ОГЛАВЛЕНИЕ

ББК 32.97
П78
УДК 681.31

Авторы: **Я. К. Трохименко, В. П. Захаров, Н. П. Ромашко, В. А. Жижко, Ю. М. Польский**

Рецензент докт. техн. наук, проф. В. П. Дьяконов

Редакция литературы по информатике и вычислительной технике

П78 **Программируемые** микрокалькуляторы: Устройство и пользование / Я. К. Трохименко, В. П. Захаров, Н. П. Ромашко и др.; Под ред. Я. К. Трохименко. — М.: Радио и связь, 1990. — 272 с., ил.

ISBN 5-256-00318-6.

Описаны структура, архитектура, элементная база и операционное программное обеспечение массовых программируемых микрокалькуляторов (ПМК) серии «Электроника» (МК-46, МК-64, БЗ-34, БЗ-54, МК-56, МК-54, МК-61, МК-52), а также перспективные ПМК «Электроника-85» и «Электроника МК-72» и тенденции их дальнейшего совершенствования. Рассмотрены особенности программирования ПМК, даны рекомендации по оптимизации программ, оценке погрешностей вычислений и другие полезные сведения.

Для пользователей и разработчиков ПМК, а также специалистов, использующих элементную базу ПМК.

П $\frac{2404060000\text{-}010}{046(01)\text{-}90}$ 145-90 ББК 32.97

ISBN 5-256-00318-6

ПРЕДИСЛОВИЕ

Программируемые микрокалькуляторы (ПМК) завоевали признание широкого круга пользователей в учебных заведениях и различных отраслях народного хозяйства как достаточно мощные индивидуальные вычислительные устройства, обеспечивающие возможность решения повседневных задач практически в любых условиях. В связи с простотой эксплуатации, надежностью и низкой стоимостью машинного времени ПМК незаменимы во всех случаях, когда решение задачи на ЭВМ большей производительности экономически нецелесообразно или практически неприемлемо.

Решение разнообразных задач с помощью ПМК рассмотрено в ряде книг [1, 2, 5, 6, 8, 14—27] и других публикациях. В настоящей книге кроме общих сведений о ПМК, освоенных промышленностью, достаточно подробно описаны особенности их структуры, элементной базы, архитектуры, прикладного программирования и перспектив развития. Эти сведения представляют интерес для пользователей, стремящихся расширить свои знания о ПМК для их более эффективного использования, а также для многочисленных радиолюбителей и специалистов различного профиля, которые могут применять достаточно надежную, малогабаритную и недорогую элементную базу ПМК для создания различных вычислительных или управляющих устройств.

Авторы книги стремились изложить материал на уровне предполагаемого круга читателей, ограничивая число используемых терминов и давая их определения. Насколько удачным окажется выбранный стиль изложения, могут судить читатели, все замечания которых будут с благодарностью приняты авторами.

Глава 1

ОСНОВНЫЕ ОСОБЕННОСТИ ПРОГРАММИРУЕМЫХ МИКРОКАЛЬКУЛЯТОРОВ

1.1. ЭВМ И МИКРОКАЛЬКУЛЯТОРЫ

Одной из основных характеристик ЭВМ является производительность, которая определяется емкостью памяти, обеспечивающей решение сложных задач без разбиения на более простые, а также быстродействием стандартных операций внутреннего обмена информацией. Производительность зависит от трудовых или эквивалентных денежных затрат на разработку, изготовление и применение ЭВМ. Практически их оценивают по стоимости часа машинного времени, равной отношению всех затрат на приобретение и эксплуатацию ЭВМ к числу часов ее работы за период амортизации. Стоимость машинного времени ЭВМ примерно пропорциональна ее производительности.

Экономическая эффективность использования ЭВМ повышается при специализации. Поэтому созданы универсальные ЭВМ, предназначенные для решения разнообразных задач обработки информации по легко сменяемым прикладным программам, и специализированные, предназначенные в основном для управления производственными и другими процессами. Применение ЭВМ определенной производительности экономически оправдано лишь в том случае, если рассматриваемую задачу нельзя решить с меньшими затратами на ЭВМ меньшей производительности или без применения ЭВМ.

В связи с экономическими требованиями в каждом поколении создаются ЭВМ различной производительности — от суперЭВМ, предназначенных для решения наиболее сложных задач, до простейших ЭВМ с минимальной стоимостью машинного времени. Так, еще в первом (ламповом) поколении наряду с ЭВМ относительно высокой производительности были созданы настольные ЭВМ, предназначенные для автоматизации выполнения арифметических операций над числами по командам, подаваемым нажатием клавиш. Такие ЭВМ в соответствии с их назначением были названы калькуляторами *.

*Следует отметить, что английские глаголы to compute и to calculate одинаково переводят на русский язык — считать, рассчитывать, вычислять, но второй из этих глаголов (связанный с латинским словом calculus, обозначавшим в древнем Риме «счетный камешек») по смыслу ближе к слову «считать». Кстати,

Во втором (транзисторном) поколении появились настольные калькуляторы, названные электронными клавишными вычислительными машинами (ЭКВМ), которые вычисляли и элементарные функции. Важное значение имело создание программируемых ЭКВМ, выполнявших вычисления по прикладным программам, предварительно введенным в память. Характерными их особенностями являлись однострочный индикатор для отображения результатов вычислений в виде отдельных чисел, ограниченная емкость памяти и отсутствие накопителей информации для длительного хранения.

Разработка микросхем с большой степенью интеграции (число элементов на единицу площади кристалла) — БИС обусловила возможность существенного уменьшения размеров и повышения производительности ЭВМ. Эти возможности могли быть реализованы в первую очередь при изготовлении ЭКВМ в связи с их относительной простотой и, следовательно, большей гибкостью производства. Поэтому впервые массовое применение БИС и началось в производстве малогабаритных ЭКВМ.

Одна из наиболее крупных в мире производителей малогабаритных ЭВМ фирма Hewlett-Packard (США) в 1972 г. приступила к серийному выпуску миниатюрных непрограммируемых ЭКВМ, получивших название карманных калькуляторов или микрокалькуляторов, а уже в следующем году начала серийное производство карманных программируемых микрокалькуляторов (ПМК).

