002
         3 10 3047 0002 0004      3110 12 0002 3045 0002
         4 12 0003 3045 0003            1 27 4001 3102 3146
```

Таблица 3

Подпрограммы сменно-спаянной памяти

Название программы	Область изменения аргумента	Аргумент x, результат y	Начальная команда	Рабочие ячейки	Количество команд и констант	Точность десятичных знаков
Перевод 2→10	(—1,1)	'0002 = x '0003 = y	3100	0004	10	10
$\frac{1}{2}\sin x$	(—1,1)	'0002 = x '0003 = y	3152	0004	9	7—8
$\frac{1}{2}\cos x$	(—1,1)	'0002 = x '0003 = y	3147	0004	12	7—8
$\frac{1}{2}\sin x$	(—2π; 2π)	'0002 = $\frac{x}{2\pi}$ '0003 = y	3264	0004	20	6—7
$\frac{1}{2}\cos x$	(—2π; 2π)	'0002 = $\frac{x}{2\pi}$ '0003 = y	3217	0004—0005	19	6—8
$\frac{1}{n}\ln x$	(0,1)	'0002 = x '0003 = y '0004 = $\frac{1}{n}$	3116	0004—0006	29	6—8
$\sqrt{a}$	(2^{-22},1)	'0002 = a '0003 = y	3163	0004	6	7—8
$\frac{1}{4}e^x$	(—1,1)	'0002 = x '0003 = y	3202	0004—0006	13	8—9
$\frac{1}{\pi}\arcsin x$	(—1,1)	'0002 = x '0003 = y	3242	0004—0006	17	7—8
$\frac{1}{\pi}\arccos x$	(—1,1)	'0002 = x '0003 = y	3244	0004—0006	16	7—8

Аргумент '0002 (двоичное число); результат '0003 (десятичное число); начальная команда 3100.

Для вычисления $\frac{1}{2}\sin x\left(\frac{1}{2}\cos x\right)$ при $-\frac{\pi}{2} \leqslant x \leqslant \frac{\pi}{2}$ используется полином наилучшего приближения

$$\frac{1}{2}\sin x = \frac{1}{2}\sin\frac{\pi}{2}y = ((((C_9y^2 + C_7)y^2 + C_5)y^2 + C_3)y^2 + C_1)y,$$

где $C_9 = 0{,}000\,079\,742\,095$; $C_7 = -0{,}000\,233\,688\,278$;
$C_5 = 0{,}03\,984\,483\,964$; $C_3 = -0{,}32\,298\,185\,553$;
$C_1 = 0{,}785\,398\,159\,235$.

С помощью этого же полинома вычисляется $\frac{1}{2}\cos x$ с предварительным приведением аргумента по формуле

$$\bar{x} = \frac{\pi}{2} - |x| \text{ или } \bar{y} = 1 - |y|.$$

Программы 2 и 3

[вычисление $y = \frac{1}{2}\sin x$ и $y = \frac{1}{2}\cos x$ $(-1 < x < 1)$]

```
      3172 00 0004 7553 6722    3175 25 1256 7405 5264
         3 20 0231 1146 1443       6 14 4417 6651 0101
         4 00 5063 2127 5453       7 20 0000 0000 0001
cos...3147 11 0002 3067 0002    3155 01 3172 0000 0003
      3150 06 3035 0002 0002       6 11 0003 0004 0003
         1 05 0000 0000 3153       7 01 0003 7173 0003
sin...   2 11 0002 3067 0002    3160 27 4001 3156 3161
         3 11 0002 0002 0004       1 11 0003 0002 0003
         4 26 0004 0000 0000       2 32 0000 0000 0000
```

Аргумент '0002 $= x$; результат '0003 $= \frac{1}{2}\sin x$ (соответственно $\frac{1}{2}\cos x$); начальная команда 3152 для $\frac{1}{2}\sin x$ и 3147 для $\frac{1}{2}\cos x$.

Программа 4

[вычисление $\frac{1}{2}\sin x$ $(-2\pi < x < 2\pi)$]

```
sin...3264 04 0000 0002 3266    3272 02 0000 0002 0002
         5 01 3035 0002 0002       3 04 0002 3043 3275
         6 04 0002 3042 3273       4 02 3042 0002 0002
         7 02 3035 0002 0002       5 12 0002 3241 0002
      3270 04 0002 3043 3272       6 04 0000 0000 3153
         1 02 3042 0002 0002       7 22 0003 0003 3146
```

Аргумент '0002 $= \frac{x}{2\pi}$; результат '0003 $= \frac{1}{2}\sin x$; начальная команда 3264.

Программа 5

[вычисление $\frac{1}{2}\cos x$ $(-2\pi < x < 2\pi)$]

```
cos...3217 05 0002 3042 3221    3231 11 0003 0004 0003
      3220 06 3035 0002 0002       2 01 0003 7173 0003
         1 05 0002 3043 3237       3 27 4001 3231 3234
         2 06 3042 0002 0002       4 11 0003 0002 0003
         3 01 3026 0000 0005       5 14 0005 0003 0003
         4 12 0002 3241 0002       6 32 0000 0000 0000
         5 06 3035 0002 0002       7 01 0000 0000 0005
         6 11 0002 0002 0004    3240 04 0000 0000 3224
         7 26 0004 0000 0000       1 04 0000 0000 0001
      3230 01 3172 0000 0003
```

Аргумент $'0002 = \frac{x}{2\pi}$; результат $'0003 = \frac{1}{2}\sin x$; начальная команда 3217.

Вычисление $\ln x$ также осуществляется по полиному наилучшего приближения следующим образом. Функция $\ln x$ вычисляется по формуле

$$\ln x = M \log_2 x,$$

где $M = 0{,}6\,931\,471\,806 = '3051$ — модуль перехода от двоичных логарифмов к натуральным;

$$\log_2 x = 4\,(C_1\tau + C_2\tau^3 + C_3\tau^5 + C_4\tau^7);$$

$$\tau = \frac{x-1}{x+1};$$

$$C_1 = 0{,}721\,347\,518\,185;$$
$$C_2 = 0{,}240\,450\,190\,571;$$
$$C_3 = 0{,}144\,146\,085\,514;$$
$$C_4 = 0{,}108\,564\,937\,823.$$

Этот полином обеспечивает вычисление $\log_2 x$ на отрезке $\left[\frac{1}{\sqrt{2}}, \sqrt{2}\right]$ с точностью до $\varepsilon = 0{,}000\,000\,000\,005$.

Для $x = < \frac{1}{\sqrt{2}}$ подбирается такое n, чтобы

$$\frac{1}{\sqrt{2}} < (\sqrt{2})^n x < 1,$$

и используется соотношение

$$\ln x = M \log_2 x = M\left[\log_2 (\sqrt{2})^n x - \frac{n}{2}\right].$$

Так как при $\frac{1}{(\sqrt{2})^{n+1}} < x < \frac{1}{(\sqrt{2})^n}$ имеем $n < 2\,|\log_2 x| < n+1$, а $|\ln x| < |\log_2 x|$, то для x на этом отрезке вводится масштабный множитель $\frac{1}{n+2}$. Счет ведется по схеме Горнера

$$4\,(((C_4\tau^2 + C_3)\,\tau^2 + C_2)\,\tau^2 + C_1)\,\tau.$$

Адреса 3112—3115 содержат соответственно коэффициенты C_4, C_3, C_2, C_1.

Программа 6

(вычисление $\ln x$)

```
      3112 01 5712 7226 4561
         3 02 2346 6040 1441
         4 03 6616 1114 4322
         5 13 4252 1661 7650
ln ...   6 01 3020 0000 0005
         7 05 3070 0002 3123
      3120 12 0002 3070 0002
         1 01 0005 3011 0005
         2 05 0000 0000 3117
         3 12 3011 0005 0004
         4 02 0005 3020 0006
         5 12 0006 0005 0005
         6 11 0002 3042 0002
         7 02 0002 3042 0006
      3130 01 0002 3042 0002
      3131 12 0006 0002 0002
         2 11 0002 0002 0006
         3 12 0002 3043 0002
         4 26 0003 0000 0000
         5 01 3112 0000 0003
         6 11 0003 0006 0003
         7 01 0003 7113 0003
      3140 27 4001 3136 3141
         1 11 0003 0002 0003
         2 11 0003 0004 0003
         3 11 0005 3042 0005
         4 02 0003 0005 0003
         5 11 0003 3051 0003
         6 32 0000 0000 0000
```

Аргумент '0002; функция '0003; масштабный множитель '0004; начальная команда 3116.

Вычисление $y = \sqrt{a}$ для $2^{-22} < a < 1$ ведется согласно схеме:

$$y_{n+1} = y_n + \Delta y_n;$$

$$\Delta y_n = \frac{1}{2}\left(\frac{a}{y_n} - y_n\right);$$

$$y_0 = 1.$$

Программа 7

(вычисление $\sqrt{a}$)

```
√ ... 3163 01 0000 3035 0003
         4 12 0002 0003 0004
         5 02 0004 0003 0004
         6 11 0004 3042 0004
      3167 01 0003 0004 0003
      3170 05 3047 0004 3164
         1 32 0000 0000 0000
```

Аргумент '0002 $= a$; результат '0003 $= \sqrt{a}$; начальная команда 3163.

Для вычисления $\frac{1}{4}e^{-x}$ при $-1 < x < 1$ используется цепная дробь

$$\frac{1}{4}e^x = \frac{1}{4} + \cfrac{\cfrac{x}{4\cdot 16}}{\cfrac{1}{16} - \cfrac{x}{2\cdot 16} + \cfrac{\cfrac{x^2}{4\cdot 16^2}}{\cfrac{3}{16} + \cfrac{\cfrac{x^2}{4\cdot 16^2}}{\cfrac{5}{16} + \cfrac{\cfrac{x^2}{4\cdot 16^2}}{\cfrac{7}{16} + \cfrac{\cfrac{x^2}{4\cdot 16^2}}{\cfrac{9}{16} + \cfrac{\cfrac{x^2}{4\cdot 16^2}}{\cfrac{11}{16} + \cfrac{\cfrac{x^2}{4\cdot 16^2}}{\cfrac{13}{16}}}}}}}}$$

Программа 8

$\left(\text{вычисление } \frac{1}{4}e^x \text{ для } -1 < x < 1\right)$

```
e^x ...3202 06 3025 3044 0004      3211 05 3044 0004 3206
       3 01 0000 0004 0005            2 02 0005 0002 0005
       4 11 0002 3201 0002            3 11 0002 3042 0002
       5 11 0002 0002 0006            4 12 0002 0005 0002
       6 12 0006 0005 0005            5 01 0002 3043 0003
       7 02 0004 3044 0004            6 32 0000 0000 0000
    3210 01 0004 0005 0005
```

Аргумент '0002 $= x$; результат '0003 $= \frac{1}{4}e^x$; начальная команда 3202.

Для вычисления $\frac{1}{\pi}\arccos x$ и $\frac{1}{2}\arcsin x$ $(-1 \leqslant x \leqslant 1)$ используется метод «цифра за цифрой». Величина $\varphi = \arccos x$ ищется по формуле

$$\varphi = \varphi_0 = \frac{\pi}{2} z_0,$$

где $z_0 = \alpha_0, \alpha_1, \ldots, \alpha_i \ldots$ $(0 \leqslant z_0 \leqslant 2$; α_i — двоичные цифры).

Если $x_0 = x > 0$, то $\cos\varphi > 0$ и φ — угол в I четверти, т. е. $0 \leqslant \varphi \leqslant \frac{\pi}{2}$; значит, $z < 1$, что видно из соотношения $\varphi = \frac{\pi}{2} z_0$ и $\alpha_0 = 0$.

Если $x < 0$, то $\cos\varphi < 0$ и φ — угол во II четверти, т. е. $\frac{\pi}{2} \leqslant x \leqslant \pi$ и $z_0 > 1$. Значит, $\alpha_0 = 1$.

Дополнительно вычисляется x_{i+1} для определения того, в какой четверти находится угол φ. Если $x_i > 0$, то $\alpha_i = 0$ и $x_{i+1} = 2x_i^2 - 1$. Если $x_i < 0$, то $\alpha_i = 1$ и $x_{i+1} = 1 - 2x_i^2$.

$\frac{1}{\pi}\arcsin x$ вычисляется по формуле

$$\arcsin x = \frac{\pi}{2} - \arccos x.$$

Программы 9 и 10

[вычисление $\frac{1}{\pi}\arccos x$ и $\frac{1}{\pi}\arcsin x\,(-1 \leqslant x \leqslant 1)$]

arcsin ...	3242	02	0000	0000	0004	3253	14	0003	0006	0003
	3	05	0000	0000	3245	4	02	3042	0005	0002
arccos ...	4	01	0000	0000	0004	5	12	0002	3263	0002
	5	01	0000	0000	0003	6	10	3042	0006	0006
	6	01	3042	0000	0006	7	05	3012	0006	3247
	7	11	0002	0002	0005	3260	31	0004	3262	3261
	3250	31	0002	3251	3253	1	02	3042	0003	0003
	1	02	0005	3042	0002	2	32	0000	0000	0000
	2	05	0000	0000	3255	3	10	0000	0000	0001

Аргумент $'0002 = x$; результат $'0003 = \frac{1}{\pi}\arccos x$ (соответственно $\frac{1}{\pi}\arcsin x$); начальная команда для $\frac{1}{\pi}\arcsin x$ 3242 и для $\frac{1}{\pi}\arccos x$ 3244.

ПОДПРОГРАММЫ СМЕННО-СПАЯННОЙ ПАМЯТИ (ССП)

Сменно-спаянная память реализована в виде блоков, каждый из которых имеет один из номеров от 1 до 5 и содержит по 100 восьмеричных (64 десятичных) кодов.

Программы на блоках группируются так, чтобы по мере надобности могли включаться такие комплекты подпрограмм, вероятность одновременного включения которых для решения задач наибольшая. Так, например, программы линейной алгебры размещены на блоках с разными номерами и поэтому могут быть включены одновременно. После включения в набор операций машины «Киев» операции относительного перехода блоки спаянной памяти можно будет включать в произвольном порядке (фиксированные адреса должны при этом оставаться только за константами).

В качестве примера приведем описание двух блоков сменно-спаянной памяти.

Блок № 1. Элементарные функции

Один из комплектов блока № 1 (адреса 3300—3377) ССП содержит программы:

1. Вычисление квадратного корня (команды 3300—3317) для значений аргумента в интервале $0 \leqslant a < 1$. Начальная команда 3300. Как обычно, по адресу 0002 помещается аргумент, по адресу 0003 выдается результат.

2. Скалярное произведение двух векторов (команды 3200—3333). Начальная команда 3320. Адреса 0002 и 0004 — фиксаторы векторов; по адресу 0003 выдается результат. Оба вектора задаются в виде α-последовательностей (в этой главе во всех случаях речь идет о последовательностях с операцией следования $+1$).

3. Умножение квадратной матрицы на вектор (команды 3334—3357). Начальный адрес 3334. 0002 — фиксатор α-последовательности, по которой задается матрица по строкам; 0004 — фиксатор последовательности, по которой задан вектор; 0003 — фиксатор вектора-результата.

4. Вычисление $\frac{1}{4}a^x$ (команды 3360—3374). Начальная команда 3360. Поскольку эта программа использует программу ln, для нее принято $'0007 = a$; $'0010 = x$; $'0003 = \frac{1}{4}a^x$.

5. Последние три адреса использованы для помещения констант: $3375 = 2^{-14}$; $3376 = 0{,}4$; 3377 = константа для сдвига на один адрес $(14 \cdot 2^{-40})$.

Сменно-спаянный блок № 1

$\sqrt{\ }$ ...	3300	01	0000	0000	0003		2	01	0004	3011	0006
	1	05	0002	0000	3317		3	34	0002	0000	0007
	2	01	3035	0000	0003		4	34	0006	0000	0010
	3	16	0002	3035	3317		5	34	0005	0000	0000
	4	01	3043	0000	0005		6	11	4000	0010	0011
	5	05	3043	0002	3311		7	01	0003	0011	0003
	6	12	0002	3043	0002		3330	01	0005	3011	0005
	7	11	0005	3042	0005		1	01	0006	3011	0006
	3310	05	0000	0000	3305		2	02	0007	3011	0007
	1	12	0002	0003	0004		3	31	0007	3324	3146
	2	02	0004	0003	0004		4	34	0002	0000	0006
	3	11	0004	3042	0004		5	01	0002	4000	0007
	4	01	0003	0004	0003		6	01	0006	0003	0005
	5	05	3021	0004	3311		7	01	0002	3011	0010
	6	11	0003	0005	0003		3340	01	0003	3011	0012
	7	32	0000	0000	0000		1	01	0004	3011	0011
(x, y) ...	3320	01	0000	0000	0003		2	01	0000	0000	0015
	1	01	0002	3011	0005		3	34	0010	0000	0013

```
        4 34 0011 0000 0000
        5 11 0013 4000 0014
        6 01 0014 0015 0015
        7 01 0010 3011 0010
     3350 01 0011 3011 0011
        1 05 0010 0007 3343
        2 34 0012 0000 0000
        3 01 0015 0000 4000
        4 01 0012 3011 0012
        5 01 0007 0006 0007
        6 05 0012 0005 3341
        7 32 0000 0000 0000
```

$\frac{1}{4}a^x$ ... 3360 01 0000 3035 0011

```
        1 01 0000 0007 0002
        2 30 3026 3363 3116
        3 11 0003 0010 0012
        4 05 0012 0004 3370
        5 01 0012 0004 0012
        6 11 0011 3065 0011
        7 05 0000 0000 3364
     3370 12 0012 0004 0012
        1 01 0000 0012 0002
        2 30 3026 3373 3202
        3 11 0003 0011 0003
        4 05 0000 0000 0001
        5 00 0004 0000 0000
        6 06 3146 3146 3146
        7 00 0000 0000 0014
```

Блок № 4. Метод Рунге-Кутта

Один из комплектов блока № 4 сменно-спаянной памяти (адреса 3600—3677) предназначен для размещения в нем программы решения систем обыкновенных дифференциальных уравнений методом Рунге-Кутта с постоянным шагом интегрирования.

Метод Рунге-Кутта в применении к системе n обыкновенных дифференциальных уравнений

$$y_i' = f_i(x,\ y_1,\ \ldots,\ y_n)\ (i = 1,2,\ \ldots,\ n) \tag{19}$$

состоит в вычислении последовательных значений функций y_{ki}, где k — номер точки, а i — номер уравнения (функции) по формулам:

$$\left.\begin{aligned}
y_{k+1,\,i} &= y_{k,\,i} + \frac{hK_{1i}}{6} + \frac{hK_{2i}}{3} + \frac{hK_{3i}}{3} + \frac{kK_{4i}}{6};\\
K_{1i} &= f_i(x,\ y_{k1},\ \ldots,\ y_{kn});\\
K_{2i} &= f_i\left(x + \frac{h}{2},\ y_{k1} + \frac{h}{2}K_{11},\ \ldots,\ y_{kn} + \frac{h}{2}K_{1n}\right);\\
K_{3i} &= f_i\left(x + \frac{h}{2},\ y_{k1} + \frac{h}{2}K_{21},\ \ldots,\ y_{kn} + \frac{h}{2}K_{2n}\right);\\
K_{4i} &= f_i(x + h,\ y_{k1} + hK_{31},\ \ldots,\ y_{kn} + hK_{3n}),
\end{aligned}\right\} \tag{20}$$

где h — шаг интегрирования.

На каждом шаге, как это следует из формул (20), четыре раза вычисляются значения правых частей.

Исходная информация — число уравнений n и начальные значения функций y_i — задается в виде α-последовательности:

$$\begin{aligned}
'\alpha &= n;\\
'(\alpha + 1) &= y_{01};\\
'(\alpha + 2) &= y_{02};\\
&\ldots\ldots\\
'(\alpha + n) &= y_{0n},
\end{aligned}$$

ведущий адрес которой помещается во II адресе наборного кода (НК)

'3005			α	

Кроме этого, в наборных кодах задаются:

'3000 $= 2h$ — удвоенный шаг интегрирования;

'3001 $= x_0$ — начальное значение независимого переменного;

'3002 $= x_{max}$ — конечное значение независимого переменного;

'3004			m	

Здесь число m означает выдачу (на печать) результатов вычислений в каждой m-й точке интегрирования.

В качестве рабочих используются адреса 0004—0043, а также $4n$ адресов, представляющих собой продолжение α-последовательности, первые n из которых используются для хранения текущих значений правых частей уравнений, следующие n—для задания аргументов для них и последние n — для последовательного вычисления величин $2hK_{ij}$, т. е.

$$\left.\begin{aligned} &'(\alpha + 3n + 1) = 2h \cdot K_{j1};\\ &'(\alpha + 3n + 2) = 2h \cdot K_{j2};\\ &\ldots\ldots\ldots\ldots\\ &'(\alpha + 4n) = 2h \cdot K_{jn}. \end{aligned}\right\} \quad (j = 1, 2, 3, 4)$$

Начальный адрес программы 3600. Текущее значение независимого переменного для вычисления соответствующих значений правых частей задается по адресу 0031. Возврат на программу Рунге-Кутта после вычисления правых частей f_i осуществляется на команду 3634.

Начальный адрес подпрограммы вычисления правых частей задается как содержимое наборного кода

3003	04			N

'3677 равно контрольной сумме данного блока ССП. По адресу 3200 (команда 3615) передается управление на подпрограмму группового перевода и печати.

'3000 = $2h$
'3001 = x_0
'3002 = x_{max},

'3003	04			N
'3004			m	
'3005			α	