Создание микрокалькуляторов знаменовало появление принципиально нового класса носимых вычислительных средств. Массовое производство непрограммируемых и особенно программируемых микрокалькуляторов обеспечило возможность автоматизации широкого круга повседневных вычислительных задач и существенного повышения производительности труда их пользователей. При этом ПМК, как и настольные программируемые ЭКВМ, отличаются от универсальных ЭВМ других классов в основном лишь малой производительностью, связанной с низкой стоимостью машинного времени. Последнее обстоятельство определяет и экономически оправданную область применения ПМК — относительно несложные вычислительные задачи, решение которых на универсальных ЭВМ других классов связано с большими затратами вследствие большей стоимости машинного времени и затратами рабочего времени пользователя на доступ к ЭВМ и ожидание результата решения задачи.

В соответствии с назначением ПМК основным требованием является низкая стоимость, обеспечивающая доступность широкому кругу пользователей. Это требование приводит к жестким ограничениям на аппаратурные затраты, определяемые количеством и стоимостью микросхем. Так как стоимость микросхем зависит от их технологии, то в

в отечественной литературе первые ЭВМ называли электронными счетными машинами (что сохранилось в названиях ЭВМ серии БЭСМ), хотя, строго говоря, их следовало назвать логическими машинами, так как автоматизация вычислений основаиа на логических операциях.

ПМК первых поколений, как правило, использовались микросхемы, изготовленные по относительно недорогой МОП-технологии. Дальнейшее развитие полупроводниковой технологии привело к существенному снижению стоимости микросхем при их массовом изготовлении. Поэтому в наиболее совершенных и дорогих ПМК стали использовать микросхемы, изготовленные по другим технологиям, и в первую очередь по КМОП-технологии, особенностью которой являются дополнительные (комплементарные) транзисторные структуры с различным типом проводимости. Микросхемы, изготовленные по этой технологии, хотя и дороже микросхем, изготовленных по МОП-технологии, но имеют большее быстродействие и потребляют значительно меньше энергии. Последнее особенно существенно для носимых ПМК с автономным питанием от аккумуляторов или сухих элементов.

Повышение степени интеграции микросхем обеспечило также создание полупроводниковых запоминающих устройств (ЗУ) большой емкости и энергонезависимых ЗУ, сохраняющих информацию при выключенном питании. Это позволило значительно расширить емкость памяти ПМК и комплектовать их встроенными или внешними накопителями информации на микросхемах, а также сменными модулями для постоянного или временного хранения информации. В карманных ПМК широкое применение нашли также экономичные жидкокристаллические индикаторы (ЖКИ).

Развитие ПМК связано с совершенствованием ЭВМ других классов. С разработкой микропроцессорных наборов (комплектов совместно работающих микросхем) упростилось создание прежде всего недорогих управляющих микроЭВМ (микроконтроллеров). Так как управляемые физические величины нельзя измерить с большой точностью, то быстродействие контроллеров повышают, обрабатывая небольшие порции информации (слова) и используя упрощенные вычислительные процедуры. Такие управляющие ЭВМ с «короткими» словами и процедурами получили название мини-ЭВМ, непосредственно не связанное вопреки существующему мнению с размерами ЭВМ. Простейшие мини-ЭВМ собирают из микропроцессорных наборов или изготовляют в виде однокристальных БИС (используемых и в ПМК), называемых иногда микромини-ЭВМ — первый префикс (микро) указывает на технологию изготовления, а второй (мини) — на особенности обработки информации.

Развитие микроэлектроники повлияло и на развитие универсальных ЭВМ высокой производительности. Стационарные ЭВМ этого класса в связи с высокой стоимостью машинного времени окупаются лишь при практически непрерывном решении достаточно сложных задач. Поэтому такие ЭВМ первых поколений работали в пакетном режиме, при котором последовательно решаются задачи по заранее заготовленному и отлаженному запасу (пакету) программ.

При работе в пакетном режиме отдельные задачи решаются быстро, но время ожидания очереди может оказаться большим. Поэтому в современных стационарных ЭВМ высокой производительности использу-

ют режим разделения времени между множеством устройств ввода-вывода информации — терминалами. Еще более эффективными оказались иерархические вычислительные системы с центральной ЭВМ высокой производительности и «интеллектуальными» терминалами в виде ЭВМ меньшей производительности, которые, в свою очередь, могут работать с несколькими терминалами пользователей. Развитием таких иерархических систем явились сети ЭВМ различной производительности с обменом информацией по линиям связи, которыми при определенных условиях могут служить телефонные или телеграфные линии.

Максимальная эффективность иерархических вычислительных систем достигается при возможности обращения пользователя к терминальной ЭВМ с достаточно низкой стоимостью машинного времени и достаточной мощностью для решения большинства задач пользователя, который при необходимости решения более сложной задачи может обратиться через сеть к ЭВМ большей производительности.

Внедрение БИС, обеспечившее существенное повышение производительности настольных ЭКВМ и миниатюризацию универсальных ЭВМ высокой производительности, привело к разработке индивидуальных ЭВМ, которые могут использоваться и в качестве терминалов вычислительной сети. В зависимости от производительности, стоимости и назначения такие микроЭВМ, получившие название персональных ЭВМ (ПЭВМ), можно условно разделить на профессиональные, конторские или учрежденческие, бытовые и карманные.

Профессиональные ПЭВМ (например, ЕС-1840, ЕС-1841, «Нейрон И9.66», «Электроника-85»), предназначенные для решения наиболее сложных задач квалифицированными специалистами, по производительности не уступают многим стационарным универсальным ЭВМ высокой производительности третьего поколения. Они снабжены комплектом достаточно совершенных внешних устройств, включая накопители информации большой емкости на жестких магнитных дисках, печатающие устройства и графопостроители для документирования результатов обработки информации, цветные или монохроматические мониторы на электронно-лучевой трубке для отображения текстов прикладных программ, результатов диагностирования работы, а также результатов обработки информации в алфавитно-цифровой или графической форме.

Конторские или учрежденческие ПЭВМ (например, ДВК-2, «Электроника Д3-28»), предназначенные для решения более простых задач обработки информации и обучения основам программирования в учебных заведениях, комплектуются менее дорогими внешними устройствами (в частности, накопителями информации на мягких магнитных дисках или магнитной ленте) и по стоимости значительно отличаются от профессиональных ПЭВМ.

Бытовые ПЭВМ (например, ПК-1 или БК-0010) предназначены для использования в основном в домашних условиях совместно с бытовыми телевизором в качестве многострочного монитора и магнитофоном в качестве накопителя информации большой емкости. Такие

ПЭВМ обычно конструктивно оформляются в виде плоской коробки с клавиатурой и разъемами для присоединения внешних устройств, комплект которых, кроме бытовых приборов, может дополнительно содержать различные приспособления для преобразования и отображения информации.

Малогабаритные носимые (карманные) ПЭВМ по назначению совпадают с ПМК и чаще всего отличаются от них многострочным жидкокристаллическим индикатором и особенностями программного представления способа решения задачи. Однако по мере совершенствования ПМК эти различия практически стерлись. Более того, на элементной базе карманных ПМК разработаны настольные с разъемами для присоединения бытовых телевизора и магнитофона, являющихся отличительной особенностью большинства бытовых ПЭВМ.

Таким образом, уже сейчас нельзя провести четкую границу между ПМК и простейшими ПЭВМ. В некоторых случаях различия между ними связаны с использованием микропроцессорных наборов для ПЭВМ и заказных БИС для ПМК, но все больше малогабаритных ПЭВМ собираются на заказных БИС, в связи с чем и это различие стирается. По этой причине многие малогабаритные ПЭВМ называют программируемыми микрокалькуляторами, и в будущем, вероятно, это название сохранится за всеми малогабаритными ПЭВМ. Во всяком случае распространенное сокращение ПМК в равной мере применимо как к программируемым микрокалькуляторам, так и к персональным микрокомпьютерам, внешне отличающимся в основном лишь входными языками.

1.2. ОСОБЕННОСТИ ВХОДНЫХ ЯЗЫКОВ

Решение на ЭВМ любой задачи преобразования информации возможно лишь после ее решения человеком и описания им способа решения задачи программой работы ЭВМ, составленной на «понятном» машине языке. Способ решения задачи отображают *алгоритмом*, состоящим из последовательности описаний операций, приводящих к искомому результату. Алгоритм представляют словесно-формульными описаниями, схемами и программами.

Словесно-формульное описание алгоритма состоит из последовательности однозначных словесно-формульных описаний отдельных операций (шагов алгоритма), обозначаемых порядковыми номерами. Описания операций в общем случае называют предписаниями, инструкциями или операторами (правилами выполнения операций, обозначенными специальными символами). Несмотря на разнообразие операций, выполняемых при решении прикладных задач, все их описания относятся к одному из двух основных типов операторов — присваивания или условных переходов.

Операторы присваивания, отображающие операции с однозначным результатом, описывают формулой

«пусть $I := A$».

В арифметических (вычислительных) операторах присваивания символом I отображают имя (идентификатор) переменной, значение которой после выполнения операции равно числу A, значению переменной A или результату вычислений по выражению A. Символ присваивания $:=$ часто заменяют знаком равенства, который в этом случае означает «равно после выполнения операции» *. Так, арифметический оператор «пусть $x = x-1$» означает, что в дальнейшем переменной x присваивается значение, на единицу меньшее предыдущего, а оператор «пусть $y = a - b \ln c$» означает, что переменной y присваивается значение, равное результату вычислений в правой части формулы после замены имен переменных их числовыми значениями. Ключевое слово «пусть» может быть заменено другим (например, «примем» или «вычислить») или опущено.

Операторы присваивания могут быть также логическими (например, «пусть x — не x») или отображать другие однозначные изменения состояния. Такие операторы часто описывают словами естественного языка, но могут быть записаны и в стандартной форме оператора присваивания. Так, оператор безусловного перехода «перейти к шагу m» равнозначен оператору «пусть шаг $= m$», а оператор прекращения выполнения алгоритма («стоп», «конец», «прекратить вычисления» и т. п.) по договоренности можно заменить, например, оператором «пусть $A = 0$», где символ A, отображающий состояние процесса, называют *флагом*.

Операторы условных переходов отображают операцию выбора следующего шага алгоритма в зависимости от выполнения заданного условия. Основная форма этого оператора: «если <условие>, то перейти к шагу ..., иначе — к шагу ...». Если при неудовлетворении проверяемого условия должен выполняться следующий по порядку шаг алгоритма, то обычно используют сокращенную форму оператора условного перехода

«если <условие>, то перейти к шагу ...».

Операторы, являющиеся элементами (словами) алгоритма, имеют самостоятельный смысл и различаются семантическим (смысловым) уровнем, соответствующим сложности выполняемой операции. Так, оператор «пусть $X = A+B\times C$» отличается более высоким уровнем по отношению к операторам равнозначной последовательности «пусть $X = B\times C$», «пусть $X = X + A$». Однако каждый из этих простых операторов является сложным по отношению к логическим операторам, последовательности которых соответствуют выполнению арифметических операций. Уровень сложности операторов должен выбираться наиболее высоким из выполнимых исполнителем алгоритма — неоправданное дробление операторов на более простые усложняет запись и вы-

*Символ присваивания $:=$ не следует смешивать со знаком определения $::=$ (который также часто заменяют знаком равенства), означающим в семантических формулах «равно по определению». Например, выражение $I ::=$ <Буква|Цифра> означает, что имя переменной I состоит из буквы и следующей за ней цифры.