```
РУНГЕ ... 3600 01 0000 3043 0040
             1 11 0040 3075 0036
             2 11 3042 3075 0037
             3 34 3005 0000 0043
             4 01 3005 4000 0033
             5 01 0033 4000 0016
             6 01 0016 4000 0035
             7 01 0036 0000 0017
          3610 01 0036 0000 0020
             1 01 3001 0000 0031
             2 01 3004 0000 0032
             3 01 0000 0000 0022
             4 01 0000 0000 0021
             5 04 3002 0031 3200
             6 01 0031 0000 0034
             7 01 3005 3011 0024
          3620 01 0033 3011 0025
             1 01 0016 3011 0026
             2 34 0024 0000 0023
             3 34 0025 0000 0000
             4 01 0023 0000 4000
             5 34 0026 0000 0000
             6 01 0000 0023 4000
             7 01 0024 3011 0024
          3630 01 0025 3011 0025
             1 01 0026 3011 0026
             2 04 0024 0033 3622
             3 05 0000 0000 3003
          3634 01 0035 3011 0027
             5 01 3005 3011 0024
             6 02 0036 0020 0020
             7 01 0020 0017 0017
          3640 04 0017 0037 3642
             1 02 0017 0037 0017
             2 34 0027 0000 0000
             3 11 4000 0017 0023
             4 34 0024 0000 0000
             5 01 4000 0023 4000
             6 01 0027 3011 0027
             7 01 0024 3011 0024
          3650 04 0024 0033 3642
             1 16 0022 3042 3673
             2 02 0040 0021 0021
             3 01 0021 0022 0022
             4 11 3000 0022 0041
             5 01 0041 0034 0031
             6 01 0035 3020 0042
             7 02 0026 3011 0026
          3660 02 0025 3011 0025
             1 02 0027 3011 0027
             2 34 0027 0000 0000
             3 11 0022 4000 0023
             4 34 0026 0000 0030
             5 34 0025 0000 0000
             6 01 0023 0030 4000
             7 04 0042 0027 3657
          3670 01 0026 0043 0026
             1 01 0025 0043 0025
             2 04 0000 0000 3003
             3 02 0032 3011 0032
             4 01 3005 0000 0002
             5 30 0032 3612 3354
             6 05 0000 0000 3613
             7 Контрольная сумма
               блока
```

СТАНДАРТНЫЕ ПРОГРАММЫ

Стандартные программы и подпрограммы оформляются с учетом стандартного способа задания входной и выходной информации к ним. В качестве адресов для хранения входной и выходной информации (аргументов и результатов) используются

адреса 0002, 0003, 0004. Для программ, обрабатывающих сложные массивы (вектор-последовательности и пр.) информация задается в виде α-последовательностей, для которых адреса 0002, 0004 служат фиксаторами. Результат выдается по адресу 0003. Это либо адрес величины, если результатом является величина (например, в программе скалярного произведения векторов), либо фиксатор, если результат в свою очередь является массивом (например, в программе умножения матриц).

Стандартные программы оформляются в условных адресах; исключение составляют программы сменно-спаянной памяти, которые оформляются в истинных адресах. Обращение к подпрограммам, не содержащим в свою очередь обращения к подпрограммам (более низкого уровня), осуществляется по команде условного перехода на подпрограмму вида

K	30	3026	$K+1$	M

(с возвратом по регистру возврата). Здесь M — начальный адрес подпрограммы; по первому адресу указывается адрес величины, по знаку которой совершается переход на подпрограмму (или адрес какой-либо отрицательной константы, например 3026, если переход на подпрограмму является безусловным). При этом перед командой обращения к подпрограмме адреса аргументов или их фиксаторы засылаются соответственно по адресам 0002, 0003, 0004.

Приводим список программ, разработанных на 1/I 1962 г. Во всех случаях, где это особо не оговорено, возврат с подпрограмм по регистру возврата.

Подпрограмма перевода с двоичной в десятичную систему. В отличие от соответствующей подпрограммы, помещенной в постоянно-спаянной памяти (см. табл. 2), эта подпрограмма не использует групповых операций (что необходимо, например, при работе в режиме моделирования плавающей запятой).

Программа занимает 9 команд; аргумент '0002; переведенное число '0003; рабочие адреса 0002—0005.

Подпрограмма перевода целых чисел из двоичной системы в десятичную. Целое двоичное число, записанное во II адресе ячейки 0002, переводится в десятичную систему и выдается на печать. Групповые операции не используются; возврат — по регистру возврата; рабочие адреса 0002, 0003; с необходимыми константами программа занимает 23 команды.

Подпрограмма группового перевода из двоичной в десятичную систему. Переводится и печатается группа чисел, заданных в виде α-последовательности с ведущим адресом 0002. Содержимое α-последовательности сохраняется. В подпрограмме используются групповые операции; рабочие адреса 0003—0007; число команд 18.

Подпрограмма группового перевода и печати с использованием

режима групповой печати. Как и в предыдущей подпрограмме, переведенные числа засылаются в α-последовательность, которая может совпадать и с исходной α-последовательностью. Ведущий адрес второй последовательности 0003. Рабочие адреса 0003—0011; используются групповые операции; число команд 26.

Подпрограмма исправления ошибок в программе. В некоторую α-последовательность записываются правильные коды и их адреса в следующем порядке: первый код, его адрес; второй код, его адрес и т. д.

В наборную ячейку 3006 помещается в I адрес ведущий адрес последовательности, во второй — ее последний адрес (т. е. в НК 3006 можно поместить команду ввода α-последовательности — зоны исправлений).

Возврат — по регистру возврата; число команд — 10; рабочие адреса 0002—0005; используется операция *Ф*.

Подпрограмма вычисления синуса для $-1 < x < 1$ *с помощью ряда Тейлора.* Точность 9—10 знаков; рабочие адреса 0004—0011; групповые операции не используются; '0002 сохраняется; число команд 20.

Подпрограмма вычисления тангенса (котангенса). Для $x < \frac{\pi}{4}$ подпрограмма вычисляет $\operatorname{tg} x$, для $x \geqslant \frac{\pi}{4}$ — значение $\operatorname{ctg} x$. Результат выдается по двум адресам:

$$'0004 = \begin{cases} +0, & \text{если программа выдает } \operatorname{tg} x; \\ -0, & \text{если программа выдает } \operatorname{ctg} x. \end{cases}$$

$$'0003 = \begin{cases} \operatorname{tg} x, & \text{если } '0004 = +0; \\ \operatorname{ctg} x, & \text{если } '0004 = -0. \end{cases}$$

Групповые операции в программе не используются; число команд 21; '0002 сохраняется; рабочие адреса 0005—0007.

Подпрограмма вычисления $\operatorname{arctg} x$ *для* $-1 < x < 1$. Используется полином наилучшего приближения

$$\operatorname{arctg} x = \sum_{i=0}^{7} C_{2i+1} x^{2i+1};$$

$C_1 = 0{,}9999993329$; $C_7 = -0{,}1390853351$; $C_{13} = 0{,}0218612288$;
$C_3 = -0{,}3332985605$; $C_9 = 0{,}0964200441$; $C_{15} = -0{,}0040540580$.
$C_5 = 0{,}1994653599$; $C_{11} = -0{,}0559098861$;

Число команд 17; '0002 сохраняется; используются групповые операции.

Подпрограмма вычисления $\ln x$ *для* $0 < x < 1$. Программа занимает 34 команды и отличается от соответствующей впаянной программы лишь большей точностью.

Программа вычисления определителей n-го порядка. Элементы определителя задаются в виде α-последовательности с фиксатором 0002. Размерность определителя содержится по ведущему адресу $'\alpha = n$. Результат выдается в виде содержимого двух адресов: при $'0004 < 0$ значение определителя равно $\frac{'0003}{'0004}$, а при $'0004 > 0$ равно $'0003 \cdot '0004$.

Максимальный порядок определителя равен 25; число команд 121; используется операция *Ф*.

Подпрограмма транспонирования прямоугольной матрицы. Информация для программы задается в виде α-последовательности ее элементов по строкам, по ведущему адресу которой помещается размерность матрицы. Фиксатором этой последовательности служит адрес 0002. По I адресу последовательности задается число строк, по II — число столбцов. Транспонирование заключается в том, что столбец заданной матрицы записывается как строка транспонированной в виде α-последовательности с фиксатором 0003. В β-последовательности элементы размещаются по строкам транспонированной матрицы; по ведущему адресу β содержится число строк, а по адресу $\beta + 1$ — число столбцов транспонированной матрицы.

Содержимое α-последовательности и фиксаторов программой сохраняется; используется операция *Ф*; число команд 18; рабочие адреса 0004—0011.

Линейная комбинация векторов. Информация к программе задается в виде двух последовательностей. В α-последовательности задается число векторов, их размерность, элементы первого, элементы второго и т. д. (в указанном порядке). Фиксатором этой последовательности служит адрес 0003. В β-последовательности указываются константы, на которые умножаются векторы. Фиксатором этой последовательности служит адрес 0003. Результат выдается в α-последовательность, содержимые адресов α' и $\alpha + 1$ которой сохраняются.

Число команд 20; используется операция *Ф*; рабочие адреса 0002—0012.

Произведение матриц и вектора на матрицу. Начальной командой программы произведения вектора на матрицу служит команда $K + 13$, произведения матриц — команда $K + 0$.

Последовательность элементов вектора или первой матрицы задается в виде α-последовательности с фиксатором 0002. В первом случае по ведущему адресу последовательности содержится его размерность; во втором — число строк, а в следующем адресе помещается число столбцов, за которым следуют элементы матрицы по строкам.

Вторая матрица задается по строкам в виде β-последовательности с фиксатором 0004. По ведущему адресу помещается число строк, за ним — число столбцов матрицы.

Результат выдается в виде γ-последовательности с фиксатором 0003, по ведущему адресу которой помещается число строк полученной матрицы, а за ним — число столбцов, далее — элементы матрицы по строкам.

Обращение к программе по адресу 0001 (т. е. команда возврата формируется в ячейке 0001). Входную информацию программа сохраняет. '0002, '0003 и '0004 не сохраняются. Число команд 35; рабочие адреса 0002—0014; используется операция *Ф*.

Произведение матриц и вектора на матрицу с масштабированием (I вариант). Начальная команда программы произведения вектора на матрицу $K+0$, матрицы на матрицу $K+2$. Входная информация задается, как и к предыдущей программе, а выходная — в виде γ-последовательности по фиксатору 0003. I адрес последовательности содержит число строк получаемой матрицы, II — число ее столбцов. Далее следуют элементы матрицы и их масштабы:

$'\gamma = m$ — число строк матрицы;

$'(\gamma + 1) = n$ — число столбцов матрицы;

$'(\gamma + 2) = a_{11}$ — первый элемент матрицы;

$'(\gamma + 3) = \mu_{11}$ — масштаб первого элемента матрицы и т. д.

Выходная информация программой сохраняется; '0002, '0003, '0004 не сохраняются; используется операция *Ф*; число команд 43; рабочие адреса 0002—0015.

Произведение матрицы и вектора на матрицу с масштабированием (II вариант). Начальная команда программы произведения вектора на матрицу $K + 0$, матрицы на матрицу $K + 2$. В отличие от предыдущей программы для результирующей матрицы вводится общий масштабный множитель для всех ее элементов, который выдается по адресу 0016. Используется операция *Ф*; число команд 48; рабочие адреса 0002—0017.

Скалярное произведение векторов. Векторы задаются в виде α-последовательностей с ведущими адресами 0002 и 0004. Результат выдается по адресу 0003. Используется операция *Ф*; число команд 12; рабочие адреса 0005—0007.

Умножение квадратной матрицы, транспонированной к заданной, на вектор. Информация о матрице задается по строкам в виде α-последовательности с ведущим адресом 0002; информация о векторе задается по фиксатору 0004. Ведущий адрес вектора-результата 0003. Используется операция *Ф*; число команд 21; рабочие адреса 0005—0015.

Умножение квадратной матрицы на вектор. 0002 — фиксатор матрицы, заданной по строкам; '0002 равно размерности вектора; 0004 — фиксатор заданного вектора; 0003 — фиксатор вектора-результата (массивы для заданного вектора и вектора-результата различны). Используется операция *Ф*; число команд 20; рабочие адреса 0005—0015.

Вычисление функций Бесселя $I_n(x)$ для дробного аргумента и целого индекса

$$I_n(x) = \frac{x^n}{2^n n!}\left\{1 - \frac{x^2}{2(2n+2)} + \frac{x^4}{2\cdot 4(2n+2)(2n+4)} - \ldots\right\}.$$

'0002 — аргумент функции; '0004 — индекс функции Бесселя (задается во II адресе); '0003 — результат. Рабочие адреса 0004—0013; групповые операции не используются; точность 8 десятичных знаков; число команд 41.

Вычисление определенных интегралов по формуле Симпсона для четного числа делений n (режим фиксированной запятой). Входная информация представляет собой следующую α-последовательность с фиксатором 0002:

$'\alpha = n$ — число интервалов деления (четное число);

$'(\alpha+1) = h$ — шаг интегрирования;

$'(\alpha+2) = y_0$ | значения подынтегральной функции в точках деления интервала интегрирования.

$'(\alpha+n+1) = y_n$ |

Формирование содержимого α-последовательности осуществляется основной программой. Результат — содержимое адреса 0003. Число команд 39; рабочие адреса 0004—0014.

Вычисление определенных интегралов по формуле Симпсона, комбинированной с формулой трапеций, для нечетного числа делений (режим фиксированной запятой):

$$y = y_1 + y_2;$$

$$y_1 = \frac{h}{3}(y_0 + 4y_1 + 2y_2 + \ldots + 4y_{n-2} + y_{(n-1)});$$

$$y_2 = \frac{h}{2}(y_{n-1} + y_n).$$

Информация задается, как и в предыдущей программе. Число команд 48; рабочие адреса 0004—0014; результат '0003.

Транспонирование квадратной матрицы. 0002 — фиксатор исходной матрицы, заданной по строкам; 0003 — фиксатор транспонированной матрицы (получаемой также по строкам). Число команд 17; рабочие адреса 0004—0012; '0002, '0003 — сохраняются.

Произведение вектора на прямоугольную матрицу. Элементы вектора задаются в α-последовательности ($'\alpha = n$) с фиксатором 0002. Матрица задается по строкам в β-последовательности ($'\beta$ — число строк; $'(\beta+1)$ — число столбцов) с фиксатором 0004. 0003 — фиксатор γ-последовательности, содержащей элементы вектора результата ($'\gamma$ равно размерности вектора). Число команд 22; используется операция Φ; содержимые '0002, '0003, '0004 — сохраняются, а также сохраняется вся входная информация; рабочие адреса 0005—0013.

Обращение матриц по методу А. П. Ершова (в режиме фиксированной запятой). Исходная информация — общий масштабный

множитель и элементы матрицы по строкам — задается в виде последовательности с фиксатором 0002. В наборный код 3000 по II адресу вводится порядок матрицы. Допускается одновременное обращение нескольких матриц одного порядка. Число матриц указывается по III адресу наборного кода 3001.

Обратные матрицы выдаются (в десятичной системе) по столбцам (с интервалами между столбцами). На первом месте печатается масштабный множитель. Число команд 48; используется операция Φ.

Определение вещественных корней полинома n-й степени. Область определения корней находится программой по формуле

$$|x| \leqslant \frac{A}{|a_0|} + 1,$$

где $A = \max |a_i|$ $(i = 0,1, \ldots, n)$; a_0 — коэффициент при старшем члене.

Используется метод проб. Признаком выделения корня служит:

а) перемена знака функции;

б) перемена знака ее производной вблизи оси ox.

Вычисления ведутся в режиме плавающей запятой. Число команд 319. Коэффициенты полинома предварительно делятся на K, где $A < K < 10A$, и задаются в последовательности по нисходящим степеням (равные нулю — кодируются нулями). В наборных кодах задается: $'3000 = n$ — степень полинома (в I адресе); $'3001 = m$ — число полиномов (в I адресе).

Результат выдается на печать в плавающем режиме (порядок, мантисса). Точность 4—5 знаков.

Вычисление вещественных собственных чисел и соответствующих им векторов методом итераций. Находится максимальное по модулю собственное число:

$$Ay_0 = y_1; \quad Ay_{i-1} = y_i;$$
$$A'y_0' = y_1'; \quad A'y_{i-1}' = y_i';$$
$$\lambda_i = \frac{(y_i', y_i)}{(y_{i-1}', y_i)},$$

где A — исходная матрица; A' — транспонированная матрица; y_0, y_0' — произвольные нулевые приближения собственных векторов.

Собственным числом матрицы A является λ_i, для которого

$$|\lambda_i - \lambda_{i-1}| < \varepsilon,$$

где ε — малое число.

Соответствующие данному λ_i векторы y_i, y_i' являются собственными векторами матриц A и A' соответственно.

Матрица $A_1 = A - \lambda [y, y']$ имеет те же собственные числа, что и матрица A, за исключением уже вычисленного первого собственного числа λ. Соответствующее собственное число матрицы

A_1 будет равно нулю. Максимальное по модулю собственное число матрицы A_1, равное второму собственному числу матрицы A, также вычисляется методом итерации. То же касается соответствующих собственных векторов. $[y, y']$ означает матричное произведение

$$\begin{pmatrix} a_1b_1 & a_1b_2 & a_1b_3 & \dots & a_1b_n \\ a_2b_1 & a_2b_2 & \dots & \dots & a_2b_n \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ a_nb_1 & \dots & \dots & \dots & a_nb_n \end{pmatrix}$$

Допускается одновременное вычисление без останова собственных чисел ряда матриц одного порядка. Результаты выдаются на печать в следующем порядке: $M\lambda_1$ — масштабный множитель; λ_1 — первое собственное число; интервал и т. д.

Программой могут быть вычислены и кратные корни, если удачно задано нулевое приближение. Имеется возможность начинать с одного из четырех нулевых приближений:

1, 1, 1, . . . , 1
1, 0, 0, . . . , 0
0, 1, 0, . . . , 0
1, 0, 0, . . . , 0, 1

Выбор начального вектора определяется I адресом НК 3001.

Если ни одно из этих нулевых приближений не приводит к результату, то нулевые векторы следует задать в виде α-последовательности и ввести вслед за исходной матрицей. При этом следует опускать построение нулевых приближений. В программе используются подпрограммы скалярного произведения, умножения матрицы на вектор и умножения транспонированной матрицы на вектор.

Порядок матрицы задается во II адресе НК 3000 (в восьмеричной системе); '3001 = ε (точность приближения); '3002 = m — 1, где m — количество вариантов.

Число команд 268; максимальный порядок матрицы 23. Составитель программы В. С. Моторная.

Нахождение минимального собственного числа матрицы методом скорейшего спуска. Расчетные формулы:

$$Ax = \lambda Bx;$$

$$\lambda_m = \frac{(x_m,\ Ax_m)}{(x_m,\ Bx_m)};$$

$$x_{m+1} = x_m - \alpha_m r_m;$$

$$r_m = Ax_m - \lambda_m Bx_m;$$

$$\alpha_m = \frac{(r_m,\ r_m)}{(r_m,\ A_m r_m)}.$$

Информация о матрицах задается в виде α-последовательности, содержащей: порядок матрицы A; ее элементы по строкам; порядок матрицы B; ее элементы по строкам. В качестве нулевого приближения используется вектор с единичными компонентами. Порядок матрицы и количество вариантов задается на наборных кодах.

Программой выдается собственное число с масштабом и компоненты соответствующего ему собственного вектора. По желанию (переключением тумблера на пульте) могут печататься все промежуточные итерации.

Программа применима для случая вещественного минимального корня. Максимальный порядок матриц 24 (без внешней памяти). Составитель программы В. С. Моторная.

Решение системы линейных алгебраических уравнений методом квадратного корня. Информация о задаче — порядок матрицы, матрица по строкам, правые части — задается в виде α-последовательности по фиксатору 0002. Результат выдается в виде α-последовательности по фиксатору 0003. Программа работает в плавающем масштабе. Число команд 282.

Решение системы линейных алгебраических уравнений методом Зейделя. Информация задается, как в предыдущей программе. Программа реализована для сменно-спаянной памяти (64 команды).

Решение систем линейных алгебраических уравнений для матриц с неполным заполнением методом Лопшица. Исходная информация о матрице коэффициентов задается в виде двух последовательностей: α-последовательность шкал и β-последовательность коэффициентов. В ячейках шкалы задаются: в I адресе — номер строки ненулевого элемента матрицы, III — номер его столбца, во II — адрес самого элемента. В β-последовательности коэффициентов кодируются лишь различные и неравные нулю элементы матрицы.

Ведущие элементы α- и β-последовательностей задаются по II адресу наборных кодов 3000 и 3001 соответственно.

Число команд 83; рабочие ячейки 0001—0012; используется операция Ф. Составитель программы Е. И. Михайлова.

Прогноз ветра на среднем уровне. Входная информация — матрица начальных данных барической топографии. Выходная информация — матрица векторов скорости ветра. Для решения соответствующей системы уравнений в частных производных разработан специальный метод численного решения. Число команд 587. Входная матрица размером 23×19. Используются групповые операции и стандартные программы и константы спаянной памяти. Методика вычислений и программа разработаны Ю. С. Фишманом.

Решение уравнения Пуассона для задачи Дирихле для прямоугольной области методом блочной итерации с применением метода прогонки. На наборных кодах задаются:

'3000 = n — количество точек по оси x;
'3001 = m — количество точек по оси y;
'3003 = h^2, где h — шаг сетки по оси x;
'3004 = l^2, где l — шаг сетки по оси y;
'3005 = ε — заданная точность.

При использовании только оперативной памяти $mn \leqslant 360$; по программе можно решать уравнение Лапласа для сетки $mn \leqslant 720$. Число команд 81. Составитель программы Г. И. Визнюк.

Решение уравнения Лапласа для прямоугольной области методом блочной итерации с применением метода прогонки (с экономией памяти по методике, предложенной И. Н. Молчановым; см. ДАН УССР, № 3, 1962). На наборных кодах задаются:

'3000 = h^2, где h — шаг сетки по оси x;
'3001 = l^2, где l — шаг сетки по оси y;
'3002 = ε — заданная точность;
'3003 = m — количество узлов по оси y;
'3004 = n — количество узлов по оси x.

Память экономится за счет того, что значения функции задаются только на четных строках сетки. Оперативная память допускает решение уравнений для сетки $mn \leqslant 35^2$.

Число команд 305. Составитель программы Г. И. Визнюк.

Алгоритм получения псевдослучайных чисел с равномерным законом распределения. Используется метод вычитов. Число команд 132.

Алгоритм раскладки прямоугольных деталей на прямоугольных листах. Разработанный алгоритм дает (по сравнению с ручной раскладкой деталей) экономию в металле на 2%. Общее количество деталей до 100; на один лист укладывается до 12 деталей. Число команд 300. Алгоритм и программа разработаны Г. Я. Машбиц.

Вычисление корреляционных функций для больших числовых массивов (с использованием магнитного барабана). Число команд 150. Составитель программы Э. К. Ядренко.

Реализация случайного процесса с корреляционной функцией $e^{-c^2|\tau|}\cos b\tau$. Число команд 250. Составитель программы Э. К. Ядренко.

Решение транспортной задачи методом Лурье-Олейника. Используется полностью оперативная и внешняя память в объеме одного магнитного барабана. Матрица стоимостей размером $m \times n$ до 3000. Составитель программы С. В. Брановицкая.

Проектирование профиля автомобильных и железных дорог с фиксированными абсциссами точек перелома — пикетами (с учетом технических требований). Одновременно в память машины вводится информация о дороге протяженностью до 20 *км*. Применяется метод последовательного анализа вариантов, предло-

женный В. С. Михалевичем и Н. З. Шором. Составитель программы А. Н. Сибирко.

Программа для вывода буквенно-цифровой информации на экран электронно-лучевой трубки. Составитель программы В. К. Елисеев.

Программирующая программа для управляющей машины широкого назначения (УМШН) с входным адресным языком, основанная на принципе поэлементной расшифровки исходной информации. ПП составляет рабочие программы только для работы в режиме фиксированной запятой. Объем ПП около 290 адресных строк или 600 команд машины «Киев». ПП обрабатывает следующие адресные формулы: засылки, предикатные, вхождения, метки безусловного перехода, нестандартные формулы, записанные в кодах УМШН, формулы печати, формулы останова.

После обработки программирующей программой исходной информации каждой зоны на печать выводится программа задачи в кодах машины УМШН, причем приказы рабочей программы печатаются с их истинными номерами. Программа составлена Р. А. Годзь, Г. А. Полищук, А. И. Стиранкой под рукодством Е. Л. Ющенко.

Гидравлический расчет водопроводных и газовых сетей. Решается нелинейная система гидравлического расчета водопроводных и газовых сетей. Допускается расчет сети, состоящей из k участков; в каждом кольце сети содержится максимум m участков. Для размещения данных в оперативной памяти требуется, чтобы $2k + 3m \leqslant 808$. Число команд 261. Время расчета сети из 250 участков 30 *мин*. Программа составлена В. С. Квакушем по методике, предложенной Б. Н. Пшеничным.

Реализация алгоритма определения времени повалки бессемеровского конвертера на заданном содержании углерода в металле. Текущая информация о ходе продувки конвертера задается датчиками, установленными в бессемеровском цехе завода им. Дзержинского (г. Днепродзержинск), преобразуется в специальном устройстве в цифровую форму, усредняется и передается по телеграфному каналу на приемный аппарат, установленный в машинном зале машины «Киев». Специальный дешифратор преобразует телеграфные коды в двоичные и вводит в запоминающее устройство машины «Киев».

Программа определяет момент повалки, и по телеграфному каналу передается управляющий сигнал в цех завода.

Работа выполнена Никитиным А. И. и другими под руководством В. М. Глушкова, К. С. Гаргера и Л. Н. Дашевского.

Определение оптимального режима процесса в карбонизационной колонне содового комбината. Решается аналогично предыдущей задаче для управляемого объекта — содового комбината (г. Славянск). Руководители работы Б. Н. Малиновский и А. Б. Тютинников, исполнители И. А. Янович, Р. С. Цвигун.

Тяговые расчеты для тепловоза. Задается информация о спрямленном профиле пути в виде длин элементов и приведенных уклонов. Программа выдает времена хода и скорости локомотива по элементам и по перегонам, расход электроэнергии, перегрев двигателя, механическую работу и другие промежуточные данные. Время счета 5 *мин* на 100 *км* пути. Число команд 400.

Методика и алгоритм разработаны Н. З. Шором и А. А. Алексеевым.

Решение систем линейных алгебраических уравнений методом сопряженных градиентов с минимизацией нормы отклонения приближенного решения от точного (см. Шаманский В. Е., «Украинский математический журнал», № 1, 1962). Для матриц с неполным заполнением используются шкалы и кодируются лишь ненулевые элементы.

Если n — порядок системы, m — число ненулевых элементов, то для размещения информации в оперативной памяти должно выполняться условие $6n + m \leqslant 673$. Для матриц с полным заполнением $n \leqslant 23$. Число команд 319. Составители программы Г. П. Правоторова и М. Ф. Яковлев.

Решение систем линейных алгебраических уравнений методом сопряженных градиентов с минимизацией нормы невязки. Как и в предыдущей программе, для матриц с неполным заполнением используются шкалы. Для размещения информации в оперативной памяти в случае матрицы с неполным заполнением должно выполняться условие $6n + m \leqslant 608$, где $n \leqslant 41$, m — число ненулевых элементов; для матриц с полным заполнением $n \leqslant 21$. Число команд 370. Составители программы Г. П. Правоторова и М. Ф. Яковлев.

Решение систем линейных алгебраических уравнений с симметричной матрицей усовершенствованным методом Гаусса (см. пример программы, составленной ПП-АК).

Решение систем дифференциальных уравнений методом Рунге-Кутта (см. пример блока сменно-спаянной памяти).

Решение уравнения в частных производных параболического типа с краевыми условиями первого рода. Решается уравнение теплопроводности и уравнение Буссинеску для областей с произвольной конфигурацией (вид уравнения задается специальным признаком). Информация о виде области задается с помощью специальных логических шкал. Используется метод сеток с явной разностной схемой.

Максимальное количество узлов сетки 750. Число команд 120. Составитель программы В. Ф. Темченко.

Аппроксимация функций двух переменных методом наименьших квадратов (см. А. С. Новик и В. Е. Шаманский, ДАН УССР, 1962, № 3). Входная информация — таблица значений функции

двух переменных $f(x, y)$ в узлах прямоугольника. Программа выдает коэффициенты аппроксимирующего полинома вида

$$P_{nm}(x, y) = \sum_{r=0}^{n} x^r \sum_{s=0}^{m} c_{sr} y^s.$$

Число команд 408. Оперативная память позволяет строить полиномы, для которых выполняется соотношение

$$\frac{\alpha\beta}{2} + \alpha + \beta + 3\max(\alpha, \beta) + 2\max(m, n) + \beta(m+1) + \frac{m^2+3m}{2} \leqslant 609,$$

где α — число узлов по x; β — число узлов по y; m и n — степени полиномов по x и по y соответственно.

Программа составлена А. С. Новик.

Алгоритм получения псевдослучайных чисел с равномерным распределением, основанный на операциях сдвиг (13) и равнозначности (17). Период чисел превышает 500 000. Число команд (с проверкой условия случайности и равномерности) 105. Составитель программы В. С. Квакуш.

Расчет диаметров и давлений газовых и водопроводных сетей. Пусть m — количество узлов, n — количество участков, k — максимальное количество участков в одном узле. Оперативная память позволяет осуществить расчеты при выполнении условия $m + 2n + 4k \leqslant 706$. Число команд 288. Программа составлена В. С. Квакушем по методике, предложенной Б. Н. Пшеничным.

Приведем далеко неполный список реализованных на машине «Киев» алгоритмов моделирования простейших процессов мышления, творчества и др.:

Алгоритм морфологического анализа русского языка, предложенный И. А. Мельчуком. Число команд 360. (См. Н. М. Грищенко, «Проблемы кибернетики», вып. 6, 1961).

Алгоритм синтаксического анализа русского языка, предложенный И. А. Мельчуком. Используется вся оперативная память и один магнитный барабан. Составитель программы С. Н. Якименко.

Алгоритм обучения распознавания смысла простых предложений (см. В. М. Глушков, Н. М. Грищенко, А. А. Стогний, Принципы построения самообучающихся систем, Гостехиздат УССР, 1962). По идее В. М. Глушкова построена самосовершенствующаяся система алгоритмов, которая после сообщения ей некоторого числа N случайно выбираемых фраз данной конструкции правильно распознает осмысленность любой фразы этой же конструкции. Алгоритм и программа разработаны А. А. Стогнием и Н. М. Грищенко.

Моделирование процессов биологической эволюции (см. А. А. Летичевский, Принципы построения самообучающихся систем, Гостех-

издат УССР, 1962). Алгоритмы программы разработаны А. А. Летичевским и А. А. Дородницыной под руководством В. М. Глушкова.

Алгоритм обучения распознаванию простейших геометрических фигур (см. В. М. Глушков, В. А. Ковалевский, В. И. Рыбак, Принципы построения самообучающихся систем, Гостехиздат УССР, 1962).

Корреляционный метод распознавания изображений. Авторы программы В. А. Ковалевский, В. К. Елисеев.

Алгоритм Глушкова синтеза автомата по представляемому им событию (см. В. М. Глушков, А. А. Стогний, «Вычислительная математика и математическая физика», 1961, № 3).

Минимизация автоматов по методу Ауфенкампа-Хона (см. А. А. Стогний, «Вычислительная математика и математическая физика», 1961, № 3). Программа составлена В. П. Клименко.

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ РЕЖИМА ПЛАВАЮЩЕЙ ЗАПЯТОЙ НА МАШИНЕ „КИЕВ"

Общая характеристика моделирующей программы (МП) *

Расчет и согласование масштабов при подготовке задач на машинах с фиксированной запятой зачастую сопряжен с трудностями, поскольку при реализации тех или иных программ осуществляются последовательности операций, на максимальные значения входных и выходных величин которых могут не накладываться никакие ограничения. В ряде случаев каждая новая задача выдвигает свою методику расчета масштабов, в результате масштабирование перерастает в искусство, не поддающееся в целом стандартизации. Поэтому для увеличения эффективного быстродействия универсальной машины с фиксированной запятой в набор ее операций следует включать специальные операции, облегчающие программную реализацию режима плавающей запятой. Среди этих операций, наряду с операциями логическими и широким набором операций передач управления, в машине «Киев» существенная роль принадлежит операции нормализации.

При разработке алгоритмов программной реализации режима плавающей запятой на машине «Киев» преследовалась цель создать такие условия, при которых допускается составление программы в режиме фиксированной запятой, а ее фактическая реализация может осуществляться в плавающем режиме. Такого рода алгоритмы принято называть моделирующими, поскольку в них моделируется программно работа недостающих электронных цепей машины, необходимых для реализации операций в плаваю-

* Алгоритмы МП разработаты И. В. Сергиенко и Л. Н. Иваненко под руководством Е. Л. Ющенко.

щем режиме. Допустимость построения моделирующих алгоритмов следует из наличия точного описания правил преобразования кодов в плавающем режиме (при выбранном способе кодирования), с одной стороны, и из универсальности самой машины, — с другой.

Основная тенденция настоящего варианта МП состоит в минимизации памяти, в отдельных случаях за счет времени вычислений.

Исходные данные для счета задаются в виде последовательности чисел с единым масштабом, который помещается в конце последовательности. Числа занимают по одной ячейке памяти. Младшие семь разрядов используются для порядков. При выдаче результатов одно число печатается в две строки (мантисса и порядок).

Программы для вычисления в режиме плавающей запятой пишутся в языке машины «Киев», на который наложен ряд ограничений. Это означает, что программист, написав программу с указанными ограничениями для счета в режиме фиксированной запятой, при неудачном подборе масштабов может перейти на счет с помощью МП, которая решит ту же задачу в режиме плавающей запятой.

Назовем режим работы машины в фиксированном режиме режимом I и соответственно в плавающем режиме — режимом II ее работы. Тогда, если при составлении программы будем пользоваться приведенным ниже набором операций машины «Киев», для решения задачи в режиме II достаточно набрать во II адресе НК 3000 номер первой команды рабочей программы и управление передать ячейке 3300 (на начало МП).

Практическая проверка системы моделирования на ряде задач, в том числе специально подобранных, с учетом статистических данных по частоте употребления различных операций при решении задач вычислительного характера, просчитанных в обоих режимах, показала, что скорость выполнения программы за счет режима II снижается примерно в 10 раз.

Программа выполнена в виде сменно-спаянной памяти (256 кодов). Оперативная память полностью остается свободной, за исключением первых 24 ячеек (0001 — 0032).

Набор операций для МП

При разработке МП имелось в виду, что в режиме плавающей запятой будут решаться задачи сугубо вычислительного характера. Поэтому основное внимание уделялось достижению минимального использования моделирующей программой оперативной памяти и высокой скорости выполнения ею арифметических операций. В связи с этим пришлось пойти на некоторое обеднение логических возможностей машины, что выразилось в запрещении

использования ряда операций. Все же следует указать, что допущенные для режима II логические операции выполняются сравнительно быстро и скорость реализации программ для решения задач вычислительного характера при увеличении используемого числа логических операций резко возрастает.

В набор операций машины «Киев» для режима II входят следующие операции: сложение (01), вычитание (02), сложение команд (03), вычитание модулей (06), циклическое сложение (07), умножение с округлением (11), деление (12), логический сдвиг (13), логическое сложение (14), логическое умножение (15), передача управления по равенству (16), сложение по модулю 2 (17), ввод чисел (20), ввод команд (21), вывод (печать) (22), обращение к внешней памяти (23, 24, 25), передача управления по знаку числа (31), останов (33), чтение по фиксатору (34), нормализация (35).

Моделирующая программа не воспринимает признаков модификации адресов. Таким образом, участки программ, использующие модификацию, не должны включаться в режим моделирования. Однако это не означает полного исключения использования *А*-регистра в режиме II. *А*-регистр может быть использован в связи с операцией *Ф* (34) для понижения ранга адреса и формирования управления циклами итерационного типа. Учитывая это, при программировании схем обозревания массивов следует пользоваться операцией *Ф*.

Структура моделирующей программы

Моделирующая программа (МП) по своей структуре напоминает алгоритм поэлементной расшифровки адресных функций, приведенный в гл. III. Моделирующая программа с помощью алгоритма, названного коммутатором операций, выделяет код очередной моделируемой команды, по которому передает управление соответствующей подпрограмме. Немоделируемые операции при этом выполняются (в режиме фиксированной запятой) непосредственно.

Коммутатор команд (команды 3300—3316, 3557—3573)