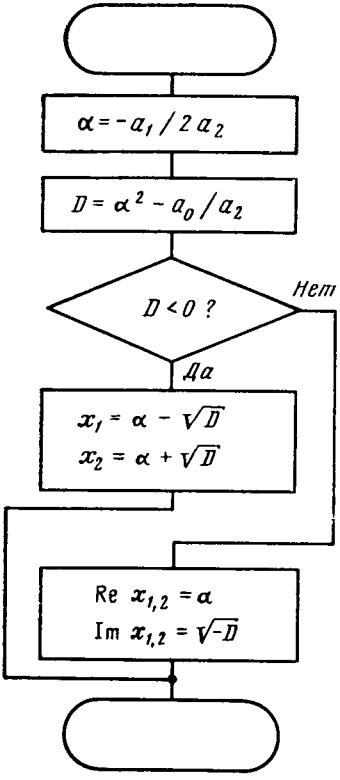


Рис. 1.1

полнение алгоритма. Так, алгоритм решения квадратного уравнения для подготовленного исполнителя достаточно представить одним оператором «решить уравнение $a_2x^2+a_1x+a_0 = 0$», для менее подготовленного исполнителя, например, в виде формулы вычисления корней «пусть $x_{1,2} = a_1/2a_2 \pm \sqrt{(a_1/2a_2)^2 - a_0/a_2}$». Если же исполнитель не умеет находить по этой формуле все возможные решения, то ее необходимо заменить более подробным словесно-формульным описанием, например:

1. Пусть $\alpha = -a_1/2a_2$.
2. Пусть $D = \alpha^2 - a_0/a_2$.
3. Если $D < 0$, то перейти к шагу 6.
4. Вычислить вещественные корни $x_1 = \alpha - \sqrt{D}$, $x_2 = \alpha + \sqrt{D}$.
5. Перейти к шагу 7.
6. Вычислить вещественную и мнимую составляющие комплексно-сопряженных корней $\mathrm{Re}\, x_{1,2} = \alpha$, $\mathrm{Im}\, x_{1,2} = \sqrt{-D}$.
7. Закончить вычисления.

Для наглядности алгоритмы представляют *схемами* (рис. 1.1), на которых операторы присваивания обозначают прямоугольниками, а операторы условных переходов — ромбами с выходами «Да» и «Нет», соответствующими удовлетворению и неудовлетворению проверяемого условия. Начало и конец алгоритма обозначают овалами. Все обозначения операторов соединяют линиями, указывающими на порядок выполнения операций.

Неразветвленные алгоритмы, не содержащие операторов условных переходов, называют *линейными*. Если один из выходов оператора условного перехода соединен с предыдущей частью алгоритма, то образуется замкнутая структура, называемая *циклом*, операции в котором многократно повторяются до удовлетворения или неудовлетворения условия, проверяемого оператором условного перехода.

Алгоритмы решения сложных задач для упрощения записи структурируют, заменяя типовые структуры (линейные части алгоритмов, циклы и т.д.) составными операторами, отображаемыми определенными сочетаниями слов или символов. К ним относятся, в частности, операторы цикла общего назначения «пока <условие>, выполнять <оператор>», где описание операции называют *телом цикла*. Если операции в цикле повторяются для ряда значений переменной x, отличаю-

щихся постоянным приращением Δx в интервале (x_1, x_2), то используют оператор цикла

«для $x = x_1$ до x_2 шаг Δx выполнять <оператор>».

Для целочисленных значений переменной x при шаге $\Delta x = 1$ этот оператор упрощают:

«для $x = x_1$ до x_2 выполнять <оператор>».

Применение составных операторов упрощает запись и чтение алгоритма, если их смысл понятен исполнителю алгоритма, в связи с чем обычно используют лишь стандартизованные описания сложных операций.

Исполнителем алгоритма может быть не только человек, но и автомат (в частности, ЭВМ), содержащий устройства ввода, распознавания и выполнения операторов алгоритма. В последнем случае алгоритм решения задачи представляют *программой* на входном языке (языке программирования) с определенным множеством символов алфавита, из которых в соответствии с лексическими правилами формируются слова (операторы), распознаваемые и исполняемые ЭВМ. Допустимое множество операторов входного языка образует его словарный запас, допустимые последовательности операторов в программе определяются синтаксическими правилами входного языка.

В цифровых ЭВМ информация передается в двоичных кодах в виде последовательности состояний, отображаемых 0 или 1. Сменные программы решения прикладных задач, называемые *прикладными программами* или программами пользователя, для первых ЭВМ также составлялись в двоичных кодах. В связи с большой трудоемкостью составления и ввода таких программ вскоре ЭВМ стали дополнять *ассемблерами* — устройствами, формирующими двоичные коды команд, управляющие выполнением простейших операций, при вводе с клавиатуры легко запоминающихся сочетаний букв и цифр. Однако составление программ решения прикладных задач на языках ассемблеров (мнемокодах) также достаточно трудоемко. Поэтому по мере расширения круга пользователей ЭВМ, не имеющих специальной подготовки по вычислительной технике, обострялась необходимость в создании языков программирования высокого уровня, обеспечивающих непосредственное отображение стандартизованных описаний алгоритмов решения задач текстами программ работы ЭВМ.

Большинство языков программирования высокого уровня относится к группе алгоритмических (или процедурных) языков, отображающих способ решения задачи последовательностью описаний выполняемых операций. Первым из наиболее известных языков был разработанный в 1954 г. Фортран (formulae translation — перевод формул), различные версии которого используются до настоящего времени. Большие надежды возлагались на язык Алгол (algorithmic language — алгоритмический язык), предложенный в 1960 г. как международный стандарт для описания алгоритмов и программ решения математических задач. В настоящее время этот язык, в отличие от Фортрана, используется редко и все более вытесняется языками следующих поколений. Среди последних следует отметить достаточно универсальный язык программирования PL/1 (program

language — язык программирования) и его версию APL/1 с удобными средствами доступа к памяти и обработки переменных различного типа. Среди специалистов широкую известность приобрел также язык Паскаль, отличающийся четкими грамматическими правилами и средствами обработки различных структур данных. Среди других языков второго поколения следует отметить Аналитик, разработанный в Институте кибернетики АН УССР с русскими ключевыми словами, который, в частности, отличается возможностью представления чисел, содержащих до 999 десятичных цифр, и преобразования математических выражений в аналитическом (символьном) виде. Первые версии Аналитика предназначались для машин МИР (Машины для Инженерных Расчетов), в настоящее время Аналитик используется и для программирования машин серий СМ ЭВМ и ЕС ЭВМ.

Внедрение ПЭВМ привело к появлению множества фирменных языков программирования и различных версий традиционных языков, ориентированных на ПЭВМ определенного типа. Наибольшее применение для программирования ПЭВМ нашел язык высокого уровня Бейсик, название которого происходит не от английского слова basic (основной), а является аббревиатурой BASIC от слов Beginner All-purpose Symbolic Instruction Code — многоцелевой язык символьных кодов для начинающих.