```
3300 01 0000 3621 0026
   1 01 0000 3622 0027
   2 02 3000 3011 0014
   3 14 3050 3026 0012
   4 01 0014 3011 0014
   5 34 0014 0000 0011
   6 15 0011 3025 0013
   7 16 0013 3600 3430
3310 16 0013 3045 3333
   1 16 0013 3044 3333
   2 16 0013 3050 3461
   3 16 0013 3601 3333
   4 16 0013 3602 3323
   5 16 0013 3603 3317
   6 31 0000 3557 0000
   7 01 0011 0000 0010
3320 01 0014 3011 0014
   1 34 0014 0000 0011
```

```
   2 31 0000 0010 0000
   3 13 3604 0011 0014
   4 34 0014 0000 0014
   5 31 0014 3326 3330
   6 15 0011 3023 0014
   7 31 0000 3305 0000
3330 13 3624 0011 0011
   1 15 0011 3023 0014
   2 31 0000 3305 0000
3557 16 0013 0426 3510
3560 16 0013 0427 3564
   1 30 3026 3562 0011
   2 16 3026 0013 3630
   3 31 0000 3304 0000
   4 13 3624 0011 0017
   5 15 0011 3032 0021
   6 14 0021 3574 0021
   7 01 0021 0000 0011
3570 31 0000 3561 0000
   1 15 0017 3023 0014
   2 02 0014 3011 0014
   3 32 0000 0000 0000
```

Моделирование операций типа сложения

Первую группу операций составляют сложение (01), вычитание (02) и вычитание модулей (06), которые моделируются одной подпрограммой. Для выполнения этих операций производится засылка аргументов операций в ячейки 0020 (первая компонента) и 0015 (вторая компонента). Подпрограмма отделяет мантиссы от порядков: 0021-мантисса I слагаемого; 0022-порядок I слагаемого; 0023-мантисса II слагаемого; 0024-порядок II слагаемого и выполняет операцию. При этом порядки чисел выравниваются, и над мантиссами выполняется операция. Результат операций нормализуется, мантисса объединяется с порядком в одну ячейку и пересылается по III адресу моделируемой команды. Если порядки равны или их разность превышает число допустимых сдвигов, т. е. результат операции с точностью до знака равен одной из компонент, выравнивание порядков с целью экономии времени опускается, и результат непосредственно посылается по III адресу моделируемой команды. Программа работает с учетом включения тумблера переполнения. Это позволяет автоматизировать процесс вычислений в случае возможных переполнений разрядной сетки.

Подпрограмма операций типа сложения

[сложение (01), вычитание (02), вычитание по модулю (06)]

```
3333 14 0013 3613 0025
   4 13 3624 0011 0017
   5 13 3604 0011 0016
   6 34 0011 0000 0015
   7 34 0016 0000 0020
3340 15 0020 3605 0021
   1 15 0020 3606 0022
   2 15 0015 3605 0023
   3 15 0015 3606 0024
   4 02 0022 0024 0016
   5 31 0016 3346 3410
   6 05 0016 3607 3352
   7 34 0017 0000 0000
3350 14 0000 0020 4000
   1 31 0000 3304 0000
   2 02 0000 0016 0016
   3 31 0023 3354 3356
   4 13 0016 0023 0023
   5 31 0000 0025 0000
   6 15 0023 3035 0023
```

```
   7 13 0016 0023 0023
3360 02 0000 0023 0023
   1 31 0000 0025 0000
   2 05 0022 3610 3365
   3 33 0000 0000 0000
   4 31 0000 3363 0000
3365 01 3012 0022 0022
   6 11 0021 3042 0021
   7 01 3042 0021 0021
3370 15 0021 3611 0016
   1 16 0000 0016 3375
   2 01 0021 3612 0021
   3 31 0000 3375 0000
   4 31 0000 3362 0000
   5 15 0021 3605 0021
   6 16 0021 0000 3446
   7 34 0017 0000 0000
3400 14 0021 0022 4000
   1 31 0000 3304 0000
   2 35 0021 0016 0021
   3 01 0016 0022 0022
   4 04 3012 0022 3370
   5 34 0017 0000 0000
   6 15 3026 0021 4000
   7 31 0000 3304 0000
3410 16 0000 0016 0025
   1 05 0016 3607 3415
   2 14 0000 0000 0021
   3 14 0024 0000 0022
   4 31 0000 0025 0000
   5 31 0021 3416 3421
   6 13 0016 0021 0021
   7 14 0024 0000 0022
3420 31 0000 0025 0000
   1 15 0021 3035 0021
   2 13 0016 0021 0021
   3 02 0000 0021 0021
   4 14 0024 0000 0022
   5 31 0000 0025 0000
```

Моделирование операций типа умножения

Вторая группа — операции умножения (11) и деления (12) — также моделируется одной подпрограммой.

При умножении порядки складываются, а результат умножения мантисс нормализуется. При делении проверяется неравенство знаменателя нулю и возможность выполнения деления. При невыполнении первого условия происходит автоматический останов. При этом по адресу 0023 содержится нуль. Если мантисса делителя больше мантиссы делимого, порядки вычитаются и происходит деление мантисс (операция возможна); в противном случае предварительно происходит сдвиг мантиссы числителя с соответствующим увеличением порядка. Как и в случае операций типа сложения, при выполнении этой подпрограммы мантиссы отделяются от порядков; нормализованный результат помещается по III адресу моделируемой команды.

Подпрограмма работает с учетом включения тумблера переполнения.

Программа операций типа умножения (команды 3430—3460, 3461—3505)

```
3426 22 0000 0000 0000
   7 16 0000 0000 0000
3430 34 0011 0000 0015
   1 13 3624 0011 0017
   2 13 3604 0011 0016
   3 34 0016 0000 0020
   4 15 0020 3605 0021
   5 15 0020 3606 0022
```

```
     6 15 0015 3605 0023
     7 15 0015 3606 0024
3440 01 0022 0024 0022
     1 02 0022 3611 0022
     2 05 0022 3610 3445
     3 33 0000 0000 0000
     4 31 0000 3443 0000
     5 04 3012 0022 3451
     6 34 0017 0000 0000
     7 14 0000 0000 4000
3450 31 0000 3304 0000
     1 11 0021 0023 0021
     2 05 0021 0000 3456
     3 35 0021 0016 0021
     4 01 0016 0022 0022
     5 04 0000 0022 3376
     6 34 0017 0000 0000
     7 14 0000 0000 4000
3460 31 0000 3304 0000
     1 34 0011 0000 0015
     2 16 0015 0000 3363
     3 13 3624 0011 0017
     4 13 3604 0011 0016
     5 34 0016 0000 0020
     6 16 0020 0000 3477
     7 15 0020 3607 0021
3470 15 0020 3607 0022
     1 15 0015 3605 0023
     2 15 0015 3606 0024
     3 02 0022 0024 0022
     4 10 0022 3611 0022
     5 05 3610 0022 3363
     6 04 0000 0022 3502
     7 34 0017 0000 0000
3500 14 0000 0000 4000
     1 31 0000 3304 0000
     2 11 0021 3042 0021
     3 01 0022 3012 0022
     4 12 0021 0023 0021
     5 31 0000 3370 0000
```

Ввод и вывод чисел в режиме II

Форма записи чисел для работы в режиме плавающей запятой следующая: младшие разряды с первого по седьмой отведены под порядок, представленный условно в виде $a = p + 64$, где p — действительный порядок числа. Так как $-64 \leqslant p \leqslant 63$, то условные порядки составляют диапазон [0—127], а числа соответственно — от 2^{-64} до 2^{+63}. Нулем считается пустая ячейка. Разряды с 8-го по 40-й заняты мантиссой. 41-й разряд — знак мантиссы.

Для ввода в машину числа представляются в ненормализованном виде в диапазоне $-1 < x < 1$, т. е. так, как принято вообще для машины «Киев». Числа, не входящие в диапазон $(-1, 1)$, вводятся следующим образом. Все n чисел делятся на подходящее число M и задаются в виде α-последовательности, на последнем месте которой ставится число M^{-1}. Всего в последовательности получается $(n + 1)$ чисел. Если $M = 1$, вместо M^{-1} записывается нуль. Выбор числа M произволен. Это может быть степень числа 2, 10 и т. д.

Подпрограмма ввода чисел по команде «ввод чисел» (20) вводит α-последовательность чисел с фиксированной запятой в принятом масштабе M, переводит их с учетом масштаба в плавающий режим и преобразованные числа помещает по тем же адресам. Адрес, по которому содержался множитель M, может использоваться для других целей.

Подпрограмма ввода чисел (команды 3630—3705)

```
3630 35 0000 0006 0000
   1 14 0000 0000 0017
   2 15 0011 3023 0005
   3 15 0011 3023 0005
   4 11 0004 3040 0003
   5 02 0005 0003 0007
   6 11 0003 3040 0005
   7 14 0004 0005 0005
3640 14 3630 0005 0004
   1 14 3050 3026 0005
   2 30 3026 3643 0004
   3 34 0003 0000 0000
   4 16 0000 4000 3650
   5 15 4000 3605 4000
   6 01 0006 3611 0006
   7 14 0006 4000 4000
3650 05 0007 0017 3655
   1 01 3011 0017 0017
   2 03 3016 0004 0004
   3 01 3011 0003 0003
   4 31 0000 3642 0000
   5 14 3011 0000 0017
   6 34 0011 0000 0000
   7 16 4000 0000 3304
3660 15 3605 4000 0004
   1 15 3606 4000 0005
   2 11 0011 3040 0006
   3 34 0006 0000 0000
   4 15 3605 4000 0022
   5 15 3606 4000 0010
   6 16 4000 0000 3702
   7 12 0022 0000 4000
3670 31 0000 3674 0000
   1 11 0022 3042 0022
   2 01 0010 3012 0010
   3 30 0000 3667 0000
   4 02 0010 0005 0010
   5 01 0010 3611 0010
   6 35 4000 0003 4000
   7 01 0003 0010 0010
3700 15 4000 3605 4000
   1 14 4000 0010 4000
   2 16 0017 0007 3304
   3 01 3011 0017 0017
   4 03 3011 0006 0006
   5 31 0000 3663 0000
```

Подпрограмма печати осуществляет перевод мантисс и порядков в десятичную систему и выдает результат на печать в виде двух строк: мантисса числа и порядок числа. Аргумент подпрограммы помещается по адресу 0004.

Подпрограмма печати

```
3510 14 0000 0000 0007
   1 15 0004 3606 0005
   2 15 0004 3605 0006
   3 16 0005 0000 3551
   4 05 3625 0005 3524
   5 05 3623 0005 3532
   6 11 0006 3626 0006
   7 02 0007 3012 0007
3520 35 0006 0004 0006
   1 01 0004 0005 0005
   2 01 0005 3615 0005
   3 31 0000 3515 0000
   4 11 0006 3614 0006
   5 35 0006 0004 0006
   6 01 3012 0007 0007
   7 01 0004 0005 0005
3530 01 0005 3627 0005
   1 31 0000 3514 0000
   2 05 3611 0005 3536
   3 11 3042 0006 0006
   4 01 3012 0005 0005
   5 31 0000 3532 0000
   6 01 0000 0006 0002
   7 30 3026 3540 3100
3540 05 0007 3616 3547
   1 31 0007 3542 3544
   2 02 0007 3617 0004
   3 31 0000 3545 0000
   4 01 0007 3617 0004
   5 14 0004 3620 0004
   6 22 0003 0004 3304
   7 14 0007 0000 0004
3550 22 0003 0004 3304
   1 22 0006 0007 3304
```

Вычисление элементарных функций в режиме II

Для вычисления значений элементарных функций разработаны специальные подпрограммы в режиме плавающей запятой. Аргумент элементарной функции помещается, как обычно, в ячейку 0002, а результат выдается в ячейках 0003 и 0004. После выполнения подпрограммы управление передается следующей команде.

В число разработанных подпрограмм входят: $\sqrt{x}$, $\sin x$, $\cos x$, e^x, $\ln x$, $\arcsin x$, $\arccos x$.

Обращение к подпрограмме вычисления той или иной элементарной функции, реализованное в моделируемой программе с помощью команды условного перехода на подпрограмму, преобразуется моделирующей программой к виду

V (14)		γ	0001
УПЧ (31)	·	*N*	

Здесь γ — адрес команды, к которой надо перейти после обращения к подпрограмме: *N* — начальный адрес подпрограммы.

Поскольку выполнение команд рабочей программы при моделировании осуществляется с помощью операции *Ф*, использование подпрограмм из постоянной памяти в режиме II не допускается, так как *A*-регистр в машине «Киев» имеет только 10 разрядов.

Специальные константы, используемые при работе всех программ МП