Основные преимущества Бейсика — наглядность представления алгоритмов решения задач и простота реализации диалогового режима пользователя с ЭВМ. Алфавит основной версии языка Бейсик содержит 26 прописных английских букв от A до Z, десятичные цифры, разделительные знаки (точка, запятая, точка с запятой, двоеточие, апостроф или кавычки, скобки), знаки отношений ($=$, $\neq$, $>$, $\geqslant$, $\leqslant$, $<$), знаки арифметических операций $+$, $-$, $*$ (умножение), $/$ (деление), $\uparrow$ (возведение в степень), а также знаки пробела и перехода на новую строку (перевода каретки). Кроме того, алфавит включает символы вычисления стандартных (встроенных) функций: тригонометрических SIN(X), COS(X), TAN(X), ATN(X), натурального логарифма LOG(X), экспоненты EXP(X), корня квадратного SQP(X), абсолютного значения ABS(X), целой INT(X) и дробной FRC(X) частей, знака SGN(X) числа, а также RAN(X) — вызова квазислучайного числа с равномерным распределением в интервале (0,1).

Имена переменных в основной версии Бейсика формируются из буквы или буквы и цифры, а порядок числа в показательной форме записывается после буквы E (exponent — порядок), например —8.5E—2 или —.85E—1, причем дробная часть числа отделяется от целой, как и в большинстве других языков программирования, точкой, а не запятой.

Для ввода последовательности (массива) чисел используется оператор

DATA a_1, a_2, ..., a_n,

где после имени оператора DATA (данные) записываются последовательность чисел, отделяемых запятыми. Имена этим числам присваиваются оператором

READ I_1, I_2, ..., I_n,

где после имени READ (читать) записываются выбранные имена переменных, отделяемые запятыми. Оператор DATA может быть считан несколькими операторами READ с числом переменных, меньшим числа элементов массива данных, содержимое которого при необходимости восстанавливают оператором RESTORE (восстановить).

Числовые значения переменным можно также присваивать оператором присваивания общего вида

LET $I_1 = I_2 = \ldots = I_m = A$,

после выполнения которого именам переменных I_1, ..., I_m присваивается значение числа A, переменной с именем A или результата вычислений по выражению A.

Оператор условного перехода

IF <условие> THEN <метка перехода>

после ключевого слова IF (если) содержит проверяемое условие, при удовлетворении которого управление передается оператору, перед которым указана мет-

ма, записанная после слова THEN (тогда). Для соединения ветвей алгоритма используют оператор безусловного перехода

GOTO <метка перехода>

Базовая версия языка Бейсик содержит также оператор цикла с заголовком

FOR $I = a_1$ TO a_2 STEP a_3,

где после ключевого слова FOR (для) записывается имя переменной и равное ей в начальном состоянии число a_1, после слова TO — число, равное граничному значению переменной, а после слова STEP (шаг) — число a_3, равное приращению переменной на каждой итерации. После заголовка цикла в программе записывают операторы, отображающие выполняемые действия на каждой итерации и называемые телом цикла. Тело цикла оканчивается оператором NEXT I (следующее I).

Основным оператором вывода информации является оператор PRINT (печатать), после имени которого записываются выводимый текст, заключенный в апострофы или кавычки, и имена переменных, числовые значения которых выводятся на дисплей или печать.

В программе каждый оператор записывается с новой строки и начинается с метки в виде целого числа. Метки записываются в порядке возрастания чисел с некоторым интервалом, обеспечивающим возможность ввода в режиме диалога промежуточных операторов, так как ЭВМ выполняет их не в порядке записи, а в порядке возрастания меток. Наибольшая метка записывается перед оператором END (конец) в конце программы, которая может содержать и операторы STOP для временного прекращения выполнения программы.

Во многих версиях Бейсика можно опускать ключевое слово LET (пусть) и записывать несколько операторов в одной строке с общей меткой. В этом случае запись программы упрощается и, например, алгоритм решения квадратного уравнения на рис. 1.1 можно представить программой

```
5    DATA  a0, a1,  a2
10   READ  A0, A1, A2
15   A=-.5*A1/A2; D=A↑2-A0/A2
20   IF  D<0 THEN 40
25   X1=A-SQR(D); X2=A+SQR(D)
30   PRINT  "REAL  ROOTS  X1=",X1, "X2= ",X2
35   STOP
40   C=SQR(-D)
50   PRINT  "COMPLEX ROOTS REX= ",A, "IMX= ",C
999  END
```

После выполнения этой программы при нажатии клавиши RUN (пуск) при $a_0 = -3$, $a_1 = 2$, $a_1 = 1$ на дисплей или печать будет выведен результат

REAL ROOTS X1 = −3 X2 = 1

а при $a_0 = 5$, $a_1 = 1$, $a_2 = -1$ — результат

COMPLEX ROOTS REX = −1 IMX = 2

Для исправления или замены текста оператора достаточно ввести исправленный текст и нажать клавишу RUN. Аналогичные операции выполняют при вводе дополнительного оператора с промежуточным значением метки, что обеспечивает простоту реализации диалога пользователя с ЭВМ.

В ПЭВМ текст вводимой программы и результаты ее выполнения обычно выводятся на экран дисплея, что существенно упрощает контроль вводимой программы и ее отладку.

Особенности входных языков ПМК первых поколений, также относящихся к процедурным языкам программирования высокого уровня, связаны с требованием минимума аппаратурных затрат.

При выполнении прикладной программы на алгоритмическом языке высокого уровня, подобного Фортрану или Бейсику, считывание кодов оператора присваивания с формулой вида $I = A$ приводит к автоматическому вызову из основной (оперативной) памяти в быструю (сверхоперативную) память (где хранятся операнды) значений переменных, имена которых содержатся в выражении A. Результат выполнения операции автоматически засылается в ячейку памяти, которой соответствует выбранное имя I результата операции. Например, при выполнении оператора $X = X + Y$ из основной памяти вызываются значения переменных X и Y и после их сложения результат засылается в ячейку памяти, где прежде хранилось значение операнда X.

При таком обмене информацией между быстрой и оперативной памятью емкость последней не всегда используется полностью, так как оперативная память разбита на области для хранения кодов программы, данных различного типа и вспомогательной информации, и при заполнении одной из областей памяти вычисления прекращаются. Ограниченная емкость памяти ПМК первых поколений заставила разработчиков изменить способ обмена информацией между быстрой и основной памятью в процессе обработки информации. Для полного использования емкости памяти во входных языках ПМК предусмотрены специальные операторы обращения к ячейкам (регистрам *) памяти данных. Так, для вызова копии содержимого регистра памяти с адресом N используют операторы FN, ИПN (из памяти), П→ xN или RCLN (recall — вызвать), а для засылки результата операции в оперативную память — операторы PN, ПN, x → ПN или STON (storage — запасать). Эти операторы равнозначны соответственно операторам присваивания вида «пусть x = PN» и «пусть PN = x», где символом x обозначена текущая переменная, а символом (именем) PN — содержимое регистра памяти N. Таким образом, в отличие от алгоритмических входных языков, при использовании которых обмен данными между быстрой и оперативной памятью выполняется автоматически, во входных языках ЭКВМ и ПМК для этой цели используются операторы обращения к памяти.

Быстрая память ПМК содержит не менее двух регистров, обычно обозначаемых символами X и Y, причем содержимое регистра X (называемого входным) отображается на индикаторе. В прикладных программах на входных языках ПМК имена текущих переменных, содержащихся в этих регистрах, не указываются (умалчиваются), а соответствующие операторы присваивания обозначают только символом операции. Например, оператор «пусть $x = \sin x$» обозначается только sin, а двухместный оператор «пусть $x = y + x$» — только знаком +. Подобное компактное представление вычислительных (функциональ-

*Слово «регистр» («ряд» в дословном переводе с латинского), означающее упорядоченную запись, в вычислительной технике используют для определения как порций (слов) информации, отображаемой рядом двоичных разрядов, так и физических устройств для их хранения. Часть регистра также называют регистром, а в случаях, когда все регистры имеют одинаковое число разрядов, это же слово используют и в качестве рабочей единицы измерения количества информации.

ных) операторов затрудняет чтение программы, но существенно упрощает конструкцию ПМК и сокращает время ввода программ.

Исходные значения переменных в ПМК вводят в быструю память операторами набора десятичных цифр, разделительного знака (запятой или точки), изменения знака, обозначаемого символами $'-$, $+/-$ или CHS (change sign — изменение знака), ввода порядка n, ВП, EE или EEX (enter exponent — ввод порядка) при показательной форме представления чисел, а также операторами набора констант (например, числа π) и квазислучайных чисел с символами СЧ или RND (random — случайный). Эти операторы используются как для предварительной засылки исходных данных в память, так и для набора операндов в процессе выполнения программы.

Для примера приведем программу решения квадратного уравнения (рис. 1.1) на входном языка ПМК Programmable-59 фирмы Texas Instrument, записав ее по строкам, близким к строкам программы на языке Бейсик, в предположении, что коэффициенты a_0, a_1 и a_2 предварительно занесены соответственно в регистры 0, 1 и 2:

RCL1 ÷ RCL2 ÷ 2 = +/— STO3
x^2 — RCL0 ÷ RCL2 = STO4
x<0? A
$\sqrt{x}$ STO5 + RCL3 = RCL3—RCL5 = R/S
LBLA $\sqrt{x}$ R/S RCL4 +/— $\sqrt{x}$ RCL3
R/S

Если корни уравнения вещественные, то после выполнения этой программы на индикаторе высвечивается значение одного из корней, а для вызова другого достаточно нажать клавишу X ⇔ T. Если корни комплексно-сопряженные, высвечивается символ ошибки (переполнения), после чего следует нажать клавишу R/S (run/stop — пуск/стоп). Это приведет к высвечиванию вещественной части корней, а при дополнительном нажатии клавиши X ⇔ T — модуля мнимой части корней.

В процессе выполнения этой программы в регистры 3 и 4 засылаются значения $\alpha = -a_1/2a_2$ и $D = \alpha^2 - a_0/a_2$. Учитывая, что в регистрах 0, 1 и 2 хранятся исходные коэффициенты, эта программа отображает следующий алгоритм решения задачи:

Пусть $\alpha = -a_1/2a_2$.
Пусть $D = \alpha^2 - a_0/a_2$.
Если $D < 0$, то перейти к шагу с меткой A.
Пусть $x_2 = \alpha + \sqrt{D}$; $x_1 = \alpha - \sqrt{D}$.
Стоп.
A «Пусть $x = \sqrt{D}$».
Стоп.
Пусть $\mathrm{Im}x_{1,2} = \sqrt{-D}$; $\mathrm{Re}x_{1,2} = \alpha$.
Стоп.

Каждая строка такого описания содержит простой или сложный оператор, соответствующий строке приведенной ранее программы на Бейсике. Это подтверждает высокий уровень входных языков ПМК, позволяющих, как и алгоритмические языки ЭВМ других классов, практически непосредственно отображать словесно-формульные описания способов решения прикладных задач с выполнениями операций над представлениями чисел без их разбиения на части, что характерно для языков низшего уровня.

В дальнейшем для определенности процедурные языки программирования высокого уровня, подобные Фортрану или Бейсику, будем называть алгоритмическими, а входные языки традиционных ПМК — компактными.

В связи с сокращенными обозначениями операторов входные языки ПМК иногда ошибочно относят к языкам ассемблеров. Однако программы на таких языках образованы последовательностями слов, управляющих выполнением простейших операций над частями двоичных представлений чисел, и, например, программа решения квадратного уравнения на языках ассемблеров содержит несколько сотен команд.