```
3600 11 0000 0000 0000
   1 06 0000 0000 0000
   2 31 0000 0000 0000
   3 25 0000 0000 0000
   4 20 0000 0000 0014
   5 37 7777 7777 7600
   6 00 0000 0000 0177
   7 00 0000 0000 0041
3610 00 0000 0000 0176
   1 00 0000 0000 0100
   2 00 0000 0000 0200
   3 00 0021 0023 0021
3614 14 6314 6314 6320
   5 00 0000 0000 0004
   6 00 0000 0000 0011
   7 00 0000 0000 0012
3620 00 0000 0000 0020
   1 05 0000 0000 3402
   2 05 0000 0000 3362
   3 00 0000 0000 0075
   4 00 0000 0000 0014
   5 00 0000 0000 0101
   6 12 0000 0000 0000
   7 20 0000 0000 0003
```

Округление чисел

При моделировании арифметических операций, а также при формировании в одну ячейку чисел, мантисса и порядок которых разделены, происходит округление чисел по правилу: если

старший разряд отбрасываемой части числа равен единице, то к младшему разряду оставшегося числа прибавляется единица; если же старший разряд отбрасываемой части числа равен нулю, то оставшаяся часть числа не изменяется. При выводе чисел, осуществляемом с помощью подпрограммы печати, округление их не учитывается с целью экономии времени. Округление чисел при выполнении арифметических операций в режиме II позволяет увеличить точность решения задач.

ТЕСТ-ПРОГРАММЫ ДЛЯ МАШИНЫ «КИЕВ».

Задача программного контроля машины состоит в том, чтобы, с одной стороны, выявить неисправности в ее работе и, с другой, получить максимально возможную информацию о том, в каком месте эта неисправность находится. В соответствии с этим тест-программы, предназначенные для проверки машины, делятся на две категории. Программы первого типа предназначены для обнаружения факта неправильной работы того или иного устройства. Для них характерно наличие большого числа разнообразных примеров с заранее неизвестными ответами. С целью проверки правильности решения задачи один и тот же ответ получают разными путями. Тест-программы второго типа предназначены для уточнения места неисправности в неправильно работающем устройстве, например в неисправном разряде сумматора арифметического устройства. В таких программах решается сравнительно небольшое число контрольных примеров с заранее известными результатами. Сопоставление неверных результатов с известными дает возможность локализовать место неисправности. Однако в настоящее время не существует еще достаточно хороших методов анализа схем и построения системы примеров, которые позволяли бы обнаружить любую неисправность

Методы, разработанные С. В. Яблонским [23] для релейно-контактных схем, малопригодны в рассматриваемом случае, так как требуют полного перечня всех возможных неисправностей. В связи с этим при разработке тест-программ для машины «Киев» пользовались некоторыми упрощающими предположениями. Так, в частности, предполагается, что для полной проверки какого-либо устройства достаточно подавать в него такие сочетания кодов, при которых в случае правильной работы в каждый его блок (например, в каждый разряд сумматора арифметического устройства) подаются все возможные комбинации входов.

Второе затруднение состоит в следующем. При проверке какого-либо устройства приходится пользоваться работой других устройств, которые в свою очередь могут быть неисправными. Вообще говоря, в некоторых (довольно редких) случаях могут появляться такие неисправности, при наличии которых с помощью программных методов контроля невозможно ответить на вопрос,

какое из устройств работает неверно. Для снижения влияния этого обстоятельства выполняется большое число однотипных примеров, у которых варьируются данные, влияющие на режим работы только проверяемого устройства.

В соответствии с изложенным для программного контроля машины «Киев» разработаны и отлажены следующие тест-программы: арифметического устройства; оперативного и пассивного запоминающих устройств; устройства управления; магнитных барабанов; печати.

Тесты устройства управления (УУ)

Работа УУ проверяется тремя тест-программами. *Первая тест-программа* предназначена для проверки правильности передач управления. В программе производятся передачи управления всеми возможными способами, в частности с регистром возврата, в различные места памяти. При этом в счетчик команд и на регистр возврата засылаются 0 и 1 во все разряды. При каждой правильной передаче управления в специально отведенный массив засылаются признаки (—0), которые проверяются после всех передач и печатаются, если хотя бы один признак отсутствует. При неправильной передаче может произойти останов раньше, чем напечатаются признаки. В этом случае их можно напечатать, передав управление соответствующей команде.

Вторая тест-программа проверяет *А*-регистр, сумматор адреса (СмА) и регистр циклов *Ц*, т. е. выполнение групповых операций. Счетчик адреса проверяется вызовом содержимого нескольких ячеек с помощью операции *Ф*. Регистр циклов проверяется с помощью операции *НГО*, при выполнении которой передача управления происходит в зависимости от того, совпадают или не совпадают содержимые регистров *А* и *Ц*. Таким образом, для проверки используется устройство совпадения, однако гарантии в его исправности не требуется, так как неисправность устройства совпадения будет обнаружена при используемых кодах.

При выполнении операции *ОГО* происходит сложение в сумматоре адреса. Правильность этого сложения проверяется по передаче управления после выполнения команды, т. е. с помощью устройства совпадения. При получении неверного результата происходит останов.

Третья тест-программа УУ проверяет регистры *А* и *С*, сумматор адреса и счетчик *С* более полно, а именно: правильность работы счетчика *С* как счетчика адресов ОЗУ и ПЗУ; правильность занесения в регистры *А*, *Ц* и счетчик *С* кодов из всех единиц и одного нуля, а также переносы в СмА. Здесь же проверяется работа 11-го разряда регистра возврата *Р*, который не проверяется первым тестом. При неправильной работе этих устройств происходит останов.

Четвертая тест-программа УУ проверяет работу СмА на кодах вида 0 ... 01 ... 1 + 0 ... 01 ... 1; работу признака групповой операции (ГО) в III адресе операций передач управления, а также во всех адресах операции нормализации, которая по управлению является наиболее сложной из всех арифметических операций.

При сбое происходит останов. Работа тестов УУ управляется знаком наборной ячейки 3000:

Тесты	I	II	III	IV	Все
Знак 3000	—	—	—	—	+
Начало	0007	0125	0240	0566	0007

Тесты оперативного запоминающего устройства (ОЗУ)

Каждая ячейка ОЗУ для считывания имеет магнитный ключ. Магнитные ключи размещаются в узлах матрицы, называемой координатной сеткой. Эта сетка состоит из 32 строк (координата x) и 32 столбцов (координата y). На запись каждой координаты требуется пять двоичных разрядов. Например, у ячейки с адресом 1043 (1 000 100 011 в двоичной системе) $x = 17$; $y = 3$; поэтому 10-разрядный адрес числа разбивается на два 5-разрядных числа, каждое из которых поступает на формирователи адреса Φ_{ax} и Φ_{ay}.

Работа ОЗУ проверяется четырьмя тестами, каждый из которых выясняет наличие одного вида неисправности, а именно:

1) неправильная запись или считывание кодов в ячейках ОЗУ;
2) неустойчивое хранение кодов в ОЗУ при многократном считывании одной ячейки и полувыборка;
3) неправильное считывание при большой нагрузке Φ_a;
4) неправильная выборка кода по адресу.

Работа названных тестов управляется знаком наборной ячейки 3000. Если этот знак — плюс, то работают по очереди все четыре теста (после четвертого — первый), в противном случае работает только тот тест, которому передается управление с пульта:

Тесты	I	II	III	IV	Все
3000	—	—	—	—	+
Начало	0022	0045	0100	0117	0022

При неисправности какой-либо ячейки ОЗУ, т. е. когда из нее считывается неверное число, во всех четырех тестах происходит останов. В этот момент на регистре команд K устанавливается команда останова

33		k	n

Здесь число k означает номер теста; n — количество уже прошедших чтений.

Адрес ячейки, при чтении которой произошел останов, находится в A-регистре; код, который должен в ней быть, — в ячейке 0020.

Тесты ОЗУ запрограммированы в двух вариантах; на начале памяти (проверяются ячейки 0200—1777) и на конце (проверяется массив ячеек от 0001 до 1600).

Первый тест ОЗУ производит однократную запись и многократное считывание в ячейках проверяемого массива специально подобранных кодов:

$$00 \ldots 0;\ 11 \ldots 1;\ 01 \ldots 0;\ 10 \ldots 1$$

и единиц нулевого, I, II и III адресов. Кроме того, есть возможность записывать код, набранный на НК № 3007. Для этого нужно в НК № 3001 набрать единицу 41-го разряда (минус). В противном случае будут записываться коды, перечисленные выше.

Второй тест ОЗУ. «Щекотка» и полувыборка. В ячейки проверяемого массива, расположенные на диагонали матрицы-памяти (№ 0000, 0041, 0102, 0143, ..., 1736, 1777), записываются во все разряды нули, в остальные ячейки — единички. Диагональные ячейки многократно считываются. После этого производится проверка содержимого других ячеек. Этот тест обнаруживает одну из двух следующих неисправностей:

1) появление нуля на месте единицы при многократном считывании нулей или наоборот;

2) полувыборка; в этом случае по адресам неисправных ячеек можно выяснить, при считывании какой из диагональных ячеек произошла полувыборка.

Третий тест ОЗУ — частотная проверка Φ_a. Происходит многократное считывание кода 0101 ... 10 из группы ячеек, имеющих общий код x:

0000—0037
0040—0077
. . . . . .
1740—1777

Четвертый тест ОЗУ. Память заполняется кодами

00 ... 001
00 ... 010
. . . . .
10 ... 000
00 ... 001
. . . . .

Затем происходит поочередное их считывание и сравнение с кодами, которые формируются по тому же правилу, что и записанные. После этого тест повторяется с обратными кодами.

ПЗУ проверяется путем вызова всего его содержимого в ОЗУ и сравнения всех кодов с введенными в машину с перфолент.

Тесты арифметического устройства (АУ)

В состав арифметического устройства машины «Киев» входят:

1) приемные регистры P_1 и P_2, используемые для хранения множимого и множителя при умножении, делителя и частного — при делении. В них же осуществляются сдвиги кодов при операциях умножения, деления, нормализации, сдвига. Регистр P_1 используется еще как счетчик окончания деления и т. д.;

2) сумматор, в котором выполняется операция неравнозначности и переносы из разряда в разряд;

3) регистр операций, дешифратор операций, блоки управления операциями и временного управления.

Арифметическое устройство проверяется двумя программами. Первая программа контролирует выполнение всех арифметических и логических операций. Каждая операция выполняется над некоторой группой кодов и результаты сравниваются с заранее известными. При этом коды подобраны таким образом, что в каждый из разрядов двух регистров и сумматора поступают все возможные сочетания входов.

Некоторые операции дают возможность частично локализовать неисправные цепи сумматора. Операция логического сложения проверяет первый регистр и установочный вход триггера каждого разряда сумматора; операция логического умножения — второй регистр и счетные входы триггеров сумматора при начальном нулевом состоянии; операция «неравнозначно» — работу сумматора при отсутствии переносов, а цепи переносов проверяются операцией сложения и циклического сложения; сдвиги в P_1 и P_2 проверяются операциями умножения, деления, нормализации и сдвига.

После проверки арифметического устройства с помощью первой программы остаются некоторые сомнения относительно его исправности, так как могут существовать неисправности, необнаруживаемые этим тестом. Более полная проверка осуществляется тестом умножения и деления псевдослучайных чисел. Эта программа

предназначена для обнаружения неисправности арифметического устройства и менее удобна с точки зрения локализации неисправных цепей.

Умножение проверяется следующим образом. Вырабатывается последовательность псевдослучайных чисел. Каждое новое число умножается на предыдущее два раза с переменой местами сомножителей. Поскольку множимое и множитель играют существенно различную роль при выполнении операции умножения, неисправность сумматора или одного из регистров должна вызвать несовпадение результатов умножений. Умножение в этой программе производится с округлением.

Деление проверяется также с помощью последовательности псевдослучайных чисел. Операция выполняется над двумя очередными числами (меньшее делится на большее) два раза. При этом второй раз деление производится специальной подпрограммой по итерационной формуле, которая должна давать частное с ошибкой, не превосходящей $\varepsilon = 2^{-36}$. Результаты сравниваются с точностью до ε.

Оба теста пробиты на одной ленте. Первый тест разбит на три куска: проверка логических операций, операции сложения и остальных операций. Набирая различные знаки в наборной ячейке 3000, тест можно запускать по кускам или все куски подряд:

Тесты	I	II	III	IV	Все
Знак НК 3000	+	+	+	+	—
Начало	0043	0110	0155	0214	0042

Выполнение программы повторяется до остановки машины с пульта управления или до получения неверного результата. Если неверный результат будет получен на одном из первых трех участков теста, то произойдет останов с выдачей на пульт управления (сигнализация сумматора) результата несравнения полученного результата с заранее известным верным результатом.

Если же не совпадут результаты двух умножений или двух делений одних и тех же чисел в IV тесте, то машина напечатает оба сомножителя (или делимое и делитель) и оба результата и остановится. В команде останова на регистре команд K в III адресе будет код выполняемой операции (умножения 11 или деления 12).

Тест печати

Цифропечатающее устройство проверяется путем печати в восьмеричной и десятичной системах четырех групп чисел (см. приложение 2). Порядок и количество печатей регулируется содер-

жимым ячейки 3000 следующим образом. При 3000 = +0 многократно печатается первая группа кодов. При переключении 3000 на —0 начинает печататься вторая группа. Затем +0 дает третью и —0 — четвертую группы.

Следует помнить, что при переходе на печать третьей группы следует включить десятичную печать. Первые две группы печатаются в восьмеричной системе. Ввод программы осуществляется с пульта управления командой

21	0001	0655	0001

Тест магнитного барабана (МБ)

Тест магнитного барабана проверяет правильность записи кодов на МБ и считывания их с МБ в ОЗУ.

Тест работает в следующем порядке:

1) запись кодов в первую половину ячеек ОЗУ;
2) запись кодов на первые места барабана;
3) считывание с МБ на вторую половину ячеек ОЗУ;
4) поодиночное сравнение кодов, записываемых и считываемых с МБ;
5) запись на следующий участок МБ;
6) переход к п. 3.

Участки проверки МБ могут быть различной длины, но не более половины ОЗУ, т. е. 500 кодов. Длина участка задается в I адресе ячейки НК 3000 (n). Во II адресе НК 3001 задается число участков разбиения барабана. Тест построен для проверки двух барабанов. Номер барабана задается в ячейке 3002

		0k00	b

Здесь $k = 1,2$; b — начальное место на барабане.

В случае несовпадения кодов машина останавливается.

Для образования кодов записи предусмотрено два случая:

а) код записи набирается вручную на НК 3007 и на все места барабана записывается один и тот же код;

б) код записи образуется по программе псевдокодов и в каждую ячейку ОЗУ для записи засылается новый код.

ГЛАВА V

ПРОГРАММИРУЮЩИЕ ПРОГРАММЫ ДЛЯ МАШИНЫ «КИЕВ»

ПОСТАНОВКА ЗАДАЧИ

В этой главе приведено описание двух программирующих программ (ПП) для машины «Киев» — ПП-АК и ПП-2.

ПП-2 является первой из разрабатываемых для машины «Киев» систем автоматического программирования, в основном рассчитанной на программирование арифметических задач. Язык ПП-2 учитывает ряд особенностей кода машины «Киев» и не является универсальным, как адресный или Алгол [13], т. е. с него предусматривается перевод только для машины «Киев». В этом недостаток ПП-2.

ПП-АК в качестве входного языка использует адресный язык, на стиль которого наложены весьма незначительные ограничения [19]. Применение адресного языка в качестве входного для всех других машин ВЦ АН УССР объясняет широкий успех внедрения этой ПП в практику.

При разработке обоих вариантов ПП учитывалось основное требование: ПП должны быть практически приемлемыми, удобными в эксплуатации. В результате обе ПП занимают промежуточное положение между большими программирующими программами и составляющими программами. Ко вторым они приближаются благодаря своей краткости и высокой скорости работы. В некотором смысле обе ПП могут рассматриваться как программы ввода, так как перевод исходной информации в код машины «Киев» они осуществляют *одновременно с вводом*, практически не задерживая последний. Но использование ПП позволяет ввести программу с любого места памяти, добиться экономии рабочих ячеек, записать программу на барабан, проконтролировать ввод и магнитную запись.

Заметим также, что с помощью ПП-АК и ПП-2 можно получать программы, предназначенные для выполнения вычислений в режиме плавающей запятой. С этой целью можно использовать один из следующих способов.

Выполнение вычислений в режиме плавающей запятой можно учитывать непосредственно при составлении адресной программы, по которой с помощью ПП будут составлены программы в режиме плавающей запятой.

Однако включение режима плавающей запятой непосредственно в адресные программы зачастую представляет собой весьма трудоемкую работу, в связи с чем можно использовать приведенный в гл. IV метод моделирования режима плавающей запятой. Поскольку использование этого метода накладывает некоторые ограничения на набор допустимых элементарных операций, а также приводит к замедлению вычислений примерно в 10 раз, его можно применять на отдельных участках программ; другие же участки могут выполняться в режиме фиксированной запятой или в плавающих масштабах.

Особым является вопрос об использовании в записи формул скобок. Как показал польский математик Ян Лукасевич, порядок выполнения операций может быть задан однозначно без применения скобок, для чего символ операции необходимо вынести вперед, а аргументы записать за ним. Например, выражение

$$(a + b) \times c = f$$

в записи Лукасевича имеет вид

$$= \times + abcf,$$

а выражение

$$\frac{\ln (a + b)}{a - b} \times c = f$$

имеет вид

$$= \times : \ln + ab - abcf.$$

Запись Лукасевича не менее удобна, чем общепринятая запись формул с закрывающими и открывающими скобками различной конфигурации. Но особенно она удобна при записи операторов преобразования для автоматического программирования. Здесь речь идет не только об исключении одного из классов исходной информации — скобок, которое позволяет упростить кодирование. Бесскобочная запись формул обладает одним весьма интересным свойством: знак первого справа символа операции относится к выполнимой операции. Таким образом, бесскобочная запись формул позволяет существенно упростить отбор информации для синтеза команд.

Переход от общепринятой записи к бесскобочной весьма прост, кроме того, может быть легко совершен самой машиной, что и осуществлено в приведенных здесь системах автоматизации. Непосредственно программирующими программами ПП-АК и ПП-2 запись Лукасевича используется в несколько измененном виде — справа налево. Наши примеры перепишутся в следующем виде:

$$fcba + \times =$$

$$fcba - ba + \ln : \times =$$

Запись в этом виде позволяет легко осуществлять отбор информации для синтеза команд. Так, в первом примере первой выполнимой операцией является сложение $a + b$ или $ba +$. Чтобы установить это, алгоритм должен для двухместной операции обозревать по три, а для одноместной — по два элемента слева направо. Во втором примере первой выполнимой операцией будет $a — b$ или $ba —$, но для ее отыскания достаточно обозревать и анализировать по одному элементу информации (ЭИ) и найти символ операции. В силу правил записи два ЭИ слева (один для одноместной операции) будут аргументами.

При программировании формул, заданных в скобочной записи, предварительно подключается алгоритм перевода скобочной записи в бесскобочную (см. [21]) с одновременной проверкой (формальной) этих записей [22].

ПРОГРАММИРУЮЩАЯ ПРОГРАММА ДЛЯ МАШИНЫ «КИЕВ» ИНФОРМАЦИЕЙ ДЛЯ КОТОРОЙ СЛУЖИТ АДРЕСНЫЙ АЛГОРИТМ (ПП-АК) *

Преимущество ПП-АК в сравнении с известными нам программирующими программами состоит прежде всего в универсальности входного алгоритмического языка. Эта особенность входного языка при возрастающем числе машин различных типов делает ПП-АК весьма перспективной.

Стремление к созданию ПП, быстродействующей и удобной для практического использования, привело к необходимости наложения некоторых ограничений. В частности, в ПП для машины «Киев» до сих пор не автоматизировано обращение к внешней памяти.

Несмотря на то что ПП-АК (без блока перевода формул в бесскобочную запись) состоит всего лишь из 576 (в десятичной системе) команд, она в качестве входного использует универсальный адресный язык с весьма незначительными ограничениями на стиль языка, включает блок экономии формул, экономию рабочих ячеек и блок вычисления предикатных функций.

Рабочая программа, составленная ПП-АК, может выводиться на перфокарты или на бумажную ленту.

Среднее быстродействие ПП-АК составляет 100 команд в минуту (с учетом ввода и вывода). Входная информация поступает зонами. Предусмотрен автоматический контроль вместимости рабочей программы на рабочем поле.

Оптимальный (наиболее приемлемый для ПП-АК) размер рабочих программ до 600 восьмеричных кодов; для создания рабочих

---

* ПП-АК разработана в ВЦ АН УССР Л. П. Быстровой под руководством Е. Л. Ющенко [19].

программ бо́льшего объема необходимо разделить адресный алгоритм на замкнутые блоки, т. е. на части с одним входом и одним выходом.

ВХОДНОЙ ЯЗЫК ПП-АК

Информацией для настоящего варианта программирующей программы служит адресная программа задачи. В соответствии с этим положением допустимыми в информации являются следующие адресные формулы: засылки (арифметические операторы); вхождения программ (операторы обращения к подпрограммам); предикатные (логические операторы) и метки безусловного перехода; останова; циклирования; печати. В отдельных случаях допускается задание информации в виде готовых приказов, которые программирующей программой целиком переносятся в рабочую программу. Такие элементы информации назовем нестандартными операторами.

Формулы засылки представляют собой часть адресной программы, осуществляющей однократное преобразование информации по некоторой последовательности адресных формул с записью результата по адресу, определяемому в общем случае как значение некоторой адресной функции. Режим фиксации запятой должен учитываться при составлении адресного алгоритма.

Пересылка содержимого адреса a в ячейку b записывается в виде $'a \Longrightarrow' b$ (обычно в адресном языке принято писать $'a \Longrightarrow b$). Таким образом, в адресном стиле ПП-АК ранг формулы, расположенной справа от знака засылки $\Longrightarrow$, повышается на единицу по сравнению с обычно употребляемым способом записи в адресном языке (можно считать, что вместо знака $\Longrightarrow$ используется знак $\Longrightarrow'$). Адреса нулевого ранга употребляются только как добавки к адресам и как счетчики в операторах циклирования. Употребление адресов нулевого ранга в других случаях запрещается. Адрес нулевого ранга должен обязательно быть вторым аргументом операции.

Часть адресной программы называется *подпрограммой*, если она имеет одну начальную входную метку $К_в$ и выходную метку Я (подпрограммы с n входами рассматриваются как n подпрограмм). Формулами вхождения подпрограмм называются адресные формулы, обозначающие реализацию подпрограммы с входом $К_в$ в данном месте адресной программы; такие формулы обозначаются через $ПК_в$.

Формулы вхождения подпрограмм кодируются на внутренней памяти как одноместные операции.

Вся входная информация для подпрограммы (информация об обрабатываемых и получаемых массивах, метках используемых подпрограмм, выходной метке и т. д.) задается в виде $'0002 =$

последовательности. Для отдельных подпрограмм с небольшим числом параметров требуется осуществление пересылок параметров непосредственно перед формулой вхождения в рабочие ячейки подпрограмм (0004, 0003, ...), кроме 0002. Параметр, который должен быть заслан в ячейку 0002, записывается как аргумент одноместной операции, кодирующей формулу вхождения данной подпрограммы.

Пример. Счет $'a + \ln 'x$ с масштабом N для ln в адресной программе должен быть записан в виде $N \Longrightarrow 0004$

$$'a + \ln 'x \Longrightarrow 'c.$$

Каждая подпрограмма должна заканчиваться знаком Я.

Все строки программы, к которым *принудительно* передается управление (и только такие), в порядке их размещения в исходной информации *помечаются метками*, представляющими собой последовательность натуральных чисел (нумеруются начиная с 1), и после каждого из них ставится знак «начало оператора» (обозначается i..., где i — номер оператора). Эти же метки используются при записи значений соответствующих предикатных формул.

Кроме операций, имеющихся в наборе машины «Киев», допускаются следующие моделируемые ПП-АК операции:

1. Не больше «$\leqslant$»:

$$a \leqslant b \Longrightarrow 'c.$$

Операция означает

$$'c = \begin{cases} \text{ложь при} & a > b; \\ \text{истина при} & a \leqslant b. \end{cases}$$

2. Не больше по модулю «$|\leqslant|$»:

$$|a| \leqslant |b| \Longrightarrow 'c.$$

Операция означает

$$'c = \begin{cases} \text{ложь при} & |a| > |b|; \\ \text{истина при} & |a| \leqslant |b|. \end{cases}$$

3. Равно «=»:

$$(a = b) \Longrightarrow 'c.$$

Операция означает

$$'c = \begin{cases} \text{ложь при} & a \neq b; \\ \text{истина при} & a = b. \end{cases}$$

4. Равно по модулю:

$$|a| = |b| \Longrightarrow c.$$

Операция означает

$$'c = \begin{cases} \text{ложь при} & |a| \neq |b|; \\ \text{истина при} & |a| = |b|. \end{cases}$$

Здесь a, b — адреса нулевого, первого или второго рангов; истина означает код -0, ложь — код $+0$.

5. Инвертирование: *не f*, где f — любая адресная формула. Логические операции *и*, *или*, *не* обозначаются соответствующими словами.

Формула останова обозначается словом «останов».

Допускаются *формулы циклирования* следующего вида:

$$Цi\ \{a_0\ (\Delta a)\ a_n \Longrightarrow' \pi\}\ K_1,$$

где a_0, Δa, a_n — адреса нулевого или первого рангов.

Формула *Ц* ставится в начале своей области действия (в начале оператора), заменяет одновременно знак метки i с троеточием и предписывает участок программы по строку K_1 исключительно (!) повторить, присваивая последовательно адресу π значения от a_0 до a_n с шагом $\Delta\alpha$.

Таким образом, можно задавать как численные значения параметров цикла, так и адреса, их содержащие; при этом численные значения могут быть только натуральными восьмеричными числами, а адреса — содержать любые величины. Необходимо помнить, что:

а) a_n задается в виде $a_n = \bar{a}_n - \Delta a$, где $\bar{a}_n$ — последнее значение параметра $'\pi$, для которого должен работать цикл;

б) если нижние границы внутреннего и внешнего циклов совпадают, то в конце цикла ставится соответственно две метки.

Пример. Пусть оператор Σ_1 используется в двойном цикле. Соответствующая адресная программа записывается так:

$$\begin{array}{c} Цi\ (\ldots)\ i+3 \\ Цi+1\ (\ldots)\ i+2 \\ \Sigma_1 \\ i+2\ \ldots \\ i+3\ \ldots \end{array}$$

Операция вывода (печати) содержимого адреса $'a$ — восьмеричного или десятичного кода — записывается в виде

$$Печ_8\,'a \text{ или } Печ_{10}\,'a.$$

Для вывода последовательности кодов применяется формула циклирования.

Пример. Напечатать числа $'(\alpha+1), \ldots, '(\alpha+n)$.

$$\begin{array}{c} ЦК\ \{1\ (1)\ n \Longrightarrow' \varphi\}\ K+1 \\ Печ_{10}\,'('\varphi+\alpha) \\ K+1\ \ldots \end{array}$$

Распределение памяти

Память машины распределяется следующим образом: исходные данные для задачи и фиксированные адреса размещаются в конце памяти машины от ячейки 1777 и выше; рабочая про-

грамма (РП), составляемая ПП-АК, располагается от ячейки 0020 и далее вниз. Рабочие ячейки выбирает ПП-АК, располагая их между программой и исходными данными.

Адресная программа представляется для кодировки в условных адресах (в буквенных выражениях).

После обработки информации ПП-АК выдает рабочую программу задачи и таблицу распределения меченых строк в памяти машины. ПП-АК без блока перевода формул в бесскобочную запись занимает 1100 ячеек ОЗУ (восьмеричных), т. е. 576_{10} адресов. Скорость работы, включая ввод и вывод, 100 команд в минуту.

Кодирование информации для ПП-АК

Перед основной информацией о задаче на четвертую зону перфоленты в указанном порядке пробиваются следующие данные о ее размерах:

1) максимальное количество меток (с троеточием) в задаче—n_3;

2) количество ячеек, занятых числовой информацией задачи,— n_4;

3) длина максимального участка информации — n_5;

4) длина первого участка информации и номер зоны на перфоленте, на которой закодирован этот участок.

Вся информация о программе делится на участки, не превышающие по длине n_5 элементов (n_5 фиксировано для данной задачи). Деление это произвольно, но при этом сохраняется целостность отдельных формул и нестандартных операторов. Каждый участок пробивается на отдельной зоне перфоленты; его информация заканчивается знаком «конец вводимого участка», после чего помещается указание о длине следующего участка в виде

		m	k

Здесь m — длина участка; k — номер зоны перфоленты, на которой размещен этот участок.

Каждый отдельный элемент информации помещается в одну ячейку ОЗУ. Первые пять разрядов ячейки (разряды кода операции) отведены под признаки, классифицирующие элемент.

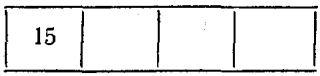
Ниже приводятся правила кодирования различных классов элементов информации.

1. Перед нестандартным оператором ставится его признак — ведущий элемент

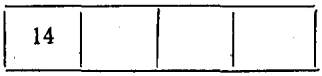
36		n	

Здесь n — число элементов нестандартного оператора. После этого кода помещаются n элементов информации, которые целиком должны быть занесены в рабочую программу.

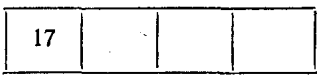
2. Начало оператора



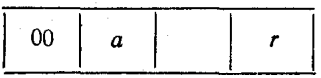
3. Конец информации о задаче



4. Конец вводимого участка



Кодирование чисел. Информация о числе должна содержать признак числа, истинный адрес числа и ранг его адреса. Признак числа кодируется на разрядах кода операции команды, адрес a — на разрядах I адреса, ранг r адреса — на разрядах III адреса:



Ранг числа может быть произвольным от 0 и выше, причем адреса нулевого ранга рассматриваются ПП-АК как соответствующие количества единиц II адреса. Применение адресов нулевого ранга допускается только для изменения адресов и для работы счетчиков.

Кодирование операций. ПП-АК различает операции одноместные, двухместные и штрих-операцию. Операция переноса по адресу отнесена к двухместным операциям.

Помимо операций элементарных, имеющихся в наборе машины «Киев» ($+$, $-$, $|-|$, $\times$, $\boxtimes$, $:$, $Л\rightarrow$, $\vee$, $\wedge$, $+Ц$, $СлК$, $\cong$), допустимы некоторые обобщенные операции, моделируемые программирующей программой путем составления программ их реализации или обращения к стандартной подпрограмме, если она имеется в ОЗУ. Среди операций, моделируемых ПП-АК, отметим $\mathrm{mod}\, a$, a^n, а также некоторые специальные операции, часто встречающиеся в программах. Этим самым как бы расширяется набор машинных операций. Как было указано, ПП-АК моделирует также некоторые логические операции.

Моделируемые операции кодируются так:

$\leqslant$	02	1350		r
$\|=\|$	02	1366		r
$=$	02	1404		r
$\|\leqslant\|$	02	1410		r
Отрицание	22	1620		r
Степень	22	1626		r
Модуль	22	1635		r

Здесь r — ранг результата операции.

Для кодирования формул в скобочной записи введены следующие коды:

код открывающей скобки

04			

код закрывающей скобки

05			

Формулы отделяются друг от друга разделительным знаком, код которого имеет вид

06			

Операторы останова, ввода и вывода кодируются как нестандартные операторы.

Элементарные операции кодируются в виде

01	00k		r

Здесь k — восьмеричный код операции; r — ранг результата.

Код операции $\Longrightarrow$ имеет вид

03			

Вычисление функций по стандартной подпрограмме кодируется как одноместная операция, код которой соответствует адресу первого приказа подпрограммы в ЗУ или метке подпрограммы

21	k_1		r

Здесь k_1 — вход в подпрограмму; r — ранг результата операции.

Штрих-операция кодируется в виде:

а) ранга адреса, если эта операция стоит непосредственно перед адресом;

б) ранга результата, если она стоит перед знаком операции.

Кодирование предикатных формул. Реализацию произвольной предикатной формулы

$$P\{F(x_1, \ldots, x_s)\} K_1 \downarrow K_2 \qquad (21)$$

можно свести к вычислению значений соответствующей ей предикатной функции $F(x_1, \ldots, x_s)$ и к реализации предикатной формулы вида

$$P\{{}^r a \leqslant 0\} K_1 \downarrow K_2, \qquad (22)$$

где ${}^r a$ — адрес r-го ранга, по которому содержится вычисленное значение предикатной функции.

Предикатные формулы вида (22) называются элементарными. Предикатная формула общего вида (21) кодируется в виде

$$F(x_1, \ldots, x_s) P K_1 \downarrow K_2.$$

Вычисление предикатной функции F программируется арифметическим оператором программирующей программы. Для кодирования элементарных предикатных функций отводится две ячейки, в которых данные размещаются следующим образом:

16	a		r
		K_2	K_1

Если предикатная формула (22) стоит в информации непосредственно после вычисления предикатной функции, значение которой получается в рабочей ячейке, то вместо a и r при кодировке предиката ставятся нули.

Если K_1 (или K_2) является меткой оператора, стоящего непосредственно вслед за предикатной формулой, то этот оператор можно специально не отмечать, а при кодировании вместо K_1 (или K_2 соответственно) ставить нуль.

Информация о циклах кодируется следующим образом: формула циклирования

$$Ц({}^{r_1}a_0 ({}^{r_2}\Delta a_0)\, {}^{r_3}a_{\text{конечн}} \Longrightarrow {}'\pi) K$$

кодируется в двух ячейках ОЗУ в виде

$K+1$	15	π	$a_{\text{конечн}}$	K_1
$K+2$	r_3	a_0	Δa_0	r_1+r_2

Здесь r_1, r_2, r_3 равны 0 или 1. При $r_1=1$ во 2-м разряде ячейки $K+1$ ставится 1, при $r_2=1$ в 1-м разряде ячейки $K+2$ ставится 1.

Алгоритмы программирования

Программирующая программа состоит из следующих блоков: *ведущий алгоритм; арифметический; обработки* нестандартных операторов; запоминания адресов операторов; присвоения истинных адресов; обработки формул циклирования.

Ведущий алгоритм программирующей программы ПП-АК вводит в ОЗУ очередной участок информации о программе, производит анализ очередного ведущего элемента информации и в соответствии с принадлежностью его к тому или иному классу передает управление соответствующему блоку ПП-АК.

Блок программирования арифметических операторов производит программирование формул с одновременной экономией формул и рабочих ячеек.

Арифметический блок передает управление ведущему оператору, сигналом для чего служит появление при обозревании несвойственного для него элемента, т. е. элемента, не являющегося ни величиной, ни операцией.

Алгоритм арифметического блока в основном копирует алгоритмы, которыми пользуется человек при ручном программировании. Экономия рабочих ячеек производится в пределах данного оператора; экономия команд — в пределах данной формулы. Расширение алгоритма до экономии команд за пределами одной формулы представляется весьма затруднительной задачей ввиду наличия в информации адресов высших рангов.

Порядок работы алгоритма арифметического блока следующий:

Поиск (слева направо) *первой выполнимой операции.* Информация просматривается до нахождения первой выполнимой операции.

Построение команды (или нескольких), реализующей найденную операцию с засылкой результата в очередную свободную рабочую ячейку.

Засылка адреса результата программируемой операции с признаком рабочей ячейки в исходную информацию на место кода этой операции.

Сжатие информации: информация от начала оператора и до ячейки, хранящей I адрес, над которым производилась операция, подтягивается вплотную к ячейке, в которой теперь хранится результат запрограммированной операции.

Пример. Пусть программируется арифметический оператор

$$\frac{('a + 'b)}{'c} \times \sin 'a \Longrightarrow '\!d.$$

В бесскобочной записи формула будет иметь вид

$$dcba + : a \sin \times \Longrightarrow).$$

Пусть информация о формуле занимает ячейки от α до $\alpha + 9$

...	d	c	b	a	+	:	a	sin	×	$\Longrightarrow$	...	...
	α	$\alpha+1$	$\alpha+2$	$\alpha+3$	$\alpha+4$	$\alpha+5$	$\alpha+6$	$\alpha+7$	$\alpha+8$	$\alpha+9$	$\alpha+10$	$\alpha+11$

Первой выполнимой операцией будет сложение **a + b**. После программирования этой операции и занесения результата в некоторую рабочую ячейку *r*, в результате сжатия информация примет вид

...	...	...	d	c	r	:	a	sin	×	$\Longrightarrow$	...
	α	$\alpha+1$	$\alpha+2$	$\alpha+3$	$\alpha+4$	$\alpha+5$	$\alpha+6$	$\alpha+7$	$\alpha+8$	$\alpha+9$	

Началом оператора становится ячейка $\alpha + 2$.

Экономия команд на уровне исходной информации. Если в пределах данной формулы имеется выполнимая операция, информация о которой совпадает с информацией о найденной выполнимой операции, то адресу результата первой из операций присваивается признак стандартной ячейки, который с признаком рабочей ячейки заносится на место кода операции аналогичной формулы, после чего информация вновь сжимается. Поиск следующей аналогичной операции продолжается до тех пор, пока не встретится знак засылки $\Longrightarrow$, означающий конец формулы.

Таким образом, если в формуле имеется несколько аналогичных операций, то после программирования первой такой операции вместо информации о каждой из них будет содержаться информация об адресе результата, при этом для всех формул, кроме последней, адрес результата будет снабжен признаком стандартной ячейки, а для последней — признаком рабочей ячейки.

Поиск следующей выполнимой операции. Поиск продолжается до тех пор, пока вся информация об арифметическом операторе не будет переработана в программу задачи.

Если адреса, входящие в рассматриваемую выполнимую операцию, являются адресами высшего ранга, перед программированием операции работает специальная подпрограмма понижения

ранга адреса. Эта подпрограмма составляет приказ понижения ранга адреса, если a — адрес второго ранга, вида

Φ	a		—

и, если ранг адреса $a > 2$, — приказы

Φ	a	—	—
Φ	4000	—	—
. . . . .			
Φ	4000		

$(n-2)$ раза

Перед записью этих приказов в программу производится проверка, не содержит ли уже A-регистр модификации нужного адреса и не менялось ли содержимое данного адреса высшего ранга с момента вызова его на A-регистр, если вызов имел место (т. е. не содержится ли данный адрес высшего ранга в III адресе приказов рабочей программы, следующих за приказом вызова его на A-регистр). В благоприятном случае экономятся приказы Φ.

Если некоторая операция K производится над адресами, из которых только один является адресом второго ранга, то составляются приказы:

Φ	b	—	—
K	4000	a	r

. . . . . .

Если оба адреса второго ранга, то операция над ними программируется в виде

Φ	b	—	r
Φ	a	—	—
K	r	4000	r

Аналогично программируются приказы для рангов выше второго. Операции с адресами нулевого ранга программируются особо.

Если перед операцией в исходной информации стоит код засылки по адресу $\Longrightarrow$, то это означает, что производится изменение содержимого адреса по первому рангу на n единиц. В этом случае программирующая программа составляет приказ

+	a	r_n	b

Здесь a — изменяемый адрес; r_n — адрес, в который ПП-АК помещает константу сдвига n, предварительно проверив, не содержится ли она среди чисел ПЗУ или среди составленных ранее констант.

Если результат операции является адресом для другой операции, программирование операции с нулевым рангом должно производиться непосредственно перед программированием операции, для которой она формирует адрес. При этом адрес нулевого ранга помещается с признаком модификации в приказ, а адрес высшего ранга — на A-регистр.

Экономия рабочих ячеек. Для рабочих ячеек программирующей программой отводится специальный массив, причем определяется только его нижняя граница (верхняя граница тоже может быть определена, так как максимальное количество рабочих ячеек равно количеству одновременно выполняемых операций, входящих в состав одной формулы).

Каждая рабочая ячейка, в которую помещается результат, отмечается в массиве засылкой в нее специального кода (—0). Если же ячейка с признаком *раб* затем участвует в выполнимой операции, то согласно алгоритму экономии команд она с этого момента считается свободной, и в нее засылается код $+0$. Таким образом, все свободные в данный момент рабочие ячейки содержат в себе $+0$, занятые —0. Как только потребуется поместить результат операции в рабочую ячейку, ПП-АК отыскивает в массиве рабочих ячеек первую свободную снизу, т. е. первую ячейку снизу, содержащую $+0$.

Присвоение истинных адресов. При распределении памяти машины необходимо учитывать, что построение алгоритмов ПП-АК требует, чтобы исходные данные для задачи помещались в конце памяти; программа задачи будет записываться с указанной по наборному коду 3007 ячейки ОЗУ:

3007				
			R	

ПП-АК получает информацию об истинных адресах чисел и сама выбирает рабочие ячейки, поэтому все приказы, составляемые ею, записываются в истинных адресах. Приказы, реализующие условные переходы и операции над кодами, вместо адресов команд содержат метки строк — номера операторов, которым передается

управление. Исключение составляют приказы обращения к стандартным подпрограммам, которые записываются также в истинных адресах. В процессе программирования задачи ПП-АК составляет таблицу распределения операторов в памяти, в которой i-я строка (ячейка) содержит истинный адрес первого приказа i-го оператора задачи.

После программирования начинает работать специальный блок ПП-АК, так называемый блок присвоения истинных адресов. Во всех приказах, содержащих номера операторов, эти номера заменяются адресами первых приказов операторов согласно таблице размещения операторов.

ПРИМЕРЫ ПРОГРАММ, СОСТАВЛЕННЫХ ПП-АК

I. Рассмотрим алгоритм метода сопряженных градиентов после первой трансформации Гаусса для решения систем линейных алгебраических уравнений

Описание метода

Пусть A — неособенная матрица. Из системы $AX = F$ первой трансформацией Гаусса получается система

$$A'AX = A'F.$$

Применение метода сопряженных градиентов к этой системе дает:

$$X = X_0 + \sum_{i=1}^{n} a_i s_i;$$

$$a_1 = \frac{(\bar{r}_{i-1}, \bar{r}_{i-1})}{(s_i, A'As_i)};$$

$$s_1 = \bar{r}_0;$$

$$\bar{r}_1 = \bar{r}_{i-1} - a_i A'As;$$

$$s_{i+1} = \bar{r}_i + b_i s_i;$$

$$b_i = \frac{(\bar{r}_i, \bar{r}_i)}{(\bar{r}_{i-1}, \bar{r}_{i-1})}.$$

Здесь $\bar{r}_i$ — невязка преобразованной системы. Ясно, что $\bar{r}_i = A'r_i$, где r_i — невязка исходной системы. Принимая это во внимание и преобразуя скалярное произведение, приходим к расчетным формулам:

$$X = X_0 + \sum_{i=1}^{n} a_i s_i;$$

$$a_i = \frac{(A'r_{i-1}, A'r_{i-1})}{(As_i, As_i)};$$

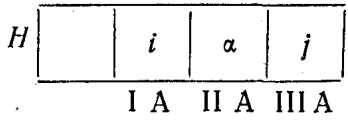
$$s_1 = A'r_0;$$

$$r_i = r_{i-1} - a_i A s_i;$$
$$s_{i-1} = A' r_i + b_i s_i;$$
$$b = \frac{(A' r_i,\ A' r_i)}{(A' r_{i-1},\ A' r_{i-1})}.$$

Адресная программа

Вектор s_1 задается в виде A-последовательности, вектор As_i — B-последовательности, вектор r_1 — C-последовательности. Решение системы X будем получать в виде D-последовательности.

В связи с тем, что исходная матрица может содержать много нулей, информация задается в виде массива чисел и шкал



Здесь i — номер строки; j — номер столбца; α — адрес ненулевого элемента.

Количество шкал равно количеству ненулевых элементов матрицы. Двум равным элементам матрицы соответствует одно число в массиве чисел и две шкалы с равными вторыми адресами.

Массив шкал задается в виде H-последовательности, n_1 — размерность этого массива. Начало массива свободных членов системы задается, все другие последовательности получаются по программе.

Программа включает в себя три подпрограммы: 1) скалярное произведение векторов; 2) произведение транспонированной матрицы на вектор; 3) произведение матрицы на вектор.

После $\left[\frac{n}{2}\right]$ циклов следует проверить, являются ли элементы C-последовательности меньше заданного ε. Если какой-либо из элементов не меньше ε, следует выполнить еще $\left[\frac{n}{2}\right]$ циклов. Если все элементы C-последовательности меньше ε (по модулю), осуществляется контроль, т. е. пересчитывается C-последовательность. Если элементы вновь вычисленной C-последовательности меньше ε, выдается D-последовательность — решение системы; в противном случае счет начинается сначала с вновь полученной C-последовательностью.

Адресный алгоритм в стиле ПП-АК

```
1 ...                                    11 4001 0006 0006
    НС  —  0012  —                       01 0006 0003 0003
    01  0000 0000 0003                   01 0002 3011 0002
    34  0002 0000 0005                   01 0004 3011 0004
    34  0004 0000 0006                   02 0005 3011 0005
    34  0002 0000 0000                   31 0005      3146
```

2 . . .
$'3012 \Longrightarrow {}' j$

3 . . .
$'3010 \Longrightarrow {}' i$
$'3011 \Longrightarrow {}' h$
$0 \Longrightarrow {}' s$

4 . . .
$' ('H + 'h) \wedge '3024 \Longrightarrow {}' r_1$
$P\,\{'r_1 = 'j\}\ 5 \downarrow 0$
$'h + '3011 \Longrightarrow {}' h$
$P\,\{'h \leqslant n_1\}\ 4 \downarrow 7$

5 . . .
$' ('H + 'h) \wedge '3022 \Longrightarrow {}' r_1$
$P\,\{'r_1 = 'i\}\ 0 \downarrow 6$
$'i \times '3040 \Longrightarrow {}' r_1$
$^2('H + 'h) \times ' ('Cr + 'r_1) + 's \Longrightarrow {}' s$
$'h + '3011 \Longrightarrow {}' h$

6 . . .
$'i + '3010 \Longrightarrow {}' i$
$P\,\{'h \leqslant n_1\}\ 0 \downarrow 7$
$P\,\{'i \leqslant 'r_3\}\ 4 \downarrow 0$

7 . . .
$'j : '3040 \Longrightarrow {}' r_1$
$'s \Longrightarrow {}' ('Ar + 'r_1)$
$'j + '3012 \Longrightarrow {}' j$
$P\,\{'j \leqslant 'r_4\}\ 3 \downarrow$ Я

10 . . .
$'3010 \Longrightarrow {}' i$
$'3011 \Longrightarrow {}' h$

11 . . .
$'3012 \Longrightarrow {}' j$
$0 \Longrightarrow {}' s$

12 . . .
$' ('H + 'h) \wedge '3024 \Longrightarrow {}' r_1$
$P\,\{'r_1 = 'j\}\ 0 \downarrow 13$
$'j : '3040 \Longrightarrow {}' r_1$
$^2('H + 'h) \times ' ('Ar + 'r_1) + 's \Longrightarrow {}' s$
$'h + '3011 \Longrightarrow {}' h$

13 . . .
$'j + '3012 \Longrightarrow {}' j$
$P\,\{'j \leqslant 'r_4\}\ 12 \downarrow 0$
$'i \times '3040 \Longrightarrow {}' r_1$
$'s \Longrightarrow {}' ('Br + 'r_1)$
$'i + '3010 \Longrightarrow {}' i$
$P\,\{'i \leqslant 'r_3\}\ 11 \downarrow$ Я

Начало программы ...

$n : '3040 \Longrightarrow 'r_3$
$n \times '3040 \Longrightarrow 'r_4$
$n - '3011 \Longrightarrow 'r_2$
$n + '3011 \Longrightarrow 'r_1$
$'C + 'r_1 \Longrightarrow 'R$
$'R + 'r_1 \Longrightarrow 'A$
$'A + 'r_1 \Longrightarrow 'B$
$'B + 'r_1 \Longrightarrow 'D$

14 ...

$'C \Longrightarrow 'Cr$
$'A \Longrightarrow 'Ar$
$\pi 2$
$n \Longrightarrow {}^2A$
$'A + '3011 \Longrightarrow '0004$
$\pi 1\,(A) \Longrightarrow '\beta_1$

15 ...

$Ц16 \left\{0\,(1) \left(\left[\frac{n}{2}\right] - '3011\right) \Longrightarrow 's_1\right\} 25$
$Ц17\,\{1\,(1)\,'r_2 \Longrightarrow 'k\}\,20$
$0 \Longrightarrow '('D + 'k)$

20 ...

$'A \Longrightarrow 'Ar$
$'B \Longrightarrow 'Br$
$\pi 10$
$n \Longrightarrow {}^2B$
$'B + '3011 \Longrightarrow '0004$
$\pi 1\,(B) \Longrightarrow '\alpha$
$Ц21\,\{1\,(1)\,'r_2 \Longrightarrow 'k\}\,22$
$'('C + 'k) - \frac{'\beta_1 \times '('B + 'k)}{'\alpha} \Longrightarrow '('C + 'k)$

22 ...

$'C \Longrightarrow 'Cr$
$'B \Longrightarrow 'Br$
$\pi 2$
$n \Longrightarrow {}^2B$
$'B + '3011 \Longrightarrow '0004$
$\pi 1\,(B) \Longrightarrow '\beta_2$
$Ц23\,\{1\,(1)\,'r_2 \Longrightarrow 'k\}\,24$
$'('D + 'k) + \frac{'\beta_1 \times '('A + 'k)}{'\alpha} \Longrightarrow '('D + 'k)$
$'('B + 'k) + \frac{'\beta_2 \times '('A + 'k)}{'\beta_1} \Longrightarrow '('A + 'k)$

24 ...

$'\beta_2 \Longrightarrow '\beta_1$

25 ...

$Ц26\,\{1\,(1)\,'r_2 \Longrightarrow 'i\}$
$P\,\{|'('C + 'i)| \leqslant \varepsilon\}\,26 \downarrow 15$

27 ...

$'D \Longrightarrow 'Ar$
$'C \Longrightarrow 'Br$
$\pi 10$
$Ц30\,\{1\,(1)\,'r_2 \Longrightarrow 'k\}\,31$
$'('R + 'k) - '('C + 'k) \Longrightarrow '('C + 'k)$

31 ...

$Ц32\,\{1\,(1)\,'r_2 \Longrightarrow 'k\}\,33$
$P\,\{|'('C + 'k)| \leqslant \varepsilon\}\,32 \downarrow 14$

33 ...

$Ц34\,\{1\,(1)\,'r_2 \Longrightarrow 'k\}\,35$
$'('D + 'k) \Longrightarrow '0002$

НС	—	0002	—
30	3026	0000	3100
22	0003	0003	0000

35 ...

Ост

Строки от метки 1 до метки 2 — подпрограмма скалярного произведения векторов (задана как нестандартный оператор); строки от метки 2 до 10 — подпрограмма умножения транспонированной матрицы на вектор; строки от метки 10 до 13 включительно — подпрограмма произведения матрицы на вектор.

Распределение памяти

0001 — j
0002 — Cr, Br
0003 — i
0004 — Ar
0005 — h
0006 — r_1
0007 — r_2
0010 — s
0011 — R
0012 — A
0013 — B
0014 — D
0015 — β_1
0016 — α
0017 — β_2

1774 — r_3
1775 — r_4
1776 — k
1777 — s_1
3000 — H
3001 — C
'3002 — n
'3003 — n_1
'3004 — $\left[\frac{n}{2}\right]$
'3005 — ε

Рабочая программа, выданная ПП-АК

```
0020 01 0000 0000 0003
   1 34 0002 0000 0005
   2 34 0004 0000 0006
   3 34 0002 0000 0000
   4 11 4001 0006 0006
   5 01 0006 0003 0003
   6 01 0002 3011 0002
   7 01 0004 3011 0004
0030 02 0005 3011 0005
   1 31 0005 0022 3146
   2 01 3012 0000 0001
   3 01 3010 0000 0003
   4 01 3011 0000 0005
   5 01 0000 0000 0010
   6 01 3000 0005 0566
   7 34 0566 0000 0000
0040 15 4000 3024 0006
   1 16 0006 0001 0045
   2 01 0005 3011 0005
   3 02 0005 3003 0566
   4 31 0566 0071 0036
   5 01 3000 0005 0566
   6 34 0566 0000 0000
   7 15 4000 3022 0006
0050 02 0006 0003 0566
   1 02 0003 0006 0565
   2 15 0565 0566 0566
   3 31 0566 0065 0054
   4 11 0003 3040 0006
   5 01 0002 0006 0566
   6 01 3000 0005 0565
   7 34 0565 0000 0000
0060 34 4000 0000 0565
   1 34 0566 0000 0000
   2 11 0565 4000 0565
   3 01 0565 0010 0010
   4 01 0005 3011 0005
   5 01 0003 3010 0003
   6 02 0005 3003 0566
   7 31 0566 0071 0070
0070 04 0003 1774 0036
   1 12 0001 3040 0006
   2 01 0004 0006 0566
   3 34 0566 0000 0000
   4 01 0010 0000 4000
   5 01 0001 3012 0001
   6 04 0001 1775 0033
   7 32 0000 0000 0000
0100 01 3010 0000 0003
   1 01 3011 0000 0005
   2 01 3012 0000 0001
   3 01 0000 0000 0010
   4 01 3000 0005 0566
   5 34 0566 0000 0000
   6 15 4000 3024 0006
   7 02 0006 0001 0566
0110 02 0001 0006 0565
   1 15 0565 0566 0566
   2 31 0566 0124 0113
   3 12 0001 3040 0006
   4 01 0004 0006 0566
   5 01 3000 0005 0565
   6 34 0565 0000 0000
   7 34 4000 0000 0565
0120 34 0566 0000 0000
   1 11 0565 4000 0565
   2 01 0565 0010 0010
   3 01 0005 3011 0005
   4 01 0001 3012 0001
   5 04 0001 1775 0104
   6 11 0003 3040 0006
   7 01 0002 0006 0566
0130 34 0566 0000 0000
   1 01 0010 0000 4000
   2 01 0003 3010 0003
   3 04 0003 1774 0102
   4 32 0000 0000 0000
   5 12 3002 3040 1774
   6 11 3002 3040 1775
   7 02 3002 3011 0007
0140 01 3002 3011 0006
   1 01 3001 0006 0011
   2 01 0011 0006 0012
   3 01 0012 0006 0013
   4 01 0013 0006 0014
   5 01 0012 0000 0004
   6 01 3001 0000 0002
   7 30 3026 0150 0032
0150 34 0012 0000 0000
   1 01 3002 0000 4000
   2 01 0012 3011 0004
```

```
   3 01 0012 0000 0002
   4 30 3026 0155 0020
   5 01 0003 0000 0015
   6 02 3004 3011 0006
   7 02 0000 3011 1777
0160 01 1777 3011 1777
   1 02 3011 3011 1776
   2 01 1776 3011 1776
   3 01 0014 1776 0566
   4 34 0566 0000 0000
   5 01 0000 0000 4000
   6 04 1776 0007 0162
   7 01 0012 0000 0004
0170 01 0013 0000 0002
   1 30 3026 0172 0100
   2 34 0013 0000 0000
   3 01 3002 0000 4000
   4 01 0013 3011 0004
   5 01 0013 0000 0002
   6 30 3026 0177 0020
   7 01 0003 0000 0016
0200 02 3011 3011 1776
   1 01 1776 3011 1776
   2 01 3001 1776 0566
   3 01 0013 1776 0565
   4 34 0565 0000 0000
   5 11 0015 4000 0564
   6 12 0564 0016 0564
   7 34 0566 0000 0000
0210 02 4000 0564 4000
   1 04 1776 0007 0201
   2 01 0012 0000 0004
   3 01 3001 0000 0002
   4 30 3026 0215 0032
   5 34 0013 0000 0000
   6 01 3002 0000 4000
   7 01 0013 3011 0004
0220 01 0013 0000 0002
   1 30 3026 0222 0020
   2 01 0003 0000 0017
   3 02 3011 3011 1776
   4 01 1776 3011 1776
   5 01 0014 1776 0566
   6 01 0012 1776 0565
   7 34 0565 0000 0000
0230 11 0015 4000 0564
   1 12 0564 0016 0564
   2 34 0566 0000 0000
   3 01 4000 0564 4000
   4 01 0012 1776 0566
   5 34 0566 0000 0000
   6 11 0017 4000 0565
   7 12 0565 0015 0565
0240 01 0013 1776 0564
   1 34 0564 0000 0000
   2 01 4000 0565 0564
   3 34 0566 0000 0000
   4 01 0564 0000 4000
   5 04 1776 0007 0224
   6 01 0017 0000 0015
   7 04 1777 0006 0160
0250 02 3011 3011 0003
   1 01 0003 3011 0003
   2 01 3001 0003 0566
   3 34 0566 0000 0000
   4 06 4000 3005 0566
   5 31 0566 0156 0256
   6 04 0003 0007 0251
   7 01 0014 0000 0004
0260 01 3001 0000 0002
   1 30 3026 0262 0100
   2 02 3011 3011 1776
   3 01 1776 3011 1776
   4 01 3001 1776 0566
   5 01 0011 1776 0565
   6 34 0565 0000 0565
   7 34 0566 0000 0000
0270 02 0565 4000 4000
   1 04 1776 0007 0263
   2 02 3011 3011 1776
   3 01 1776 3011 1776
   4 01 3001 1776 0566
   5 34 0566 0000 0000
   6 06 4000 3005 0566
   7 31 0566 0145 0300
0300 04 1776 0007 0273
   1 02 3011 3011 1776
   2 01 1776 3011 1776
   3 01 0014 1776 0566
   4 34 0566 0000 0002
   5 30 3026 0312 3100
   6 22 0003 0003 0313
   7 04 1776 0007 0302
0310 33 0000 0000 0000
```

II. Приведем программы прямого и обратного хода решения системы линейных алгебраических уравнений с симметричной матрицей усовершенствованным методом Гаусса (см. гл. III).

Адресный алгоритм в стиле ПП-АК

```
Прямой ход...
'ψ — 1 ⟹ 'r
0 ⟹ 's
1 ⟹ 'δ
1...
's + 'δ ⟹ 's
's ⟹ 's1
'δ ⟹ 'q
1 ⟹ 'δ1
2...
'φ ⟹ 'β
1 ⟹ 'π
3...
'β + 'π ⟹ 'β
    ' ('φ + 's1 + 'δ + 1) — ' ('φ +
+ 's + 'π) × '('φ + 's1 + 'π) :
: 2β ⟹ '('φ + 's1 + 'δ + 1)
    'π + 1 ⟹ 'π
    P {'π ⩽ 'δ} 3 ↓ 0
    'q + 1 ⟹ 'q
    's1 + 'q ⟹ 's1
    'δ1 + 1 ⟹ 'δ1
    P {'δ1 ⩽ ('ψ — 'δ + 1)} 2 ↓ 0
    'δ + 1 ⟹ 'δ
    P {'δ ⩽ 'r} 1 ↓ 0
    Я
```

Исходное адресное отображение

β — 0001	q — 0006
r — 0002	$'\delta_1$ — 0007
s — 0003	π — 0010
δ — 0004	r_1 — 0011
s_1 — 0005	φ — 3000
	ψ — 3001

Рабочая программа, выданная ПП-АК (прямой ход)

```
0020 02 3001 3011 0002
   1 01 0000 0000 0003
   2 01 3011 0000 0004
   3 01 0003 0004 0003
   4 01 0003 0000 0005
   5 01 0004 0000 0006
   6 01 3011 0000 0007
   7 01 3000 0000 0001
0030 01 3011 0000 0010
   1 01 0001 0010 0001
   2 01 3000 0005 0077
   3 01 0077 0004 0076
   4 01 0076 3011 0076
   5 01 3000 0003 0075
   6 01 0075 0010 0075
   7 34 0075 0000 0075
0040 34 0001 0000 0000
   1 12 0075 4000 0075
   2 01 0077 0010 0077
   3 34 0077 0000 0000
   4 11 0075 4000 0075
   5 34 0076 0000 0000
   6 02 4000 0075 4000
   7 01 0010 3011 0010
0050 04 0010 0004 0031
   1 01 0006 3011 0006
   2 01 0005 0006 0005
   3 01 0007 3011 0007
```

```
   4 01 3001 3011 0077
   5 02 0077 0004 0077
   6 04 0007 0077 0027
   7 01 0007 3011 0007
0060 04 0004 0002 0023
   1 33 0000 0000 0000
```

Адресный алгоритм в стиле ПП-АК

Обратный ход...

$'n \Longrightarrow 'i$
$0 \Longrightarrow 'j$
$'\varphi - 'n \Longrightarrow 'D$
1...
$'\varphi - 'n - j \Longrightarrow 'B$
$'n - 'i \Longrightarrow 'r$
$0 \Longrightarrow m'$; $0 \Longrightarrow 'k$; $0 \Longrightarrow 's$
2...
$P\{'k = 'r\}\, 3 \downarrow 0$
$'B - 'm \Longrightarrow 'B$
$'('A - 'k) \times {}^2B + 's \Longrightarrow 's$
$'k + 1 \Longrightarrow 'k$
$'n - 'k \Longrightarrow 'm$
$P_0 \to 2$
3...
$'s + '('\varphi - 'j) \Longrightarrow 'r$
$0 - 'r \Longrightarrow 'r$
$'r : {}^2D \Longrightarrow '('A - 'j)$
$'D - 'i \Longrightarrow 'D$
$'j + 1 \Longrightarrow 'j$
$'i - 1 \Longrightarrow 'i$
$P\{'i = 0\}\, 0 \downarrow 1$
$'n - 1 \Longrightarrow 'r$
4...
$'A - 'r \Longrightarrow '0002$
УПП 3026 ↓ 3100
22 0003 0003 ↓
$'r - 1 \Longrightarrow 'r$
$P\{0 \leqslant 'r\}\, 4 \downarrow 0$
Я

Исходное адресное отображение

φ — 3000
n — 3001
D — 0001
B — 0002
j — 0003
i — 0004
r — 0005
m — 0006
k — 0007
s — 0010
A — 0011

Рабочая программа, выданная ПП-АК (обратный ход)

```
0020 01 3001 0000 0004
   1 01 0000 0000 0003
   2 02 3000 3001 0001
   3 02 3000 3001 0077
   4 02 0077 0003 0002
   5 02 3001 0004 0005
   6 01 0000 0000 0006
   7 01 0000 0000 0007
0030 01 0000 0000 0010
   1 16 0007 0005 0043
   2 02 0002 0006 0002
   3 02 0011 0007 0077
   4 34 0077 0000 0077
   5 34 0002 0000 0000
   6 11 0077 4000 0077
   7 01 0077 0010 0010
0040 01 0007 3011 0007
   1 02 3001 0007 0006
   2 31 0000 0031 0031
   3 02 3000 0003 0077
   4 34 0077 0000 0000
   5 01 0010 4000 0005
   6 02 0000 0005 0005
   7 34 0001 0000 0000
0050 12 0005 4000 0077
   1 02 0011 0003 0076
```