Следует добавить, что в большинстве языков программирования высокого уровня (исключением является, например, язык программирования Форт) использованы синтаксические правила ввода последовательностей операторов и операндов, характерные для обычной алгебраической записи формул, отображающих в большей степени отношения между числами, чем последовательность операций. В подобной записи символы одноместных операций записываются перед операндом, например $\sin x$, а двухместных — между операндами, например $x+y$. Однако такая алгебраическая запись удобна для описания отношений между переменными и не соответствует фактическим операциям, выполняемым человеком или ЭВМ только после ввода операндов. Так, при вычислениях по несложной формуле $y = a - b \ln (c/d)$ и считывании символов правой части равенства слева направо приходится запоминать все знаки операций и операндов и выполнять операции лишь после считывания закрывающей скобки. Это существенно усложняет конструкцию ЭВМ, которая должна не только запомнить все вводимые символы, но и учитывать допустимый порядок выполнения различных операций, например, умножения перед вычитанием, деления перед вычислением логарифма.

В связи с подобными недостатками синтаксических правил алгебраической записи во входных языках большинства ПМК используют правила обратной (называемой польской для двухместных операций) записи: одно- и двухместные функциональные операторы записывают после операндов, например $x \sin$ вместо $\sin x$ или $y \uparrow x +$ вместо $y + x =$. Символ $\uparrow$, разделяющий набираемые операнды, соответствует оператору засылки содержимого регистра X в регистр Y. Это позволило существенно упростить конструкцию ПМК, так как отпадает необходимость в запоминании кодов операторов, которые могут выполняться непосредственно после их считывания. Например, вычисления по формуле $y = a - b \ln (c/d)$ соответствуют обратной записи $a \uparrow b \uparrow c \uparrow d \div \ln \times -$ и выполняются непосредственно после считывания в программе функциональных операторов.

Характерной особенностью входных языков ПМК с синтаксическими правилами обратной записи является замена оператора = вывода результата (обязательного во входных языках с алгебраической записью) оператором, обозначаемым символом $\uparrow$, В $\uparrow$, Enter $\uparrow$ (enter $\uparrow$ — ввести вверх) и обычно не вводимым для разделения операндов, вызы-

ваемых из памяти. Примером может служить программа на входном языке с синтаксическими правилами обратной записи для ПМК HP-41C, представляющая алгоритм решения квадратного уравнения (предполагается предварительный ввод коэффициентов a_0, a_1 и a_2 в регистры памяти 0, 1 и 2 соответственно):

```
    RCL1 CHS RCL2 ÷ 2 ÷ STO3
    x² RCL0 RCL2 ÷ — STO4
    x<0? GOTO 01
    √x STO5 RCL3 +RCL3 RCL5 — R/S
01  √x R/S RCL4 CHS √x RCL3
    R/S
```

Эта программа, как и предыдущая, записана по строкам, соответствующим строкам программы на Бейсике, но в общем случае может быть записана и в виде последовательности операторов, как в языках PL/1 или Аналитик. Ввод исходных данных и вывод результатов для этой и предыдущей программы совпадают.

Следует обратить внимание, что в предыдущей программе метка вводится символами LBL (lable — метка) и буквенным символом метки, а во входном языке ПМК HP-41C используются двузначные целочисленные метки с переходом к ним по оператору безусловного перехода GO TO (go to — идти на), вводимого в состав оператора условного перехода.

ПМК HP-41C и Programmable-59 наиболее дорогие, с большой емкостью памяти, допускающей ввод программ, содержащих более 1000 шагов. Во входных языках массовых ПМК с небольшой емкостью памяти шаги программы адресуются числами 00, 01, 02, ..., по которым и выполняется переход к нужному оператору. Они указываются только в операторах условных и безусловных переходов, а метки указываются в программах дважды — как в операторах переходов, так и перед «отмеченными» операторами. Переход по адресам позволяет сократить длину программы, несколько затрудняя ее чтение.

1.3. ОСОБЕННОСТИ АРХИТЕКТУРЫ

Организация обмена информацией, называемая архитектурой ЭВМ, основана на принципах, сформулированных еще в 1945 г.:

информация представляется в двоичных кодах и разбивается на слова, имеющие самостоятельный смысл;

разнотипные слова различают по способу использования, а не по способу кодирования;

слова размещают в ячейках памяти и обозначают номерами ячеек, называемых адресами слов;

алгоритм решения прикладной задачи отображают однозначной последовательностью управляющих слов, называемых командами;

решение задачи сводится к последовательному выполнению команд в порядке, определяемом алгоритмом и назначением команд.

Выбор двоичных кодов для представления информации в ЭВМ обусловлен простотой реализации символов 0 и 1 двумя дискретными состояниями материальных носителей информации (например, «нет» и «есть» магнитное поле или «включено» и «выключено» электрическое напряжение). Двоичная позиция с символом 0 или 1 отображает минимальное количество информации, называемое *битом*, а n-битовым словом представляют до 2^n различных состояний, соответствующих большим количествам информации *.

Для двоичного представления символов, вводимых с клавиатуры или других устройств ввода информации в ЭВМ, обычно используют стандартные 7- или 8-битовые коды. Числовые данные чаще всего представляют в двоичной системе счисления.

Напомним, что в позиционных m-ичных системах счисления с основанием m и цифрами 0, 1, ..., $m-1$ целые числа представляют последовательностями $a_r a_{r-1} \ldots a_1 a_0$ позиций, называемых m-ичными разрядами. При этом значащая (отличающаяся от нуля) цифра в разряде a_0 отображает соответствующее натуральное число, в других разрядах — число, в m раз большее, чем в соседнем правом разряде. Сумма натуральных чисел, соответствующих содержимому всех разрядов m-ичного числа, равна натуральному (десятичному) числу

$$a_r m^r + a_{r-1} m^{r-1} + \ldots + a_1 m^1 + a_0 m^0 \leqslant m^r - 1. \qquad (1.1)$$

Например, десятичное представление числа

$$a_{10} = 687 = 6 \cdot 10^2 + 8 \cdot 10^1 + 7 \cdot 10^0 < 10^3 - 1 = 999$$

соответствует шестнадцатеричному представлению числа** $a_{16} = 2AF$, так как $2 \cdot 16^2 + 10 \cdot 16^1 + 15 \cdot 16^0 = 687$, или двоичному представлению $a_2 = 1010101111$, так как $1 \cdot 2^9 + 0 \cdot 2^8 + 1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 512 + 128 + 32 + 8 + 4 + 2 + 1 = 687$.