```
   2 34 0076 0000 0000        2 31 0077 0023 0063
   3 01 0077 0000 4000        3 02 3001 3011 0005
   4 02 0001 0004 0001        4 02 0011 0005 0002
   5 01 0003 3011 0003        5 30 3026 0066 3100
   6 02 0004 3011 0004        6 22 0003 0003 0067
   7 02 0000 0004 0077        7 02 0005 3011 0005
0060 02 0004 0000 0076     0070 31 0005 0064 0071
   1 15 0076 0077 0077        1 33 0000 0000 0000
```

ПРОГРАММИРУЮЩАЯ ПРОГРАММА ПП-2*

В основу разработки ПП-2 положены следующие требования:

1. Язык ПП-2 должен быть близок к языку машины «Киев».
2. ПП-2 не должна загромождать внутреннюю память.
3. ПП-2 должна быть удобна в эксплуатации.

При этом имелось в виду, что чем больше входной язык программирующей программы будет отличаться от языка машины, тем труднее будет ориентироваться в рабочей программе (РП). С этим связан чрезвычайно актуальный вопрос отладки рабочих программ: где искать ошибки — на уровне входного языка или в РП?

Операторы программ

В рабочей программе для машин прежних образцов необходимо было различать операторы переадресации, восстановления, формирования констант переадресации и другие. Для машины, имеющей регистр модификации адреса, допустима более простая классификация операторов программ: различается четыре типа операторов.

1. Операторы преобразования типа арифметических операторов, которые перерабатывают внешнюю относительно РП информацию (исходные данные и данные, получаемые в процессе счета).

2. Логические операторы, определяющие порядок выполнения преобразований информации и управляющие процессом счета.

3. Операторы модификации адресов, управляющие выборкой информации из памяти в процессе ее преобразования в циклах.

4. Технические операторы, которые управляют процессом ввода и вывода информации из машины, обменом кодами с внешними запоминающими устройствами, изменяют режим работы машины и задают остановку.

---

* ПП-2 разработана Л. Н. Иваненко под руководством Е. Л. Ющенко [8]. В программировании арифметического оператора принимала также участие Н. М. Грищенко.

Рассмотрим способы записи операторов для программирующей программы, не касаясь их представления в операциях конкретных машин.

Операторы преобразования обычно представляются алгебраическими формулами и формулами булевой алгебры от адресов различных рангов. Назовем эти адреса входными. Набор алгебраических операций может включать операции $|-|$, $\vee$, $\wedge$, $E(x)$ и т. д. Вычисление по формуле предполагает получение однозначного результата. Адрес, по которому записывается результат, назовем выходным и будем выделять особым знаком *. Общий вид формулы будет

$$F(A_1, \ldots, A_n, D) \Rightarrow A^*,$$

где D — набор операций; A_i — адреса.

Одним из методов записи *логических операторов* является построение предикатных формул, которые в зависимости от истинности или ложности некоторого высказывания дают разветвления вычислительного процесса по двум направлениям. Можно также идти по пути построения набора элементарных логических операторов (стандартных предикатных формул), из которых строятся операторы многих классов задач. Особенно удобными являются наборы операторов для узких классов задач, например таких, как машинный перевод.

Вопрос о выборе набора крупных операторов для достаточно широкого класса задач еще не решен. Можно отметить решение, в котором переход на оператор преобразования определяется по набору значений логических переменных. Для записи оператора в этом виде необходимо выделить все допустимые наборы значений логических переменных и сопоставить им номера операторов преобразования. Эта форма записи удобна для логических задач, хотя возникают трудности при ее переводе в программы ЦАМ.

Стремясь сделать ПП-2 максимально короткой и предопределяя до некоторой степени ее применение к программированию в первую очередь арифметических задач, мы избрали наиболее простое, но не оптимальное решение и использовали в качестве элементарных логических операторов предикативные формулы, реализуемые схемно машиной «Киев», а именно:

$$P\{a \leqslant b\}\, x \downarrow n+1;$$
$$P\{|a| \leqslant b\}\, x \downarrow n+1;$$
$$P\{|a| = |b|\}\, x \downarrow n+1;$$
$$P\{a \leqslant -0\}\, x \downarrow y,$$

где x, y — произвольные номера; $n+1$ — номер команды, следующей за той, в которой записана предикатная формула.

В последнем случае записана передача управления по знаку числа. Этой передаче может предшествовать вычисление истинности некоторого высказывания, значения переменных в котором записаны в знаковых разрядах ячеек памяти. Собственно вычисления выполняются оператором преобразования.

Таким образом, предикатные формулы записываются в виде готовых команд машины «Киев», но с условными адресами (x, y). Обработка логического блока ПП-2 заключается в присвоении истинных адресов.

Что касается записи операторов модификации для ПП-2, то и здесь удалось избежать специальных построений благодаря непосредственному использованию в языке, с одной стороны, операций машины «Киев», с другой — нестандартных операторов.

Основная роль *технических операторов* — управлять обменом кодами с внешними запоминающими устройствами. Часть из них может быть записана заранее, так как объем числового материала и его распределение в памяти известны. Что же касается подпрограмм, то команды для их записи на барабан формируются самой ПП-2 по окончании программирования. Те технические операторы, которые могут быть записаны до программирования, представляются командами машины «Киев» и вводятся как нестандартные операторы.

Распределение памяти и кодирование

Ввиду ограниченности объема оперативной памяти машины возникает необходимость членения программ на подпрограммы и их последовательный ввод с барабана. В ПП-2 предусмотрена реализация этого процесса. Весь исходный материал для программирования делится на части, из которых впоследствии составляются подпрограммы. В начале каждой части ставится знак НП (начало подпрограммы), в конце — КП (конец подпрограммы). Последняя подпрограмма носит название центрального управления (ЦУ) и имеет особые знаки начала (НЦ) и конца (КЦ). Подпрограммы непосредственно между собой не связаны. Каждая имеет свой набор стандартных ячеек для получения исходных данных и выдачи результатов. Поэтому в оперативной памяти в каждый данный момент должна находиться одна подпрограмма. Вызов подпрограмм с барабана, необходимые пересылки исходных данных и передачу управления подпрограмм реализует ЦУ, находящееся в памяти постоянно. Этим предопределяется распределение памяти, схематически показанное в табл. 4. Номера первых команд подпрограмм совпадают. ЦУ начинается после наиболее длинной подпрограммы.

Так как в оперативной памяти помещается ограниченный объем исходных данных ПП-2, то приходится делить их на зоны по k элементов в каждой, где k равно половине соответствующего

Таблица 4

Распределение памяти

<table>
<tr><td rowspan="2">Распределение памяти машины «Киев»</td><td colspan="6">Оперативная память 1024</td><td rowspan="2">Пассивная память 512 ячеек</td></tr>
<tr><td>Γ_4</td><td>Γ_x</td><td>Γ_2</td><td>Γ_3</td><td>Γ_1</td><td>Γ_a</td></tr>
<tr><td>Для ПП-2</td><td>Рабочие ячейки ПП-2</td><td>Массив мест для записи РП</td><td rowspan="2">Массив рабочих ячеек РП</td><td>Внешняя информация ПП-2</td><td>Словарь ОЗУ</td><td rowspan="2">Словарь ВЗУ</td><td>ПП-2</td></tr>
<tr><td>Для РП</td><td>Рабочие ячейки стандартных подпрограмм</td><td>Массив подпрограмм Массив ЦУ</td><td colspan="2">Внешняя информация РП</td><td>Стандартные подпрограммы</td></tr>
</table>

массива мест в ОЗУ. Первоначально в машину вводятся две зоны исходных данных; в процессе программирования использованные элементы информации (ЭИ) стираются, и запись сжимается. В момент, когда в массиве остается менее чем k ЭИ, с перфоленты вводится следующая зона. Деление на зоны производится механически и не имеет ничего общего с делением на подпрограммы.

Этапы и циклы работы ПП-2. Рабочие программы, создаваемые ПП-2, состоят из подпрограмм и ЦУ. Все подпрограммы программируются одинаково. Для ЦУ отличие состоит в адресе первой команды. Поэтому внешний цикл заключается в программировании подпрограмм. В нем необходимо различать три этапа: первый — программирование арифметических формул и перезапись элементов остальных операторов; второй — присваивание истинных адресов; третий — контрольное суммирование и запись на барабан.

Уже упоминалось о простоте поиска первой выполнимой операции при представлении формул в записи Лукасевича слева направо. Как только операция найдена, формируется команда. Особое исследование проводится перед заполнением III адреса. Если после операции в исходной информации стоит выходной адрес, то он записывается в III адрес команды. Появление выходного адреса означает конец программирования формулы. В противном случае в III адрес ставится адрес рабочей ячейки. Одновременно производится экономия рабочих ячеек с использованием их массива. Массив рассматривается в направлении уменьшения адресов, и адрес первого места, содержащего $+0$, ставится в команду, по этому же адресу записывается—0. Если номера рабочих ячеек попадают в I или II адрес команды, соответствующий адрес «освобождается», т. е. по этому адресу записывается $+0$. Отме-

чается номер наибольшей (с наименьшим адресом) рабочей ячейки и сравнивается с номером очередной команды. При пересечении массивов происходит аварийная остановка ПП-2.

Одноместная операция программируется в виде обращения к стандартным подпрограммам. Рабочие ячейки экономятся, как и в предыдущем случае. После записи готовой команды в рабочую программу исходная информация ПП-2 сжимается, т. е. стираются использованные элементы информации и остаток массива подтягивается вперед и в определенные моменты осуществляется считывание следующей зоны. Признаком конца записи служит $+0$. Нестандартные операторы записываются после знака W (n) и переписываются в рабочую программу. Процесс продолжается до появления сигнала «конец подпрограммы». Затем наступает второй этап программирования.

Теперь подпрограмма просматривается и выделяются по очереди все команды с условными адресами. По условному адресу в словаре ОЗУ находится истинный адрес, который подставляется в команду. Особо обрабатывается элемент информации T (пересылка в условных адресах), который необходим для организации передач управления с ЦУ на подпрограммы и внутри подпрограмм с возвращением к определенному месту программы. В команде $Tx_1 - x_2$ оба адреса условные. Здесь x_1 — место хранения константы, задающей возвращение с подпрограммы; x_2 — адрес некоторой стандартной ячейки, используемой в качестве регистра возврата. Подпрограммы заканчиваются передачей управления на эту стандартную ячейку, а так как она в свою очередь содержит команду безусловного перехода, то окончательно управление передается к нужному месту программы.

Используя ЭИ F и W, можно построить произвольное число регистров возврата. В команде T, помимо присвоения истинных адресов, производится замена кода операции на знак плюс (пересылка).

Третий этап — вспомогательный. Вычисляется контрольная сумма подпрограммы и записывается перед первой командой. Затем формируются команды записи подпрограммы на барабан вместе с ее контрольной суммой и реализуется запись. Формируются также команды чтения с барабана и запоминаются в словаре ВЗУ на N- и $(N+1)$-месте от начала. После этого подпрограмма вновь читается в ОЗУ, суммируется и сравниваются контрольные суммы. При несовпадении следует аварийная остановка ПП-2.

Очевидно, следующий элемент за КП в исходной информации — НП или НЦ. При НП цикл повторяется. При НЦ устанавливается адрес первой команды ЦУ. По сигналу КЦ, помимо суммирования и контроля, вся программа выводится на печать в виде, пригодном для введения ее в машину (перфорируется).

Таким образом, удалось добиться значительной близости языков

ПП-2 и машины «Киев», Для увеличения компактности записи в одну ячейку записывается по два элемента информации арифметических операторов, хотя и для них допускается кодирование по одному элементу в ячейке, а также вычеркивание элементов, закодированных ошибочно дважды. Слова всех остальных операторов занимают полные ячейки. Признаком конца массива служит единица в конце его записи. Благодаря этому и расположению ПП-2 в блоках сменно-спаянной памяти ПП занимает всего 300 восьмеричных (192 десятичных) ячеек оперативной памяти (без блока перевода в бесскобочную запись).

ПП-2 выполнена в виде двух блоков — арифметического оператора, формирующего и записывающего арифметические команды и экономящего рабочие ячейки, и блока обработки остальных слов языка ПП-2, ввода с перфокарт, записи на барабан и т. д.

Распределение памяти. При использовании ПП-2 предусматривается типовое распределение памяти (см. табл. 4). В первую очередь оценивают длину будущей рабочей программы (РП) и, если выясняется, что она вся не может разместиться в оперативной памяти, находят целесообразное деление на подпрограммы.

Поскольку подпрограммы РП хранятся во внешней памяти (барабан) и считываются одна поверх другой, то в РП для управления процессом считывания и выполнения различных преобразований информации должен быть блок, постоянно находящийся в ОЗУ. Этот блок назван центральным управлением (ЦУ). Если программа полностью размещается в ОЗУ, тогда она и есть ЦУ.

Информация о распределении памяти помещается в наборных кодах (НК) памяти с номерами 3000—3004:

3000			Γ_4	
3001			Γ_2	
3002			k	
3003			$2m$	

Здесь m — число подпрограмм, с помощью которого определяется граница словаря ВЗУ; Γ_4 — адрес начала записи РП в ОЗУ. В НК 3004 кодируется номер барабана и места на нем.

Затем определяется объем внешней информации (ВИ) РП, которая включает исходные данные и промежуточные значения переменных. Допустим, что ВИ составляет n слов, тогда $\Gamma_2 = \Gamma_1 - n = \Gamma_0 - 2m - n$. После этого определяется N — число операторов в подпрограммах или в ЦУ, задающее объем словаря ОЗУ. Тогда $\Gamma_3 = \Gamma_1 - N = \Gamma_0 - 2m - N$. Теперь можно рассчитать

объем зоны для внешней информации ПП-2. В интервале (Γ_2, Γ_3) должно помещаться две зоны. Поэтому, беря разность $\Gamma_3 - \Gamma_2$ и уменьшая ее до ближайшего четного $[\Gamma_3 - \Gamma_2]$, получаем

$$k = \frac{[\Gamma_3 - \Gamma_2]}{2}.$$

Минимальное значение $\Gamma_4 = 31$, так как ячейки 0001—0030 используются самой ПП-2.

Входной язык ПП-2

Типы слов, принятые для записи алгоритмов в языке ПП-2, представлены в табл. 5.

Таблица 5

Типы слов входного языка ПП-2

№ типа слова	Элемент информации (слово)	Символ	Кодировка			
			0-й А	IA	IIA	IIIA
1	Входной адрес	a	01	a		
2	Выходной »	a^*	21	a		
4	Входной модифицируемый адрес	$\bar{a}$	01	$4000 + a$		
3	Выходной » »	$\bar{a}^*$	21	$4000 + a$		
5	Одноместная операция	s	23	№ входной команды подпрограммы		
6	Двухместная »	P	03	Код в языке «Киев»		
7	Сравнение со знаком	$Ср1$	04	a	b	x
8	» по модулю	$Ср2$	05	a	b	x
9	» на совпадение	$Ср3$	16	a	b	x
10	Передача управления по знаку числа	$УПЧ$	31	a	x	y
11	Метка оператора	F_x	11		x	
12	» подпрограммы	M_i	10			i
13	» нестандартного оператора	W	17		n	
14	Печать	$Печ$	22	a	b	x
15	Пересылка в истинных адресах $И \to И$	$\Longrightarrow$	37	a		b
16	Пересылка в условных адресах $У \to У$	T_3	36		x	y
17	Пересылка $И \to У$	T_6	36	x	a	
18	Пересылка $У \to И$	T_5	36	a		x
19	Начало подпрограммы	НП	12			
20	Конец »	КП	13			i
21	Начало центрального управления	НЦ	14			
22	Конец центрального управления	КЦ	15			
23	Конец зоны входной информации	!	07			

Примечание. a, b — истинные адреса; x, y — номера операторов; i — номер подпрограммы; n — число ЭИ в нестандартном операторе.

Типы 1—6 используются для записи арифметических операторов.

Типы 7—10 употребляются для записи логических операторов. Их отличие от соответствующих аналогов в языке машины «Киев» состоит только в том, что в соответствующих адресах вместо номера команды ставится номер оператора, к которому передается управление.

Метка оператора ставится перед тем оператором, к которому имеют место передачи управления, и указывает его номер.

Метка подпрограммы ставится в ЦУ и служит сигналом для ввода в ОЗУ подпрограммы, номер которой указан в метке.

Метка нестандартного оператора ставится перед группой кодов, вводимых в РП непосредственно в языке машины «Киев» (без их переработки ПП-2).

Употребление слов типов 7—10 в нестандартных операторах допускается только в том случае, если они служат не для передач управления внутри нестандартного оператора, а только для передач вне его. В последнем случае в адресах ставятся номера операторов. Допускается употребление машинных операций с номерами 20 и более, что в свою очередь еще расширяет входной язык ПП-2.

Если тип 16 допускает замену арифметическим оператором $a0 + b^*$, то роль типов 17—18 в языке существенна, так как они позволяют оперировать командами и, в частности, пересылать их или переадресовывать (последнее совершенно необходимо для принятого режима плавающей запятой, где использование групповых операций запрещено). Собственно T включает три слова или три варианта пересылок:

$$У \to У;\ И \to У;\ У \to И,$$

где $У$ — условный адрес, а $И$ — истинный.

Пример. Требуется переадресовать команду $cba \times$ в I и III адресах (умножение вектора на константу). В языке ПП-2 запись имеет вид

$$F(x)\, ab \times c^* T_6(x,\ f)\, f\delta_{101} + f' T_5(f,\ x).$$

Этой записи соответствует программа

$$\begin{array}{r} n \times a\ b\ \ c \\ n+1+n-\ \ f \\ n+2+f\ \delta_{101}\ f \\ n+3+f\ -\ n \end{array}$$

Здесь δ_{101} — константа вида 00 0001 0000 0001. Команды $n+1$ и $n+3$ — избыточные, но следует учесть, что прямая переадресация в машине «Киев» употребляется редко. Пересылкой T_3 можно воспользоваться для восстановления команды n.

Назначение типов 20—23 определяется их названием. Поскольку тип 23 отмечает конец записи в зоне, он всегда ставится на последнем месте.

Программу можно записывать в символическом виде в одну строку. При этом вместо символа адреса α надо писать само значение адреса a, т. е. вместо α^* пишется $21a$; вместо $\bar{\alpha}^*$ пишется $214\,000 + a$; одноместные и двухместные операции записываются их символами $\ln a$, $\sin \bar{a}$, $+$, $\times$ и т. д. Для передач управления нужно в скобках указать аргументы предикатной формулы $Cp2(abx)$. Также поступают при записи других слов.

Можно непосредственно записывать программу в языке ПП-2 на бланках для перфорации. При этом нужно помнить, что элементы информации арифметических операторов вносятся по два в одну ячейку, а признак второго слова в ячейке записывается во II адресе. a запишется так:

03	0001	0001	a

Если запись арифметического оператора обрывается на половине ячейки, то вторую половину оставляют пустой. Остальные слова языка ПП-2 занимают полные ячейки.

Запись исходной информации механически разбивается на зоны равной длины и в конце каждой зоны ставится знак «!» (код 07). Зоны получают номера от 0001.

Последняя зона может оказаться неполной, поэтому ее следует после знака «!» дополнить маркерами (нулями) до положенного числа в k маркеров.

Арифметический блок ПП-2

Арифметический блок ПП-2 программирует арифметические формулы любой глубины, состоящие из величин, двухместных и одноместных операций и записанные в бесскобочной записи Лукасевича справа налево.

Подготовка внешней информации для арифметического блока ПП-2. Арифметический блок составляет команды для программ, работающих в режиме фиксированной запятой, поэтому масштабные множители должны предусматриваться программистом заранее и вноситься в арифметическую формулу.

Каждой величине арифметической формулы необходимо сопоставить ячейку памяти машины «Киев», которую назовем входной ячейкой, и заранее подобрать ячейку для помещения результата вычисления по данной формуле, которую назовем выходной ячейкой.

Для примера рассмотрим формулу

$$f = (b - a)\frac{d}{a \ln (b + c)}, \qquad (23)$$

где a, b, c, и d — некоторые числа, а f — результат вычисления

по формуле. Разместим эти числа соответственно в ячейки H_1, H_2, H_3 и H_4, а результат f поместим в выходную ячейку H^*:

$$H^* = \frac{H_4}{H_1 \ln (H_3 + H_2)} (H_2 - H_1). \qquad (24)$$

Запись формулы (24) в бесскобочной записи Лукасевича имеет вид

$$H_1 H_2 - H_1 H_2 H_3 + \ln \times H_4 : \times H^*. \qquad (25)$$

Кодирование элементов арифметических формул. Для ввода арифметических формул в машину каждый элемент кодируется в 17 разрядах ячейки машины «Киев». Таким образом, в каждой ячейке машины «Киев» размещается два элемента.

В первых пяти разрядах каждой группы кодируются признаки, позволяющие различать величины и операции. В следующих 12 разрядах размещены:

а) адрес, в котором находится величина, если кодируемый элемент — величина;

б) номер первой команды подпрограммы, если кодируемый элемент — одноместная операция;

в) код операции, если кодируемый элемент — двухместная операция.

Выходные ячейки кодируются так же, как и величины, только в первых пяти разрядах помещается специальный признак (см. табл. 5). Если H_1, H_2, H_3 и H_4 соответствуют ячейкам 0007, 0010, 0011, 0012, а H^* — 0014, то элементы формулы (24) кодируются каждый в отдельности следующим образом:

H_1 01 0007	ln 23 3116
H_2 01 0010	× 03 0011
— 03 0002	H_4 01 0012
H_1 01 0007	: 03 0012
H_2 01 0010	× 03 0011
H_3 01 0011	H^* 21 0014
+ 03 0001	

Разместив по два элемента в каждую ячейку, получим следующие восьмеричные числа, которые вместе с элементами логических операторов задачи составляют внешнюю информацию для ПП-2.

$K+1$	01	0007	0001	0010
$K+2$	03	0002	0001	0007
$K+3$	01	0010	0001	0011
$K+4$	03	0001	0023	3116
$K+5$	03	0011	0001	0012
$K+6$	03	0012	0003	0011
$K+7$	21	0014	0000	0000

Последовательность обработки входной информации и записи результатов для формулы (25) приведена в табл. 6, где U — фиксатор элементов входной информации. Перечеркивание клеток в табл. 6 обозначает стирание информации.

Таблица 6

Работа арифметического блока

№ выполнимой операции	H_1	H_2	—	H_1	H_2	H_3	+	ln	×	H_4	:	×	H^*	
	U													
		U												
1			U											$-H_2H_1r_1$
	r_1	×	×	H_1	H_2	H_3	+	ln	×	H_4	:	×	H^*	
				U										
					U									
						U								
2							U							$+H_3H_2r_2$
	r_1	×	×	H_1	r_2	×	×	ln	×	H_4	:	×	H^*	
3								U						$+ r_3 - 0002$ УПП 3026 к. 5 к. 3116
	r_1	×	×	H_1	r_2	×	×	×	×	H_4	:	×	H^*	$+0003-r_2$
4									U					$\times r_2H_1r_2$
	r_1	×	×	r_2	×	×	×	×	×	H_4	:	×	H^*	
										U				
5											U			$:H_4r_2r_2$
	r_1	×	×	r_2	×	×	×	×	×	×	×	×	H^*	
6												U		$\times r_2r_1H^1$
	×	×	×	×	×	×	×	×	×	×	×	×	×	

ПРИМЕР ПРОГРАММЫ, СОСТАВЛЕННОЙ ПП-2

Требуется вычислить значения несобственного интеграла по формуле

$$I_n = \int_0^{\infty} e^{-x} f(x)\,dx \approx \sum_{k=1}^{n} A_k f(x_k)$$

при определенных значениях параметров, где A_k — некоторые числа, а функция $f(x)$ задана соотношением

$$f(x_k) = (x_k + 4\lambda)^{2\left(\frac{R}{\lambda} - 1\right)} \psi_0(x_k).$$

При помощи арифметического блока ПП-2 была запрограммирована формула для I_n при $n = 0$ и $x_k = x_1$ при $k = 1$.

$$I_0 = A_1 f(x_1),$$

где

$$f(x_1) = (x_1 + 4\lambda)^{2\left(\frac{R}{\lambda} - 1\right)} \psi_0(x_1);$$

$$\psi_0(x_1) = D_0 B_0(\xi)\,[b_0(x_1)];$$

$$B_0(\xi) = 2\xi - \gamma;$$

$$b_0(x_1) = (x_1 + 2\lambda)\left[\frac{2\left(\frac{R}{\lambda} - 1\right)}{x + 2\lambda} - 1\right].$$

Здесь D_0 — некоторая константа.

Перепишем I_0 в виде

$$I_0 = \exp\left\{\ln A_1 + \ln D_0 + 2\left(\frac{R}{\lambda} - 1\right)\ln(x + 4\lambda) + \ln[12\xi^3 - 6\xi^2\gamma - 8\xi] + \right.$$
$$\left. + \ln\left[2\left(\frac{R}{\lambda} - 1\right) + \left(x + 2\lambda\right)\right]\right\}.$$

После выбора масштабных множителей, размещения величин в адреса $H_i\,(1 \leqslant i \leqslant 16)$ и экономии формул имеем окончательно

$$H'' = \frac{H_5}{H_{12}} - H_{11}; \qquad H' = \frac{H_1}{H_{12}H_2} + H_{11};$$

$$I_0 = \exp\{\ln H_3 + \ln H_4 + H_{12}H''\ln(H_1 + H_{13}H_2) + \ln[H_{14}H'H'H' - H_{15}H'H'H_6 - H_{16}H']\}\exp\{\ln[H_{12}H'' + H_1 + H_{12}H_2]\}.$$

Запись Лукасевича слева направо этих формул

$$H_{11}H_2H_5 : - H''H_{11}H_2H_{12} \times H_1 : + H'H_3 \ln H_4 \ln +$$
$$+ H''H_{12} \times H_2H_{13} \times H_1 + \ln \times + H'H_{16} \times H_6H' \times H' \times H_{15} \times -$$
$$- H'H' \times H' \times H_{14} \times - \ln + H''H_{12}H_2H_{12} \times H_1 + + \ln + eH'.$$

Заметим, что формулы кодируются в такой последовательности, в какой их нужно программировать (H_1, ... , H_{16} соответственно 0001, ... , 0016):

```
K + 10  00 0000 0000 0000
        1  01 0011 0001 0002
        2  01 0005 0003 0012
        3  03 0002 0021 0021
        4  01 0011 0001 0002
        5  01 0012 0003 0011
        6  01 0001 0003 0012
        7  03 0001 0021 0020
K + 20  01 0003 0023 3116
        1  01 0004 0023 3116
        2  03 0001 0001 0021
        3  01 0012 0003 0011
        4  01 0002 0001 0013
        5  03 0011 0001 0001
        6  03 0001 0023 3116
        7  03 0011 0003 0001
K + 30  01 0020 0001 0016
        1  03 0011 0001 0006
        2  01 0020 0003 0011
        3  01 0020 0003 0011
        4  01 0015 0003 0011
        5  03 0002 0001 0020
        6  01 0020 0003 0011
        7  01 0020 0003 0011
K + 40  01 0014 0003 0011
        1  03 0002 0023 3116
        2  03 0001 0001 0021
        3  01 0012 0003 0011
        4  01 0002 0001 0012
        5  03 0011 0001 0001
        6  03 0001 0003 0001
        7  23 3116 0000 0000
K + 50  03 0001 0000 0000
K + 51  00 0000 0000 0000
        2  23 3220 0021 0020
        3  07 0000 0000 0000
```

Рабочая программа получена следующая:

```
0050 12 0005 0002 0177
     1  02 0177 0011 0021
     2  11 0012 0002 0177
     3  12 0001 0177 0177
     4  01 0177 0011 0020
     5  01 0003 0000 0002
     6  30 3026 0057 3116
     7  01 0003 0000 0177
0060 01 0004 0000 0002
     1  30 3026 0062 3116
     2  01 0003 0000 0176
     3  01 0176 0177 0177
     4  11 0012 0021 0176
     5  11 0013 0002 0175
     6  01 0001 0175 0175
     7  01 0175 0000 0002
0070 30 3026 0071 3116
     1  01 0003 0000 0175
     2  11 0175 0176 0176
     3  01 0176 0177 0177
     4  11 0016 0020 0176
     5  11 0020 0006 0175
     6  11 0020 0175 0175
     7  11 0015 0175 0175
0100 02 0175 0176 0176
     1  11 0020 0020 0175
     2  11 0020 0175 0175
     3  11 0014 0175 0175
     4  02 0175 0176 0176
     5  01 0176 0000 0002
     6  30 3026 0107 3116
     7  01 0003 0000 0176
0110 01 0176 0177 0177
     1  11 0012 0021 0176
     2  11 0012 0002 0175
     3  01 0001 0175 0175
     4  01 0175 0176 0176
     5  01 0076 0000 0002
     6  30 3026 0117 3116
     7  01 0003 0000 0176
0120 01 0176 0177 0177
     1  01 0179 0000 0002
     2  30 3026 0123 3220
     3  01 0003 0000 0020
```

Приложение 1

ПУЛЬТ УПРАВЛЕНИЯ (ПУ)

В настоящем приложении описаны лишь те управляющие и сигнальные элементы, с которыми приходится встречаться при отладке программы и решении задач. Элементы, с помощью которых производится проверка правильности работы машины, из описания исключены.

Пульт управления (ПУ) состоит из наклонной панели сигнализации (рис. 3) и горизонтальной панели управления (рис. 4).

На панели сигнализации расположены контрольные лампочки, связанные с различными блоками устройств управления (УУ), арифметического (АУ) и запоминающего (ЗУ). Эти лампочки позволяют судить о действиях, происходящих в данный момент в различных блоках машины.

Набор сигнальных лампочек верхнего ряда *См* связан с сумматором и показывает его работу. При чтении содержимого сумматора следует иметь в виду, что числа на сумматоре представлены в модифицированном обратном коде. Лампочки *Зн2С* и *Зн1С*, в которых записывается знак числа на сумматоре, служат одновременно для сигнализации о переполнении разрядной сетки: если число в сумматоре больше или равно +1, запишется 10, если оно меньше или равно —1, запишется 01. При отсутствии переполнения в этих разрядах при положительном числе запишется 00, а при отрицательном 11. Лампочка *ЗнР2* связана со знаковым разрядом регистра второго числа и показывает знак числа, выбираемого по II адресу.

На пульт управления выведены три тумблерные ячейки (наборные коды НК) 3000, 3001, 3002, включенные параллельно этим же ячейкам на шкафу ПЗУ. Каждая ячейка пульта управляется тумблером. При включенном тумблере (положение вверх) работает ячейка пульта, при выключенном — соответствующая ячейка ПЗУ. Если в задаче используются все НК, то на панели НК следует набрать коды для задачи, а на пульте — команды ввода, исправления ячеек и т. д. После ввода тумблер НК выключается. Если в программе используются НК, которые нужно менять в процессе счета, то их следует набирать на ПУ.

Набор из 32 лампочек, обозначенный *КОп*, сигнализирует об операции, которая выполняется в данный момент. Все лампочки этого набора расположены в порядке кодировки операции и снабжены их символами.

Лампочка *ТрВыб* зажигается при работе машины в режиме обмена с внешними запоминающими устройствами, а именно: перфолентой (перфокартами), магнитной лентой и магнитными барабанами.

На индикацию *А*-регистра (*РА*) подается адрес, участвующий в выполнении данного такта команды. *А*-регистр сбрасывается при этом в 0, поэтому содержимое его для отладок не используется.

Блок индикации регистра возврата (*РВ*) характеризует состояние регистра возврата. *РВ* сбрасывается в 0 при выполнении команды *ПРВ*.

Лампочка *ТрПр* (триггер признака условного перехода) зажигается при выполнении команды, вырабатывающей признак условного перехода, если этот

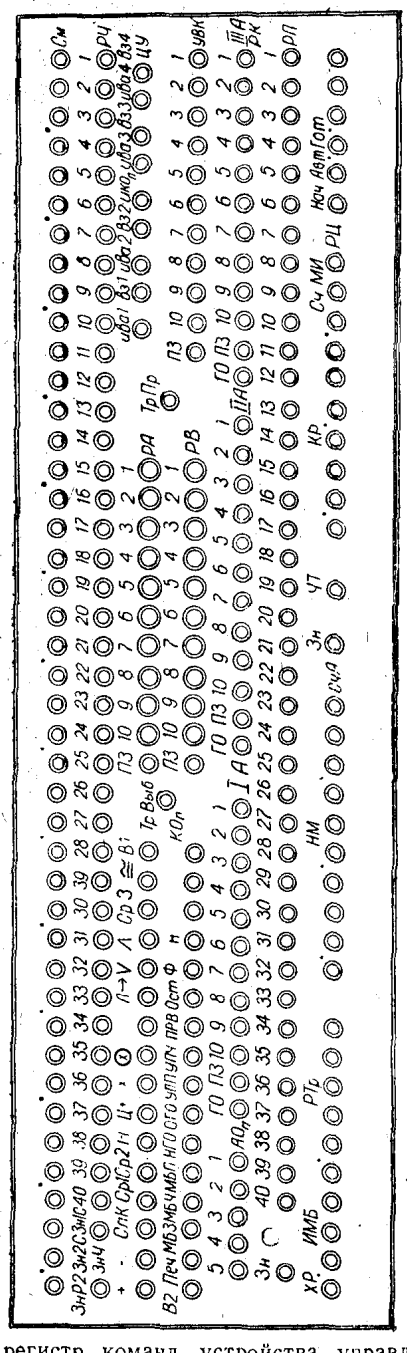


Рис. 3. Панель сигнализации.

признак выработан. Лампочка *Авт* зажигается при работе машины на автомате. Лампочка *Зн* зажигается при выходе из разряда.

Блоки сигнальных лампочек *РА* и регистр циклов *РЦ* характеризуют состояние регистров *А* и *Ц* в момент работы машины.

Блок сигнальных лампочек *ЦУ* предназначен для контроля по тактам работы машины. Лампочки *ива1*, *ива2*, *ива3* сигнализируют о передаче в запоминающее устройство импульсов выборки соответственно содержимого I, II и III адресов команды, лампочки *вз1*, *вз2* — об окончании выборки в арифметическое устройство содержимого соответствующих адресов, лампочка *вз3* — об окончании записи числа по III адресу, лампочки *ива4* и *вз4* — о подаче импульса выборки выполняемой команды и о приеме этой команды в устройство управления, лампочка *ИКОп* — об окончании выполнения операции арифметическим устройством.

Блок сигнальных лампочек *УВК* (устройства выдачи команд) показывает содержимое счетчика команд *С* — номер выполняемой команды. При чтении номера команды следует иметь в виду, что он изменяется по окончании выполнения арифметическим устройством предыдущей команды (зажигание сигнала *ИКОп*).

На сигнализацию *РК* регистра команд *К* поступает команда, подлежащая выполнению. Момент ее поступления на регистр команд совпадает с моментом срабатывания *вз4*. Для удобства чтения индикатор *РК* разбит на группы в соответствии со структурой команды. В момент останова на *РК* содержится выполненная команда, на *УВК* — номер следующей команды, для команд *УПЧ*, *УПП*, *ОГО* — номер выполненной команды, для команд *В1*, *В2*, *Печ*, *МБЗ*, *МБЧ* — нуль. Поэтому на последних командах нельзя делать останов по номеру команды.

Выборка и исполнение команды происходят в следующем порядке:

а) в соответствии с номером, записанным на *УВК*, из запоминающего устройства выбирается команда в регистр команд устройства управления. Эта команда записывается на *РК*. О начале и конце выборки сигнализируют лампочки *ива4* и *вз4*;

б) выбирается содержимое I адреса команды, записанной на *РК*. Номер адреса записывается на *РА*. Начало и конец выборки отмечают лампочки *ива1* и *вз1*. *РА* сбрасывается в нуль;

в) выполняется то же, что и в п. б) для II адреса;

г) выполняется операция, предусмотренная выбранной командой. Окончание ее отмечается зажиганием лампочки *ИКОп*;

д) происходит изменение *УВК* для выборки следующей команды. Одновременно результат выполнения операции передается в запоминающее устройство. Начало и конец посылки отмечаются лампочками *ива3* и *вз3*;

е) происходит переход к выполнению операций, указанных в п. а.

Лампочки *Зп* и *Чт* сигнализируют о режиме работы: запись или чтение. Сигнализация *КР* (коммутатор разворота), *СчМИ* (счетчик маркерных импульсов), *Нач* и *Гот* (начало и готовность) контролирует процесс обмена кодами.

На панели управления (рис. 4) размещен тумблерный регистр команд, на котором может быть набрана произвольная команда. Этот регистр команд позволяет выполнять с пульта желаемую команду.

Кнопка *УО* (установка нуля) предназначена для установки в нуль всех регистров устройств при начальном запуске и при выполнении различных операций с пульта управления.

Включением тумблера *НР — Бл УВК* производится блокировка изменения содержимого счетчика команд, что позволяет многократно выполнять одну и ту же команду. Тумблер *УО СчА* блокирует работу *СчА*, *РЦ* и *РВ*. Это дает возможность при отладке программ повторять их куски внутри циклов и стандартных подпрограмм, так как при нажатии кнопки *УО* и включении тумблера *УО СчА*, *РВ* и *РЦ* не сбрасываются в нуль.

После передачи управления на желаемый участок программы тумблер отключается для дальнейшей работы со счетчиком адреса или регистром возврата.

Тумблер *Пч* подключает выводные устройства на быстродействующую печать, медленную печать или читающий автомат.

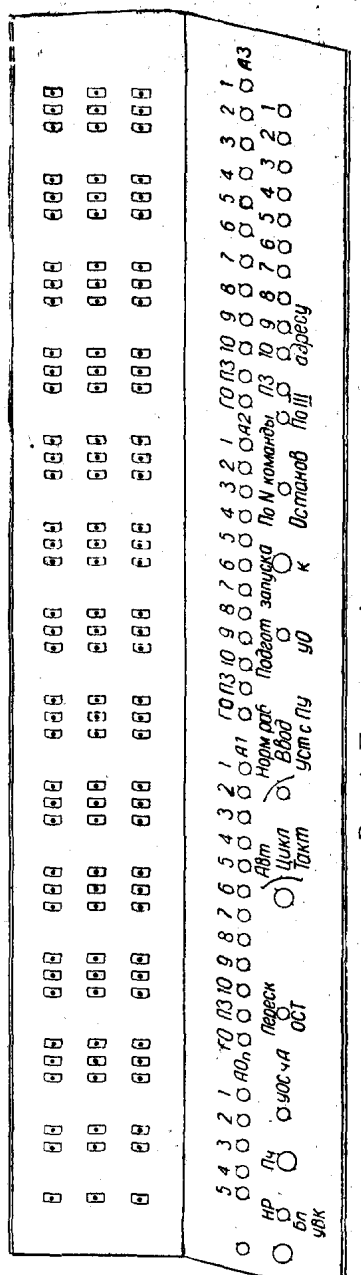


Рис. 4. Панель управления.

Переключатель *Авт-Цикл-Такт* предназначен для переключения машины на различные виды работы. Положению *Авт* этого переключателя соответствует автоматическая работа, т. е. автоматическое выполнение программы, введенной в машину; положению *Цикл* — выполнение программы по циклам, т. е. по командам; положению *Такт* — выполнение программы по тактам.

Выше упоминалось, что выполнение команды производится в четыре такта: выборка команды, выборка I адреса, выборка II адреса и выдача по адресу с одновременной перестройкой устройства управления на выполнение следующей команды. Если при выполнении программы по циклам вся команда выполняется при одном нажатии ключа *К*, то для включения команды по тактам требуется четыре нажатия на ключ в соответствии с перечисленными тактами.

Переключатель *Норм. раб — Уст с ПУ* служит для переключения машины на выполнение команды, набранной на пульте управления (положение *Уст с ПУ*) и на нормальную работу (положение *Норм. раб*), т. е, на выполнение программы, записанной во внутренней памяти машины.

Кнопка *Останов* предназначена для останова машины. При нажатии на эту кнопку машина останавливается после выполнения команды, находящейся в момент нажатия кнопки на регистре команд.

Тумблер останова имеет три положения: нейтральное, останов по номеру команды и останов по III адресу. Вместе с тумблерным регистром останова этот тумблер предназначен для набора операций «останов по III адресу» и «останов по номеру команды».

В момент останова машины на *РК* находится код только что выполненной операции, на *См* — результат этой операции, на *УВК* — номер следующей команды, а для операций типа передач управления (*УПП, УПЧ*) — номер выполненной команды.

Во время операции останова на *См* остаются результаты предыдущей команды. На *ПУ* имеется тумблер блокировки выхода из разряда «с перескоком», позволяющий при выходе из разряда, пропустив одну команду, продолжать работу. Специальной кнопкой при включенном тумблере производится стирание куба *ОЗУ*.

Порядок выполнения действий, связанных с отладкой программы и решением задач.

1. Начальный ввод программы:

а) установить перфоленту (или колоду перфокарт);

б) переключатель *Авт-Цикл-Такт* установить в положение *Авт*, а переключатель *Норм. раб — Уст с ПУ* в положение *Уст с ПУ;*

в) набрать на тумблерном регистре команд команду ввода 20 $\alpha\beta$ (для ввода чисел) или 21 $\alpha\beta$ (для ввода команд), где α и β — начальная и конечная ячейки программы;

г) включить устройство ввода;

д) нажать на кнопку *УО*;

е) нажатием ключа *К* произвести ввод программы.

Лента должна прокрутиться полностью. Остаток является характеристикой данной ленты, и его следует запомнить.

При рассмотренном способе машина останавливается после окончания ввода. Этот способ требует после ввода специального перехода к выполнению программы. Однако можно набрать команду ввода в наборной памяти и команду передачи управления, передать управление по адресу, по которому набрана нужная команда, и, таким образом, без дополнительных переключений перейти к выполнению введенной программы.

Проверка содержимого ячейки запоминающего устройства:

а) переключатель *Авт-Цикл-Такт* установить в положение *Такт*.

б) переключатель *Норм. раб. — Уст с ПУ* установить в положение *Норм. раб.;*

в) набрать в III адресе тумблерного регистра команд номер требуемой ячейки;

г) нажать кнопку *УО*;

д) нажать ключ *К* и прочитать содержимое вызванной ячейки на сигнализационном блоке *См* пульта управления.

3. Выполнение команды, записанной в запоминающем устройстве:

а) в зависимости от того, в каком режиме требуется выполнить команду, устанавливается переключатель *Авт-Цикл-Такт*. Положению *Авт* соответствует автоматическое выполнение программы, положению *Цикл* — выполнение программы по командам, положению *Такт* — выполнение программы по тактам;

б) произвести действия, указанные в пп. б, в, г, при описании начального ввода;

в) нажим на ключ *К* при установке *Авт* переключателя *Авт-Цикл-Такт* запускает машину на автоматическую работу; при положении *Цикл* нажим на ключ *К* приводит к выполнению одной команды, для выполнения следующей команды требуется вторичный нажим на ключ; при положении *Такт* для выполнения одной команды требуется нажать на ключ *К* четыре раза.

4. Выполнение команды с пульта управления:

а) набрать в тумблерном регистре команд пульта управления требуемую команду;

б) переключатель *Авт-Цикл-Такт* установить в положение *Цикл;*

в) переключатель *Норм. раб.— Уст с ПУ* установить в положение *Уст с ПУ*;

г) нажать кнопку *УО;*

д) нажим на ключ *К* выполняет команду, набранную в тумблерном регистре команд.

5 Исправление содержимого ячейки:

а) набрать в наборной памяти требуемое содержимое ячейки;

б) набрать в тумблерном регистре команд пульта управления команду пересылки кода, набранного в наборной памяти в требуемую ячейку;

в) выполнить действия, указанные в пп. б—д при описании выполнения команды с пульта управления.

6. Останов по III адресу и останов по номеру команды:

а) установить тумблер останова в положение требуемого режима останова;

б) набрать нужный адрес или номер команды на тумблерном регистре ячейки останова;

в) при запуске машины на автоматическую работу произойдет останов по заказанному режиму.

Проверка правильности ввода программы или отдельных ее участков может быть произведена либо последовательной проверкой участка, либо выводом на восьмеричную печать этого участка командой вывода, выполненной с пульта.

Останов машины, работающей в режиме автоматической работы, производится нажатием кнопки *Останов*. Запуск машины для продолжения счета производится нажатием ключа *К*.

Приложение 2

ТЕСТЫ

Тест устройства управления (УУ)

Команды 0001—0006 — контрольное суммирование программы.

» 0163—0173 — проверка операции умножения с признаком групповой операции (ГО) во всех адресах.

» 0241—0250 — проверка работы устройства ввода команд (*УВК*) как счетчика (с 1-го по 10-й разряд).

» 0251—0275 — проверка работы *УВК* при передаче на него кода из всех единиц и одного нуля и кода из всех нулей и одной единицы (с 1-го по 10-й разряд).

» 0276—0277 — проверка 11-го разряда *УВК*.

» 0300—0303 — проверка 11-го разряда регистра возврата (*РВ*).

» 0304—0425 — проверка *А*-регистра и сумматора адреса (*СмА*) на кодах из всех единиц и одного нуля.

Команды 0426—0510 — проверка работы регистра циклов (*РЦ*)
» 0511—0565 — проверка переносов в *СмА*.
» 0566—0617 — проверка работы *СмА*.
» 0620—0641 — проверка работы признака ГО в операциях передачи управления.
» 0643—0662 — проверка работы операции нормализации с признаком ГО во всех адресах.
» 0700—0706 — проверка работы признака ГО в операциях передачи управления при наличии единицы в 11-м разряде.

```
0001 01 0000 0000 1220
   2 26 0706 0000 0000
   3 07 4001 1220 1220
   4 27 4001 0003 0005
   5 22 1220 1220 0006
   6 33 0000 0000 0000
   7 01 0075 0000 1524
0010 01 0076 0000 1525
   1 01 0077 0000 1526
   2 01 0100 0000 1527
   3 01 0101 0000 1530
   4 01 0102 0000 1247
   5 01 0103 0000 1250
   6 01 0104 0000 1251
   7 01 0105 0000 1252
0020 01 0106 0000 1253
   1 01 3035 0000 0107
   2 01 3035 0000 0110
   3 01 3035 0000 0111
   4 01 3035 0000 0112
   5 01 3035 0000 0113
   6 01 3035 0000 0114
   7 01 3035 0000 0115
0030 01 3035 0000 0116
   1 01 3035 0000 0117
   2 01 3035 0000 0120
   3 01 3035 0000 0121
   4 01 3035 0000 0122
   5 01 3035 0000 0123
   6 01 3035 0000 0124
   7 04 0000 0002 0052
0040 02 0000 0000 0107
   1 05 0002 0000 0052
   2 02 0000 0000 0110
   3 16 0002 0000 0052
   4 02 0000 0000 0111
   5 31 0000 0046 0052
   6 02 0000 0000 0112
   7 30 0000 0000 0052
0050 02 0000 0000 0113
   1 04 0000 0000 0053
   2 22 0107 0124 0006
   3 02 0000 0000 0114
   4 05 0000 0000 0056
   5 22 0107 0124 0006
   6 02 0000 0000 0115
   7 16 0000 0000 0061
0060 22 0107 0124 0006
   1 02 0000 0000 0116
   2 31 0002 0060 0063
   3 02 0000 0000 0117
   4 30 0002 1527 0066
   5 22 0107 0124 0006
   6 02 0000 0000 0120
   7 32 0000 0000 0000
0070 26 0124 0107 0000
   1 31 4000 0074 0072
   2 27 4001 0071 0073
   3 31 3000 0125 0007
   4 22 0107 0124 0006
   5 33 0000 0000 0000
   6 02 0000 0000 0123
   7 04 0000 0000 1252
0100 02 0000 0000 0121
   1 30 0002 1250 0067
   2 33 0000 0000 0000
   3 02 0000 0000 0122
   4 04 0000 0000 1525
   5 02 0000 0000 0124
   6 04 0000 0000 0070
   7 00 0000 0000 0000
0110 00 0000 0000 0000
   1 00 0000 0000 0000
   2 00 0000 0000 0000
   3 00 0000 0000 0000
   4 00 0000 0000 0000
   5 00 0000 0000 0000
   6 00 0000 0000 0000
   7 00 0000 0000 0000
0120 00 0000 0000 0000
   1 00 0000 0000 0000
   2 00 0000 0000 0000
   3 00 0000 0000 0000
   4 00 0000 0000 0000
   5 01 0207 0000 1525
   6 01 0210 0000 1252
   7 34 0212 0000 0215
0130 16 0215 0207 0132
   1 33 0000 0000 0000
   2 34 0213 0000 0215
   3 16 0215 0210 0135
   4 33 0000 0000 0000
   5 34 0214 0000 0215
   6 16 0215 0211 0140
   7 33 0000 0000 0000
```

```
0140 26 1525 1525 0142
   1 33 0000 0000 0000
   2 26 1252 1252 0144
   3 33 0000 0000 0000
   4 01 0000 0000 0216
   5 26 0005 0000 0000
   6 01 4207 0216 0216
   7 27 4001 0146 0150
0150 16 0216 0217 0152
   1 33 0000 0000 0000
   2 26 1777 0525 0000
   3 27 5252 0154 0155
   4 33 0000 0000 0000
   5 26 1525 0252 0000
   6 27 5253 0157 0160
   7 33 0000 0000 0000
0160 26 1000 0777 0000
   1 27 4001 0162 0163
   2 33 0000 0000 0000
   3 26 1775 1253 0000
   4 01 3042 0000 4000
   5 01 3042 0000 4001
   6 10 4000 4001 4002
   7 16 4002 3043 0171
0170 33 0000 0000 0000
   1 01 0000 0000 4000
   2 01 0000 0000 4001
   3 27 4002 0164 0174
   4 31 3000 0240 0125
   5 00 0000 0000 0000
   6 00 0000 0000 0000
   7 00 0000 0000 0000
0200 00 0000 0000 0000
   1 00 0000 0000 0000
   2 00 0000 0000 0000
   3 00 0000 0000 0000
   4 00 0000 0000 0000
   5 00 0000 0000 0000
   6 00 0000 0000 0000
   7 00 0000 0000 0001
0210 00 0000 0000 0002
   1 00 0000 0000 0003
   2 00 0000 1525 0000
   3 00 0000 1252 0000
   4 00 0000 0211 0000
   5 00 0000 0000 0000
   6 00 0000 0000 0000
   7 00 0000 2777 0006
0220 01 3043 0000 0200
   1 16 3043 0200 0245
   2 01 3044 0000 0200
   3 16 3044 0200 0251
   4 16 0000 0000 0254
   5 16 0000 0000 0256
   6 16 0000 0000 0260
   7 16 0000 0000 0262
0230 16 0000 0000 0264
   1 16 0000 0000 0266
   2 16 0000 0000 0270
0233 16 0000 0000 0272
   4 16 0000 0000 0274
   5 16 0000 0000 0276
   6 00 0000 0000 0025
   7 32 0000 0000 0000
0240 01 0170 0000 0001
   1 01 0220 0000 0777
   2 01 0221 0000 1000
   3 01 0170 0000 1001
   4 04 0000 0000 0777
   5 01 0222 0000 1777
   6 01 0223 0000 0001
   7 01 0170 0000 0002
0250 04 0000 0000 1777
   1 01 0170 0000 1777
   2 01 0224 0000 1776
   3 04 0000 0000 1776
   4 01 0225 0000 1775
   5 04 0000 0000 1775
   6 01 0226 0000 1773
   7 04 0000 0000 1773
0260 01 0227 0000 1767
   1 04 0000 0000 1767
   2 01 0230 0000 1757
   3 04 0000 0000 1757
   4 01 0231 0000 1737
   5 04 0000 0000 1737
   6 01 0232 0000 1677
   7 04 0000 0000 1677
0270 01 0233 0000 1577
   1 04 0000 0000 1577
   2 01 0234 0000 1377
   3 04 0000 0000 1377
   4 01 0235 0000 0777
   5 04 0000 0000 0777
   6 01 0170 0000 1777
   7 03 0236 0224 1776
0300 30 3026 1776 0237
   1 03 0236 0225 1775
   2 30 3026 1775 0237
   3 03 0236 0226 1773
   4 30 3026 1773 0237
   5 03 0236 0227 1767
   6 30 3026 1767 0237
   7 03 0236 0230 1757
0310 30 3026 1757 0237
   1 03 0236 0231 1737
   2 30 3026 1737 0237
   3 03 0236 0232 1677
   4 30 3026 1677 0237
   5 03 0236 0233 1577
   6 30 3026 1577 0237
   7 03 0236 0234 1377
0320 30 3026 1377 0237
   1 03 0236 0235 0777
   2 30 3026 0777 0237
   3 01 0170 0000 1147
   4 30 3026 0325 3146
   5 01 0170 0000 1043
```

```
0326 01 0330 0000 0001
     7 30 3026 3043 0237
0330 04 0000 0000 0331
     1 01 0000 0000 1777
     2 01 3042 0000 1776
     3 26 0000 1776 0000
     4 16 3042 4000 0336
     5 33 0000 0000 0000
     6 16 4000 3042 0340
     7 33 0000 0000 0000
0340 01 3042 0000 1775
     1 26 0000 1775 0000
     2 16 3042 4000 0344
     3 33 0000 0000 0000
     4 16 4000 3042 0346
     5 33 0000 0000 0000
     6 01 3042 0000 1773
     7 26 0000 1773 0000
0350 16 3042 4000 0352
     1 33 0000 0000 0000
     2 16 4000 3042 0354
     3 33 0000 0000 0000
     4 01 3042 0000 1767
     5 26 0000 1767 0000
     6 16 3042 4000 0360
     7 33 0000 0000 0000
0360 16 4000 3042 0362
     1 33 0000 0000 0000
     2 01 3042 0000 1757
     3 26 0000 1757 0000
     4 16 3042 4000 0366
     5 33 0000 0000 0000
     6 16 4000 3042 0370
     7 33 0000 0000 0000
0370 01 3042 0000 1737
     1 26 0000 1737 0000
     2 16 3042 4000 0374
     3 33 0000 0000 0000
     4 16 4000 3042 0376
     5 33 0000 0000 0000
     6 01 3042 0000 1677
     7 26 0000 1677 0000
0400 16 3042 4000 0402
     1 33 0000 0000 0000
     2 16 4000 3042 0404
     3 33 0000 0000 0000
     4 01 3042 0000 1577
     5 26 0000 1577 0000
     6 16 3042 4000 0410
     7 33 0000 0000 0000
0410 16 4000 3042 0412
     1 33 0000 0000 0000
     2 01 3042 0000 1377
     3 26 0000 1377 0000
     4 16 3042 4000 0416
     5 33 0000 0000 0000
     6 16 4000 3042 0420
     7 33 0000 0000 0000
0420 01 3042 0000 0777
```

```
0421 26 0000 0777 0000
     2 16 3042 4000 0424
     3 33 0000 0000 0000
     4 16 4000 3042 0426
     5 33 0000 0000 0000
     6 01 0000 0000 1777
     7 01 3035 0000 1776
0430 26 1776 1770 0000
     1 27 4001 0431 0432
     2 16 4000 3035 0434
     3 33 0000 0000 0000
     4 01 3035 0000 1775
     5 26 1775 1770 0000
     6 27 4001 0436 0437
     7 16 3035 4000 0441
0440 33 0000 0000 0000
     1 01 3035 0000 1773
     2 26 1773 1770 0000
     3 27 4001 0443 0444
     4 16 3035 4000 0446
     5 33 0000 0000 0000
     6 01 3035 0000 1767
     7 26 1767 1763 0000
0450 27 4001 0455 0451
     1 16 4000 3035 0453
     2 33 0000 0000 0000
     3 01 3035 0000 1757
     4 26 1757 1753 0000
     5 27 4001 0455 0456
     6 16 4000 3035 0460
     7 33 0000 0000 0000
0460 01 3035 0000 1737
     1 26 1737 1733 0000
     2 27 4001 0462 0463
     3 16 4000 3035 0465
     4 33 0000 0000 0000
     5 01 3035 0000 1677
     6 26 1677 1673 0000
     7 27 4001 0467 0470
0470 16 4000 3035 0472
     1 33 0000 0000 0000
     2 01 3035 0000 1577
     3 26 1577 1573 0000
     4 27 4001 0474 0475
     5 16 4000 3035 0477
     6 33 0000 0000 0000
     7 01 3035 0000 1377
0500 26 1377 1374 0000
     1 27 4001 0501 0502
     2 16 4000 3035 0504
     3 33 0000 0000 0000
     4 01 3035 0000 0777
     5 26 0777 0770 0000
     6 27 4001 0506 0507
     7 16 4000 3035 0511
0510 33 0000 0000 0000
     1 26 0000 0001 0000
     2 16 0002 4001 0514
     3 33 0000 0000 0000
```

```
0514 16 0004 4003 0516
   5 33 0000 0000 0000
   6 16 0010 4007 0520
   7 33 0000 0000 0000
0520 16 0020 4017 0522
   1 33 0000 0000 0000
   2 16 0040 4037 0524
   3 33 0000 0000 0000
   4 16 0100 4077 0526
   5 33 0000 0000 0000
   6 16 0200 4177 0530
   7 33 0000 0000 0000
0530 16 0400 4377 0532
   1 33 0000 0000 0000
   2 01 3043 0000 1000
   3 16 3043 4777 0535
   4 33 0000 0000 0000
   5 26 0000 0003 0000
   6 16 4001 0004 0540
   7 33 0000 0000 0000
0540 26 0000 0007 0000
   1 16 4001 0010 0543
   2 33 0000 0000 0000
   3 26 0000 0017 0000
   4 16 4001 0020 0546
   5 33 0000 0000 0000
   6 26 0000 0037 0000
   7 16 4001 0040 0551
0550 33 0000 0000 0000
   1 26 0000 0077 0000
   2 16 4001 0100 0554
   3 33 0000 0000 0000
   4 26 0000 0177 0000
   5 16 4001 0200 0557
   6 33 0000 0000 0000
   7 26 0000 0377 0000
0560 16 4001 0400 0562
   1 33 0000 0000 0000
   2 26 0000 0777 0000
   3 16 4001 1000 0565
   4 33 0000 0000 0000
   5 31 3000 0566 0240
   6 26 0000 0003 0000
   7 16 4003 0006 0571
0570 33 0000 0000 0000
   1 26 0000 0007 0000
   2 16 0016 4007 0574
   3 33 0000 0000 0000
   4 26 0000 0017 0000
   5 16 4017 0036 0577
   6 33 0000 0000 0000
   7 26 0000 0037 0000
0600 16 0076 4037 0602
   1 33 0000 0000 0000
   2 26 0000 0077 0000
   3 16 4077 0176 0605
   4 33 0000 0000 0000
   5 26 0000 0177 0000
   6 16 0376 4177 0610
```

```
0607 33 0000 0000 0000
0610 01 3035 0000 0776
   1 26 0000 0377 0000
   2 16 4377 3035 0614
   3 33 0000 0000 0000
   4 01 3035 0000 1776
   5 26 0000 0777 0000
   6 16 3035 4777 0620
   7 33 0000 0000 0000
0620 26 0000 1000 0000
   1 01 0664 0000 1623
   2 04 0000 0000 4623
   3 33 0000 0000 0000
   4 01 0665 0000 1626
   5 05 0000 0000 4626
   6 33 0000 0000 0000
   7 01 0666 0000 1631
0630 16 0000 0000 4631
   1 33 0000 0000 0000
   2 01 0667 0000 1634
   3 31 3036 0634 4634
   4 33 0000 0000 0000
   5 01 0670 0000 1637
   6 31 3035 4637 0637
   7 33 0000 0000 0000
0640 01 3146 0000 1642
   1 30 3026 0643 4642
   2 33 0000 0000 0000
   3 35 0671 1777 4000
   4 16 1777 0672 0646
   5 33 0000 0000 0000
   6 16 1000 0673 0650
   7 33 0000 0000 0000
0650 35 0674 4000 1777
   1 16 1000 0675 0653
   2 33 0000 0000 0000
   3 16 1777 0676 0655
   4 33 0000 0000 0000
   5 01 0677 0000 1000
   6 35 4000 1776 1777
   7 16 1776 3026 0661
0660 33 0000 0000 0000
   1 16 1777 0677 0663
   2 33 0000 0000 0000
   3 04 0000 0000 0700
   4 04 0000 0000 0624
   5 04 0000 0000 0627
   6 04 0000 0000 0632
   7 04 0000 0000 0635
0670 04 0000 0000 0640
   1 05 2525 2525 2525
   2 20 0000 0000 0001
   3 12 5252 5252 5252
   4 21 2525 2525 2525
   5 20 0000 0000 0003
   6 32 5252 5252 5250
   7 12 5252 5252 5252
0700 26 0000 0140 0000
   1 30 3026 0702 7006
```

```
0702 26 0000 0137 0000
   3 30 3026 0706 0704
   4 31 0670 7007 0705
   5 33 0000 0000 0000
   6 31 3000 0007 0566
```

НК:

```
3006 33 0000 0000 0000
3007 33 0000 0000 0000
```

Тесты запоминающего устройства (ЗУ)

Адреса 0001—0013 — константы.
Команды 0014—0021 — контрольное суммирование теста.

Первый тест ОЗУ

Команды 0022—0027 — выбор кода для записи.
» 0030—0032 — запись кода в ОЗУ.
» 0034—0037 — считывание кода с проверкой.
» 0040—0042 — счетчик числа чтений.
Команда 0043 — переход к записи следующей константы.

Второй тест ОЗУ (*«щекотка» и полувыборка)*

Команды 0046—0047 — выбор констант для записи в куб ОЗУ и в диагональные ячейки.
» 0050—0052 — запись в куб.
» 0053—0061 — запись в диагональные ячейки и многократное их считывание («щекотка»).
» 0062—0070 — проверка недиагональных ячеек.
» 0072—0073 — изменение константы записи.

Третий тест ОЗУ

Команды 0100—0103 — запись кода 1010 ... в ОЗУ.
» 0104—0110 — чтение по *x*.

Четвертый тест ОЗУ

Команды 0122—0136 — запись «бегающей» единицы или нуля.
» 0140—0155 — чтение ОЗУ с проверкой.

```
0001 37 7777 7777 7777
   2 25 2525 2525 2525
   3 12 5252 5252 5252
   4 01 0001 0001 0001
   5 33 0000 0000 0777
   6 01 0004 0000 0020
   7 01 0000 0000 0020
0010 00 0041 0041 0000
   1 26 1777 1737 0000
   2 00 0040 0040 0000
   3 26 2000 1740 0000
   4 01 0000 0000 0210
0015 26 0166 0000 0000
   6 07 4001 0210 0210
   7 27 4001 0016 0020
0020 22 0210 0210 0021
   1 33 0000 0000 0000
   2 31 3001 0025 0023
   3 01 3002 0000 0020
   4 05 0000 0000 0030
   5 01 0007 0000 0026
   6 01 0000 0000 0020
   7 03 3010 0026 0026
0030 26 1777 0177 0000
```

```
0031 01 0020 0000 4001
   2 27 4001 0031 0033
   3 03 3011 3200 0036
   4 26 0000 0200 0000
   5 16 4000 0020 0037
   6 33 0000 1000 0000
   7 27 4001 0035 0040
0040 03 3012 0036 0036
   1 05 0036 0165 0034
   2 31 3001 0043 0044
   3 05 0026 0006 0026
   4 31 3000 0045 0022
   5 01 0000 0000 0014
   6 01 0000 0000 0020
   7 01 3036 0000 0021
0050 26 1777 0200 0000
   1 01 0020 0000 4000
   2 27 4001 0051 0053
   3 26 1573 0000 0000
   4 01 0021 0000 4204
   5 03 0000 3200 0015
   6 07 4204 4204 0000
   7 03 3012 0015 0015
0060 05 0015 0005 0056
   1 27 4041 0054 0062
   2 01 0077 0000 0063
   3 26 0245 0205 0000
   4 16 4000 0020 0066
   5 33 0000 0002 0000
   6 27 4001 0064 0067
   7 03 0010 0063 0063
0070 05 0063 0011 0063
   1 31 0014 0072 0076
   2 02 0000 0000 0014
   3 01 0000 0000 0021
   4 01 3036 0000 0020
   5 05 0000 0000 0050
   6 31 3000 0100 0045
   7 26 0245 0205 0000
0100 01 0116 0000 0105
   1 26 1600 0000 0000
   2 01 0002 0000 4200
   3 27 4001 0102 0104
   4 03 3011 0065 0107
   5 26 0240 0200 0000
   6 16 4000 0002 0110
   7 33 0000 0000 0000
0110 27 4001 0106 0111
   1 03 3012 0107 0107
   2 05 0107 0166 0105
   3 03 0012 0105 0105
   4 05 0105 0013 0104
   5 31 3000 0117 0100
   6 26 0240 0200 0000
   7 01 0000 0000 0014
0120 01 3012 0000 0017
   1 01 0017 0000 0021
   2 26 0000 0200 0000
   3 01 0021 0000 0020
   4 31 0014 0126 0125
   5 01 3036 0020 0020
   6 01 0020 0000 4000
   7 16 0021 3042 0133
0130 16 0021 3026 0135
   1 13 3012 0021 0021
   2 05 0000 0000 0136
   3 01 3026 0000 0021
   4 05 0000 0000 0136
   5 01 3012 0000 0021
   6 27 4001 0123 0137
   7 01 0017 0000 0021
0140 26 0000 0200 0000
   1 01 0021 0000 0020
   2 31 0014 0144 0143
   3 01 3036 0020 0020
   4 16 4000 0020 0146
   5 33 0000 0004 0000
   6 16 0021 3042 0152
   7 16 0021 3026 0154
0150 13 3012 0021 0021
   1 05 0000 0000 0155
   2 01 3026 0000 0021
   3 05 0000 0000 0155
   4 01 3012 0000 0021
   5 27 4001 0141 0156
   6 13 3012 0017 0017
   7 16 0017 3042 0161
0160 05 0000 0000 0121
   1 31 0014 0162 0164
   2 02 0000 0000 0014
   3 05 0000 0000 0120
   4 31 3000 0022 0117
   5 33 0000 0001 0077
   6 33 0000 0003 0177
```

Тесты арифметического устройства (АУ)

Примеры для проверки АУ

Двоичные коды	команды
$(0) \vee (1) = (1)$	0043—0045
$(01) \vee (0) = (01)$	0046—0053
$(1) \vee (1) = (1)$	0054—0056
$(1) \wedge (0) = (0)$	0057—0061

(0) ∧ (0) = (0)	0062—0064
(0) ∧ (10) = (0)	0065—0067
(1) ∧ (1) = (1)	0070—0072
(01) ≅ (10) = (0)	0073—0075
(10) ≅ (01) = (0)	0076—0100
(01) ≅ (01) = (1)	0101—0103
(10) ≅ (10) = (1)	0104—0107

Восьмеричные коды

05 (25) + 05 (25) = 12 (52)	0110—0112
(25) + 12 (52) = 05 (25)	0113—0115
07 5736 7573 6745 + 07 5736 7573 6745 = = 17 3675 7367 5712	0116—0120
05 (25) + 32 (52) = (25)	0121—0123
12 (52) + (25) = 05 (25)	0124—0126
00 0100 0400 2001 + 30 0040 0200 1000 = = 27 7737 7577 6777	0127—0131
20 2001 0004 0020 + 00 4002 0010 0040 = = 00 2001 0004 0020	0132—0134
1 (0) + (0) 1 = (0) 1	0135—0142
1 (0) + (0) 20 = 0 (20)	0143—0145
1 (0) + 00 0100 0400 2001 = = 00 0100 0400 2001	0146—0150
17 3675 7367 5712 + 17 3675 7367 5712 = = 16 7573 6757 3624	0151—0154
12 (52) — 12 (52) = 1 (0)	0155—0157
(25) — (25) = 1 (0)	0160—0162
12 (52) *СлК* (25) = 3 (7)	0163—0165
(0) *СлК* 02 (52) = 02 (52)	0166—0170
2 (0) 1 × 3 (0) = (0) 1	0171—0173
3 (0) 1 × 0 (7) = 3 (0)	0174—0176
0 (7) × (07) = 1 (7) 5	0177—0201
05 (25) × 32 (52) = 23 (43)	0202—0204
01 (0) : 03 (0) = 05 (25)	0205—0207
200020 (0) : 000 (16) 0 = (2) 00	0210—0212

Команды 0214—0226 — образование псевдослучайного **числа.**

» 0240—0245 — итерационное деление.

» 0256—0345 — константы.

( ) — означают число в периоде.

```
0001 25 5564 3455 4124
   2 12 5252 5252 5252
   3 25 2525 2525 2525
   4 05 2525 2525 2525
   5 32 5252 5252 5252
   6 07 5736 7573 6745
   7 17 3675 7367 5712
0010 00 0100 0400 2001
   1 30 0040 0200 1000
   2 20 2001 0004 0020
0013 00 4002 0010 0040
   4 00 2001 0004 0020
   5 00 0000 0000 0001
   6 00 0000 0000 0040
   7 27 7737 7577 6777
0020 16 7573 6757 3624
   1 20 0000 0000 0001
   2 30 0000 0000 0000
   3 30 0000 0000 0001
   4 17 7777 7777 7775
```

```
    5 23 4343 4343 4343
    6 01 0000 0000 0000
    7 03 0000 0000 0000
0030 20 0020 0000 0000
    1 00 0160 0000 0000
    2 22 2222 2222 2200
    3 27 7777 7777 7777
    4 01 0100 0000 0367
    5 26 0346 0000 0000
    6 07 4002 0367 0367
    7 27 4001 0036 0040
0040 16 0367 0001 0351
    1 22 0367 0367 0256
    2 01 3066 0000 0353
    3 14 0000 3036 0347
    4 16 0347 3036 0046
    5 33 0000 0000 0000
    6 14 0002 0000 0347
    7 16 0347 0002 0051
0050 33 0000 0000 0000
    1 14 0003 0000 0347
    2 16 0347 0003 0054
    3 33 0000 0000 0000
    4 14 3036 3036 0347
    5 16 0347 3036 0057
    6 33 0000 0000 0000
    7 15 3036 0000 0347
0060 16 0347 0000 0062
    1 33 0000 0000 0000
    2 15 0000 0000 0347
    3 16 0347 0000 0065
    4 33 0000 0000 0000
    5 15 0000 0003 0347
    6 16 0347 0000 0070
    7 33 0000 0000 0000
0070 15 3036 3036 0347
    1 16 0347 3036 0073
    2 33 0000 0000 0000
    3 17 0002 0003 0347
    4 16 0347 0000 0076
    5 33 0000 0000 0000
    6 17 0003 0002 0347
    7 16 0347 0000 0101
0100 33 0000 0000 0000
    1 17 0002 0002 0347
    2 16 0347 3036 0107
    3 33 0000 0000 0000
    4 17 0003 0003 0347
    5 16 0347 3036 0107
    6 33 0000 0000 0000
    7 31 3000 0043 0110
0110 01 0004 0004 0347
    1 16 0347 0002 0113
    2 33 0000 0000 0000
    3 01 0003 0002 0347
    4 16 0347 0004 0116
    5 33 0000 0000 0000
    6 01 0006 0006 0347
    7 16 0347 0007 0121

0120 33 0000 0000 0000
    1 01 0004 0005 0347
    2 16 0347 0003 0124
    3 33 0000 0000 0000
    4 01 0002 0003 0347
    5 16 0347 0004 0127
    6 33 0000 0000 0000
    7 01 0010 0011 0347
0130 16 0347 0017 0132
    1 33 0000 0000 0000
    2 01 0012 0013 0347
    3 16 0347 0014 0135
    4 33 0000 0000 0000
    5 01 3026 0015 0347
    6 16 0347 0015 0140
    7 33 0000 0000 0000
0140 01 3026 0016 0347
    1 16 0347 0016 0143
    2 33 0000 0000 0000
    3 01 3026 0010 0347
    4 16 0347 0010 0146
    5 33 0000 0000 0000
    6 01 3026 0013 0347
    7 16 0347 0013 0151
0150 33 0000 0000 0000
    1 07 0007 0007 0347
    2 16 0347 0020 0154
    3 33 0000 0000 0000
    4 31 3000 0110 0155
    5 02 0002 0002 0347
    6 16 0347 3026 0160
    7 33 0000 0000 0000
0160 06 0003 0003 0347
    1 16 0347 3026 0163
    2 33 0000 0000 0000
    3 03 0002 0003 0347
    4 16 0347 3036 0166
    5 33 0000 0000 0000
    6 13 0000 0261 0347
    7 16 0347 0261 0171
0170 33 0000 0000 0000
    1 11 0021 0022 0347
    2 16 0347 0015 0174
    3 33 0000 0000 0000
    4 11 0023 3035 0347
    5 16 0347 0022 0177
    6 33 0000 0000 0000
    7 11 3035 3035 0347
0200 16 0347 0024 0202
    1 33 0000 0000 0000
    2 11 0004 0005 0347
    3 16 0347 0025 0205
    4 33 0000 0000 0000
    5 12 0026 0027 0347
    6 16 0347 0004 0210
    7 33 0000 0000 0000
0210 12 0030 0031 0347
    1 16 0347 0032 0213
    2 33 0000 0000 0000
```

```
     3 05 0000 0000 0263
     4 01 0353 0000 0354
     5 10 0354 3042 0353
     6 31 0353 0221 0217
     7 02 0000 0353 0353
0220 01 0353 3042 0353
     1 15 0354 0260 0352
     2 12 0352 3044 0352
     3 15 0354 3044 0351
     4 05 0000 0351 0226
     5 02 0000 0352 0352
     6 17 0352 0353 0353
     7 11 0353 0354 0355
0230 11 0354 0353 0356
     1 16 0355 0356 0234
     2 22 0353 0356 0233
     3 33 0000 0000 0000
     4 06 0353 0000 0355
     5 05 0354 0355 0240
     6 06 0354 0000 0355
     7 01 0353 0000 0354
0240 01 0354 0000 0356
     1 02 3035 0355 0357
     2 11 0356 0357 0352
     3 01 0356 0352 0356
     4 11 0357 0357 0357
     5 05 0357 0000 0247
     6 05 0000 0000 0242
     7 12 0354 0355 0357
0250 02 0357 0356 0352
     1 05 0352 3002 0254
     2 22 0354 0357 0253
     3 33 0000 0000 0012
     4 31 3000 0214 0043
     5 31 3001 0214 0043
     6 33 0000 0000 0000
     7 05 0000 0000 0042
0260 01 7777 7777 7777
     1 02 5252 5252 5252
     2 05 2525 2525 2524
     3 35 0002 0350 0347
     4 16 0347 0002 0266
     5 33 0000 0000 0000
     6 16 0350 3026 0270
     7 33 0000 0000 0000
0270 35 0000 0350 0347
     1 16 0347 0000 0273
     2 33 0000 0000 0000
     3 16 0350 3026 0275
     4 33 0000 0000 0000
     5 35 3026 0350 0347
     6 16 0347 3026 0300
     7 33 0000 0000 0000
0300 16 0350 3026 0302
     1 33 0000 0000 0000
     2 35 3177 0350 0347
     3 16 0347 0335 0305
     4 33 0000 0000 0000
     5 16 0350 0336 0307
     6 33 0000 0000 0000
     7 13 0337 0000 0347
0310 16 0347 0000 0312
     1 33 0000 0000 0000
     2 13 0337 3026 0347
     3 16 0347 0000 0315
     4 33 0000 0000 0000
     5 13 0340 3026 0347
     6 16 0347 0343 0320
     7 33 0000 0000 0000
0320 13 0341 3177 0347
     1 16 0347 3026 0323
     2 33 0000 0000 0000
     3 13 0342 3177 0347
     4 16 0347 3012 0326
     5 33 0000 0000 0000
     6 13 0344 0003 0347
     7 16 0347 3026 0331
0330 33 0000 0000 0000
     1 13 0345 0003 0347
     2 16 0347 0000 0334
     3 33 0000 0000 0000
     4 05 0000 0000 0360
     5 30 0000 0000 0000
     6 20 0000 0000 0047
     7 00 0000 0000 0014
0340 20 0000 0000 0014
     1 00 0000 0000 0050
     2 20 0000 0000 0050
     3 00 0020 0000 0000
     4 20 0000 0000 0100
     5 20 0000 0000 0101
```

Тест печати

Команды 0001—0006 — контрольное суммирование теста.

» 0007—0013 — восьмеричная печать.

» 0014—0017 — десятичная печать.

После восьмеричной печати происходит останов; далее тумблер печати переключается на десятичную печать.

0001	26	0654	0001	—
2	07	4000	0654	0654
3	27	4001	0002	0004
4	16	0654	0655	0007
5	22	0654	0654	0006
6	33	—	—	—
7	22	0020	0307	0010
0010	31	3000	0007	0011
1	22	0310	0343	0012
2	31	3000	0013	0011
3	33	—	—	—
4	22	0344	0627	0015
5	31	3000	0014	0016
6	22	0630	0653	0017
7	31	3000	0006	0016
0020	01	—	—	—
1	—	1000	—	—
2	—	0100	—	—
3	—	0010	—	—
4	—	0001	—	—
5	—	0000	1000	—
6	—	—	0100	—
7	—	—	0010	—
0030	—	—	0001	—
1	—	—	—	1000
2	—	—	—	0100
3	—	—	—	0010
4	—	—	—	0001
5	31	—	—	—
6	30	1000	—	—
7	20	0100	—	—
0040	20	0010	—	—
1	20	0001	—	—
2	20	—	1000	—
3	20	—	0100	—
4	20	—	0010	—
5	20	—	0001	—
6	20	—	—	1000
7	20	—	—	0100
0050	20	—	—	0010
1	20	—	—	0001
2	02	—	—	—
3	00	2000	—	—
4	—	0200	—	—
5	—	0020	—	—
6	—	0002	—	—
7	—	—	2000	—
0060	—	—	0200	—
1	—	—	0020	—
2	—	—	0002	—
3	—	—	—	2000
4	—	—	—	0200
5	—	—	—	0020
6	—	—	—	0002
7	32	—	—	—
0070	30	2000	—	—
1	20	0200	—	—
2	20	0020	—	—
3	20	0002	—	—

0074	20	—	2000	—
5	20	—	0200	—
6	20	—	0020	—
7	20	—	0002	—
0100	20	—	—	2000
1	20	—	—	0200
2	20	—	—	0020
3	20	—	—	0002
4	03	—	—	—
5	—	3000	—	—
6	—	0300	—	—
7	—	0030	—	—
0110	—	0003	—	—
1	—	—	3000	—
2	—	—	0300	—
3	—	—	0030	—
4	—	—	0003	—
5	—	—	—	3000
6	—	—	—	0300
7	—	—	—	0030
0120	—	—	—	0003
1	33	—	—	—
2	30	3000	—	—
3	20	0300	—	—
4	20	0030	—	—
5	20	0003	—	—
6	20	—	3000	—
7	20	—	0300	—
0130	20	—	0030	—
1	20	—	0003	—
2	20	—	—	3000
3	20	—	—	0300
4	20	—	—	0030
5	20	—	—	0003
6	04	—	—	—
7	—	4000	—	—
0140	—	0400	—	—
1	—	0040	—	—
2	—	0004	—	—
3	—	—	4000	—
4	—	—	0400	—
5	—	—	0040	—
6	—	—	0004	—
7	—	—	—	4000
0150	—	—	—	0400
1	—	—	—	0040
2	—	—	—	0004
3	34	—	—	—
4	30	4000	—	—
5	20	0400	—	—
6	20	0040	—	—
7	—	—	—	—
0160	—	—	—	—
1	20	0004	—	—
2	20	—	4000	—
3	20	—	0400	—
4	20	—	0040	—
5	20	—	0004	—
6	20	—	—	4000

0167	20	—	—	0400
0170	20	—	—	0040
1	20	—	—	0004
2	05	—	—	—
3	—	5000	—	—
4	—	0500	—	—
5	—	0050	—	—
6	—	0005	—	—
7	—	—	5000	—
0200	—	—	0500	—
1	—	—	0050	—
2	—	—	0005	—
3	—	—	—	5000
4	—	—	—	0500
5	—	—	—	0050
6	—	—	—	0005
7	35	—	—	—
0210	30	5000	—	—
1	20	0500	—	—
2	20	0050	—	—
3	20	0005	—	—
4	20	—	5000	—
5	20	—	0500	—
6	20	—	0050	—
7	20	—	0005	—
0220	20	—	—	5000
1	20	—	—	0500
2	20	—	—	0050
3	20	—	—	0005
4	06	—	—	—
5	—	6000	—	—
6	—	0600	—	—
7	—	0060	—	—
0230	—	0006	—	—
1	—	—	6000	—
2	—	—	0600	—
3	—	—	0060	—
4	—	—	0006	—
5	—	—	—	6000
6	—	—	—	0600
7	—	—	—	0060
0240	—	—	—	0006
1	36	—	—	—
2	30	6000	—	—
3	20	0600	—	—
4	20	0060	—	—
5	20	0006	—	—
6	20	0000	6000	—
7	20	—	0600	—
0250	20	—	0060	—
1	20	—	0006	—
2	20	—	—	6000
3	20	—	—	0600
4	20	—	—	0060
5	20	—	—	0006
6	07	—	—	—
7	—	7000	—	—
0260	—	0700	—	—
1	—	0070	—	—

0262	—	0007	—	—
3	—	—	7000	—
4	—	—	0700	—
5	—	—	0070	—
6	—	—	0007	—
7	—	—	—	7000
0270	—	—	—	0700
1	—	—	—	0070
2	—	—	—	0007
3	37	—	—	—
4	30	7000	—	—
5	20	0700	—	—
6	20	0070	—	—
7	20	0007	—	—
0300	20	—	7000	—
1	20	—	0700	—
2	20	—	0070	—
3	20	—	0007	—
4	20	—	—	7000
5	20	—	—	0700
6	20	—	—	0070
7	20	—	—	0007
0310	00	1234	5670	1234
1	04	0123	4567	0123
2	03	4012	3456	7012
3	02	3401	2345	6701
4	01	2340	1234	5670
5	00	1234	0123	4567
6	07	0123	4012	3456
7	06	7012	3401	2345
0320	05	6701	2340	1234
1	04	5670	1234	0123
2	03	4567	0123	4012
3	02	3456	7012	3401
4	01	2345	6701	2345
5	—	—	—	—
6	20	1234	5670	1234
7	34	0123	4567	0123
0330	23	4012	3456	7012
1	32	3401	2345	6701
2	21	2340	1234	5670
3	30	1234	0123	4567
4	27	0123	4012	3456
5	36	7012	3401	2345
6	25	6701	2340	1234
7	34	5670	1234	0123
0340	23	4567	0123	4012
1	32	3456	7012	3401
2	21	2345	6701	2340
3	37	7777	7777	7777
4 +	10	—	—	—
5 +	01	—	—	—
6 +	—	1000	—	
7 +	—	0100	—	
0350 +	—	0010	—	
1 +	—	0001	—	
2 +	—	—	1000	
3 +	—	—	0100	
4 +	—	—	0010	

```
0355 + — — 0001
   6 — 10 — —
   7 — 01 — —
0360 — — 1000 —
   1 — — 0100 —
   2 — — 0010 —
   3 — — 0001 —
   4 — — — 1000
   5 — — — 0100
   6 — — — 0010
   7 — — — 0001
0370 + 20 — —
   1 + 02 — —
   2 + — 2000 —
   3 + — 0200 —
   4 + — 0020 —
   5 + — 0002 —
   6 + — — 2000
   7 + — — 0200
0400 + — — 0020
   1 + — — 0002
   2 — 20 — —
   3 — 02 — —
   4 — — 2000 —
   5 — — 0200 —
   6 — — 0020 —
   7 — — 0002 —
0410 — — — 2000
   1 — — — 0200
   2 — — — 0020
   3 — — — 0002
   4 + 30 — —
   5 + 03 — —
   6 + — 3000 —
   7 + — 0300 —
0420 + — 0030 —
   1 + — 0003 —
   2 + — — 3000
   3 + — — 0300
   4 + — — 0030
   5 + — — 0003
   6 — 30 — —
   7 — 03 — —
0430 — — 3000 —
   1 — — 0300 —
   2 — — 0030 —
   3 — — 0003 —
   4 — — — 3000
   5 — — — 0300
   6 — — — 0030
   7 — — — 0003
0440 + 40 — —
   1 + 04 — —
   2 + — 4000 —
   3 + — 0400 —
   4 + — 0040 —
   5 + — 0004 —
   6 + — — 4000
   7 + — — 0400
```

```
0450 + — — 0040
   1 + — — 0004
   2 — 40 — —
   3 — 04 — —
   4 — — 4000 —
   5 — — 0400 —
   6 — — 0040 —
   7 — — 0004 —
0460 — — — 4000
   1 — — — 0400
   2 — — — 0040
   3 — — — 0004
   4 + 50 — —
   5 + 05 — —
   6 + — 5000 —
   7 + — 0500 —
0470 + — 0050 —
   1 + — 0005 —
   2 + — — 5000
   3 + — — 0500
   4 + — — 0050
   5 + — — 0005
   6 — 50 — —
   7 — 05 — —
0500 — — 5000 —
   1 — — 0500 —
   2 — — 0050 —
   3 — — 0005 —
   4 — — — 5000
   5 — — — 0500
   6 — — — 0050
   7 — — — 0005
0510 + 60 — —
   1 + 06 — —
   2 + — 6000 —
   3 + — 0600 —
   4 + — 0060 —
   5 + — 0006 —
   6 + — — 6000
   7 + — — 0600
0520 + — — 0060
   1 + — — 0006
   2 — 60 — —
   3 — 06 — —
   4 — — 6000 —
   5 — — 0600 —
   6 — — 0060 —
   7 — — 0006 —
0530 — — — 6000
   1 — — — 0600
   2 — — — 0060
   3 — — — 0006
   4 + 70 — —
   5 + 07 — —
   6 + — 7000 —
   7 + — 0700 —
0540 + — 0070 —
   1 + — 0007 —
   2 + — — 7000
```

```
  3 + — —  0700
  4 + — —  0070
  5 + — —  0007
  6 — 70 —  —
  7 — 07 —  —
0550 — — 7000  —
  1 — — 0700  —
  2 — — 0070  —
  3 — — 0007  —
  4 — —  — 7000
  5 — —  — 0700
  6 — —  — 0070
  7 — —  — 0007
0560 + 80 —  —
  1 + 08 —  —
  2 + — 8000  —
  3 + — 0800  —
  4 + — 0080  —
  5 + — 0008  —
  6 + —  — 8000
  7 + —  — 0800
0570 + —  — 0080
  1 + —  — 0008
  2 — 80 —  —
  3 — 08 —  —
  4 — — 8000  —
  5 — — 0800  —
  6 — — 0080  —
  7 — — 0008  —
0600 — —  — 8000
  1 — —  — 0800
  2 — —  — 0080
  3 — —  — 0008
  4 + 90 —  —
  5 + 09 —  —
  6 + — 9000  —
  7 + — 0900  —
```

```
0610 + — 0090  —
  1 + — 0009  —
  2 + —  — 9000
  3 + —  — 0900
  4 + —  — 0090
  5 + —  — 0009
  6 — 90 —  —
  7 — 09 —  —
0620 — — 9000  —
  1 — — 0900  —
  2 — — 0090  —
  3 — — 0009  —
  4 — —  — 9000
  5 — —  — 0900
  6 — —  — 0090
  7 — —  — 0009
0630 + 01 2345 6789
  1 + 90 1234 5678
  2 + 89 0123 4567
  3 + 78 9012 3456
  4 + 67 8901 2345
  5 + 56 7890 1234
  6 + 45 6789 0123
  7 + 34 5678 9012
0640 + 23 4567 8901
  1 + 12 3456 7890
  2 — 01 2345 6789
  3 — 90 1234 5678
  4 — 89 0123 4567
  5 — 78 9012 3456
  6 — 67 8901 2345
  7 — 56 7890 1234
0650 — 45 6789 0123
  1 — 34 5678 9012
  2 — 23 4567 8901
  3 — 12 3456 7890
  4 — —  —  —
  5 — —  —  —
```

Тест магнитного барабана (МБ)

Команды 0001—0006 — восстановление команд обращения к МБ.
» 0007—0010 — счетчик сдвигов по МБ.
» 0011—0015 — запись кодов в ОЗУ.
» 0016—0021 — формирование команд обращения к МБ.
» 0024—0030 — запись на МБ и чтение.
» 0042—0047 — контрольное суммирование теста.

```
 Вв 21 0001 0062 0000
0001 01 3200 0000 0032
   2 01 0051 0000 0030
   3 01 0052 0000 0024
   4 01 0053 0000 0025
   5 01 0054 0000 0026
   6 01 0055 0000 0027
   7 01 3011 0000 0056
0010 01 3011 0000 0057
   1 10 3000 3040 0060
   2 34 0057 0000 0000
   3 01 3007 0000 4062
```

```
0014 01 3011 0057 0057
   5 05 0057 3000 0012
   6 03 3002 0024 0024
   7 03 3002 0026 0026
0020 03 3000 0025 0025
   1 03 3000 0027 0027
   2 12 3000 3040 0061
   3 03 0061 0030 0030
   4 00 0000 0000 0000
   5 00 0000 0000 0000
   6 00 0000 0000 0000
   7 00 0000 0000 0000
```

```
0030 00 0000 0000 0000
   1 16 4063 5031 0033
   2 33 0000 0000 0000
   3 27 4001 0031 0034
   4 03 3012 0032 0032
   5 03 0060 0024 0024
   6 03 0060 0026 0026
   7 01 3011 0056 0056
0040 05 0056 3001 0024
   1 31 0000 0001 0000
   2 01 0000 0000 0062
   3 26 0062 0000 0000
   4 07 4001 0062 0062
   5 27 4001 0044 0046
0046 22 0062 0062 0047
   7 33 0000 0000 0000
0050 31 0000 0001 0000
   1 26 0000 0000 0000
   2 25 0001 0000 0000
   3 23 0063 0062 0026
   4 25 0000 0000 0000
   5 24 1031 1030 0030
   6 00 0000 0000 0000
   7 00 0000 0000 0000
0060 00 0000 0000 0000
   1 00 0000 0000 0000
   2 00 0000 0000 0000
```

Система команд машины «Киев»

Арифметические и логические операции

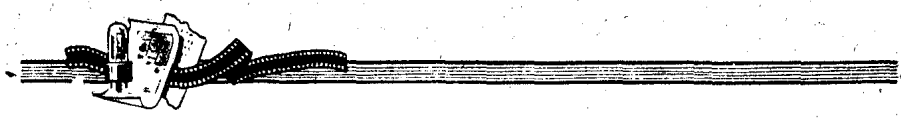
КОп	Название операции	Возможность модификации адресов	Аварийный останов
01	Сложение	Да	|Рез| ⩾ 1
02	Вычитание	»	|Рез| ⩾ 1
03	Сложение команд	»	—
06	Вычитание модулей	»	—
07	Циклическое сложение	»	—
10	Умножение без округления	»	—
11	Умножение с округлением	»	—
12	Деление	»	—
13	Сдвиг логический	»	—
14	Логическое сложение	»	—
15	Логическое умножение	»	—
17	Неравнозначность	»	—
35	Нормализация	»	—

Операции управления

КОп	Проверяемое условие	В случае выполнения	В случае невыполнения	'A	'Ц	'P
04	$'('A_1 + 'E_1'A) \leqslant '('A_2 + 'E_2'A)$	ПУ $'A_3 + 'E_3'A$	$'C+1 \Rightarrow C$	Не изменяется	Не изменяется	Не изменяется
05	$\lvert'('A_1 + 'E_1'A)\rvert \leqslant \lvert'('A_2 + 'E_2'A)\rvert$	ПУ $'A_3 + 'E_3'A$	$'C+1 \Rightarrow C$	То же	То же	То же
16	$'(A_1 + 'E_1'A) = '('A_2 + 'E_2'A)$	ПУ $'A_3 + 'E_3'A$	$'C+1 \Rightarrow C$	» »	» »	» »
31	$'('A_1 + 'E_1'A) \leqslant -0$	ПУ $'A_3 + 'E_3'A$	ПУ $'A_2 + 'E_2'A$	» »	» »	» »
30	$'('A_1 + 'E_1'A) \leqslant -0$	ПУ $'A_3 + 'E_3'A$	$'C+1 \Rightarrow C$	» »	» »	$'A_2 \Rightarrow P$
32	—	ПУ $'P$	—	» »	» »	$0 \Rightarrow P$
26	$'A = 'Ц$	ПУ $'A_3$	$'C+1 \Rightarrow C$	$'A_2 \Rightarrow A$	$'A_1 \Rightarrow Ц$	Не изменяется
27	$'A = 'Ц$	ПУ $'A_3$	ПУ $'A_2$	$'A + 'A_1 \Rightarrow A$	Не изменяется	То же
34	—	$''A_1 \Rightarrow A_3$	—	$'A_1 \Rightarrow A$	То же	» »

Операции обращения к ВУ

КОп		КОп	
20	Ввод с переводом чисел в ячейки от $'A_1$ до $'A_2$	23	Запись на МБ
21	Ввод без перевода чисел в ячейки от $'A_1$ до $'A_2$	23	Чтение с МБ
22	Вывод кодов из ячеек от $'A_1$ до $'A_2$	25	Подготовительная операция для МБ
		33	Останов



ЛИТЕРАТУРА

1. Глушков В. М., Об оптимальном объеме оперативных запоминающих устройств, ДАН УССР, 1960, № 5.

2. Глушков В. М., Два универсальных критерия эффективности вычислительных машин, ДАН УССР, 1960, № 4.

3. Глушков В. М., Керуючі машини автоматизованого виробництва, Товариство по розповсюдженню політичних і технічних знань, Київ, 1960.

4. Гнеденко Б. В., Глушков В. М., Ющенко К. Л., Математичні параметри універсальної цифрової машини «Київ», Збірник праць ОЦ АН УРСР, т. II, Київ, 1961.

5. Гнеденко Б. В., Королюк В. С., Ющенко Е. Л., Элементы программирования, Физматгиз. М., 1961.

6. Дашевський Л. Н., Погребінський С. Б., Шкабара К. О., Структурна схема та основні принципи побудови цифрової автоматичної машини «Київ», Збірник праць ОЦ АН УРСР, т. II, Київ, 1961.

7. Ершов Е. П., Программирующая программа для БЭСМ, М., Изд-во АН СССР, 1958.

8. Іваненко Л. М., Ющенко К. Л., Основні питання побудови програмуючої програми для машини «Київ», Збірник праць ОЦ АН УРСР, т. II, «Київ», 1961.

9. Королюк В. С., Шкабара К. О., Ющенко К. Л., Групові операції машини «Київ», Збірник праць ОЦ АН УРСР, т. II, Київ, 1961.

10. Королюк В. С., Ющенко К. Л., Питання теорії і практики програмування, Збірник праць ОЦ АН УРСР, т. I, Київ, 1961.

11. Летічевський О. А., Еквівалентність в одному класі адресних алгоритмів, Збірник праць ОЦ АН УРСР, т. I, Київ, 1961.

12. Система стандартных подпрограмм, Сборник под ред. Шура-Бура М. Р., Физматгиз. М., 1959.

13. Сообщение об алгоритмическом языке Алгол, Изд-во АН СССР, М., 1960.

14. Фаддеев Д. К., Фаддеева В. Н., Вычислительные методы линейной алгебры, Физматгиз, М., 1960.

15. Ющенко Е. Л., Адресное программирование, Заочный семинар «Кибернетика на транспорте», КДНТП, 1962.

16. Ющенко К. Л., Адресні алгоритми та цифрові математичні машини, Збірник праць ОЦ АН УРСР, т. II, Київ, 1961.

17. Ющенко К. Л., Адресні алгоритми та цифрові автоматичні машини, ДАН УРСР, т. VI, Київ, 1962.

18. Ющенко К. Л., Рівні та стилі адресної мови та проблема автоматизації програмування, ДАН УРСР, т. VII, Київ, 1962.

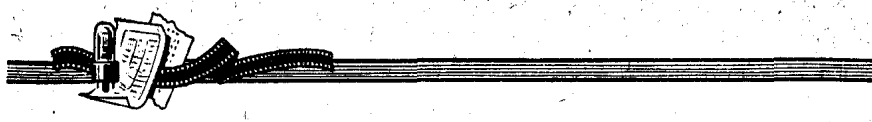
19. Ющенко К. П., Бистрова Л. П., Програмуюча програма, інформацією для якої служить адресний алгоритм, Збірник праць ОЦ АН УРСР, т. III, Київ, 1961.

20. Ющенко К. Л., Дрючина М. О., Бібліотека підпрограм машини «Київ», Збірник праць ОЦ АН УРСР, т. II, Київ, 1961.

21. Ющенко К. Л., Костюченко О. І., Алгоритм перекладу дужкового запису формул у бездужковий запис Лукасевича, Збірник праць ОЦ АН УРСР, т. III, Київ, 1961.

22. Ющенко К. Л., Михайлова О. І., Алгоритми формальної перевірки правильності дужкового та бездужкового запису формул з одно- і двомісцевими операціями, Збірник праць ОЦ АН УРСР, т. III, Київ, 1961.

23. Яблонский С. В., Труды математического института им. Стеклова, т. I, АН СССР, М., 1958.



ОГЛАВЛЕНИЕ

Глушков Виктор Михайлович
(акад. АН УССР)
Ющенко Екатерина Логвиновна
(канд. физ.-мат. наук)

ВЫЧИСЛИТЕЛЬНАЯ МАШИНА „КИЕВ"
(МАТЕМАТИЧЕСКОЕ ОПИСАНИЕ)

Редактор издательства *О. А. Немчукова*
Переплет художника *Л. Б. Сергия*
Технический редактор *С. М. Шафета*
Корректор *Г. В. Зеленина*

Сдано в набор 13.VI.1962 г. Подписано к печати 12/X 1962 г. Формат бумаги $60\times90/_{16}$ Объем: 11,5 физич. лист., 11,5 условн. лист., 12,05 учетно-издат. лист. Тираж 8000. БФ 37755. Цена 75 коп.

Государственное издательство технической литературы УССР
Киев, 4, Пушкинская, 28

Отпечатано с матриц Книжной ф-ки им. Фрунзе Главполиграфиздата Министерства культуры УССР, Харьков, Донец-Захаржевская, 6/8, в типографии «Коммунист» Главполиграфиздата Министерства культуры УССР, Харьков, Пушкинская, 29. Зак. 822.