Связь между представлениями чисел в двоичной и других позиционных системах счисления с основанием m наиболее простая, когда $m = 2^k$ кратно двум. В этом случае содержимое каждого m-ичного разряда отображается содержимым k двоичных разрядов. В частности, каждому восьмеричному разряду соответствует тройка, а шестнадцатеричному — тетрада (четверка) двоичных разрядов, отсчитываемых справа (табл. 1.1).

Разбив, например, двоичное число $a_2 = 010100111101$ на тройки разрядов 010 100 111 101 и заменив каждую из них соответствующей восьмеричной цифрой (0, 1, ..., 7), получим восьмеричное представление того же числа $a_8 = 2475$. Аналогично разбив исходное двоичное число (для простоты m-ичные представления абстрактного понятия числа называют m-ичными числами) на тетрады 0101 0011 1101, получим шестнадцатеричное число $a_{16} = 53D$. В связи с удобством подобных преобразований при описании работы ЭВМ двоичные слова независимо от их смысла рассматривают как представления чисел и часто отображают более короткими восьмеричными или шестнадцатеричными числами.

---

*В качестве единиц измерения больших количеств информации (определяемых числом возможных состояний) используют байт (8 бит), килобит (1 Кбит = 1024 бит), килобайт (1 Кбайт = 1024 байт), мегабит (1 Мбит = 1024 Кбит) и мегабайт (1Мбайт = 1024 Кбайт).

**В шестнадцатеричной системе счисления используют цифры 0, 1, ..., A, B, C, D, E, F, где буквами обозначены натуральные числа от 10 до 15.

Таблица 1.1

Представление первых членов натурального ряда чисел в системах счисления с основанием

$m=10$	$m=10/2$	$m=2$	$m=4$	$m=8$	$m=16$
0	0	0	0	0	0
1	1	1	1	1	1
2	10	10	2	2	2
3	11	11	3	3	3
4	100	100	10	4	4
5	101	101	11	5	5
6	110	110	12	6	6
7	111	111	13	7	7
8	1000	1000	20	10	8
9	1001	1001	21	11	9
10	10000	1010	22	12	A
11	10001	1011	23	13	B
12	10010	1100	30	14	C
13	10011	1101	31	15	D
14	10100	1110	32	16	E
15	10101	1111	33	17	F
16	10110	10000	100	20	10
17	10111	10001	101	21	11
18	11000	10010	102	22	12
19	11001	10011	103	23	13
20	100000	10100	110	24	14
21	100001	10101	111	25	15
22	100010	10110	112	26	16
23	100011	10111	113	27	17
24	100100	11000	120	30	18
25	100101	11001	121	31	19
26	100110	11010	122	32	1A
27	100111	11011	123	33	1B
28	101000	11100	130	34	1C
29	101001	11101	131	35	1D
30	110000	11110	132	36	1E
31	110001	11111	133	37	1F
32	110010	100000	200	40	20
33	110011	100001	201	41	21

Действительные (в общем случае с дробной частью) числа представляют в m-ичных позиционных системах последовательностями разрядов $a_r m^r_r \ldots a_0 m^0$, $a_{-1} m^{-1} \ldots a_{-s} m^{-s}$ с запятой * после разряда a_0. Такое представление отображает натуральное (десятичное) число

$$a_{10} = a_r m^r + a_{r-1} m^{r-1} + \ldots + a_0 m^0 + a_{-1} m^{-1} + \ldots + a_{-s} m^{-s}.$$

Действительные двоичные числа в системах счисления с кратными основаниями отображаются точно (при отсчете разрядов от запятой,

*В большинстве алгоритмических языков для этой цели используют точку.

но при некратных основаниях точное представление дробных чисел может оказаться невозможным в связи с бесконечным числом разрядов, требуемым для точного отображения дробной части.

В ЭВМ однотипные двоичные слова характеризуются *форматом*, определяющим общее число разрядов (длину) слова и их распределение между полями (частями) слова, имеющими самостоятельный смысл (например, поле из одного бита для представления знака числа). Фиксированное число десятичных или двоичных разрядов (разрядность чисел или разрядная сетка ЭВМ) представлений чисел ограничивает диапазон (A_{min}, A_{max}) их абсолютных значений, а также точность, определяемую максимальным числом разрядов дробной части.

Диапазон десятичных чисел с k разрядами в естественной форме (с запятой, фиксированной после целой части) относительно небольшой и ограничен по абсолютным значениям пределами $A_{min} = 10^{1-k}$ и $A_{max} = 10^k - 1$, причем нижняя граница определяет и предельную точность представления чисел. Поэтому в современных ЭВМ и ПМК действительные числа вводят и выводят в основном в десятичной *нормализованной показательной* (с плавающей запятой) *форме* $a_{10} = M \cdot 10^n$, где M — мантисса с определенным положением запятой, а n — целый порядок. В большинстве ЭВМ используют нормализованную форму с запятой перед первой значащей цифрой мантиссы, значения которой находятся в интервале $0{,}1 \leqslant M \leqslant 1$. В ПМК используют нормализованную форму с запятой после первой значащей цифры мантиссы, когда $1 \leqslant M \leqslant 10$. В последнем случае при k разрядах мантиссы и l разрядах порядка диапазон представления чисел ограничен значениями $A_{min} = 1 \cdot 10^{1-10^l}$ и $A_{max} = (10 - 10^{1-k})\, 10^{10^l - 1}$. Так, при $k=8$ и $l = 2$ $A_{min} = 1 \cdot 10^{-99}$ и $a_{max} = 9{,}999999 \cdot 10^{99}$. Для сравнения укажем, что в большинстве версий языка Фортран целочисленные значения порядка лежат в интервале $-32 \leqslant n \leqslant 32$.

Таким образом, показательная форма представления чисел обеспечивает значительное расширение диапазона чисел по сравнению с естественной формой, причем этот диапазон в основном определяется разрядностью порядка, а предельная точность представления чисел — количеством разрядов мантиссы, выделяемых для представления дробной части числа.

В ЭВМ числа представляют двоичными кодами, причем чаще всего используют двоичную систему счисления. В связи с некратностью оснований $m = 2$ и $m = 10$ десятичное число не всегда может быть точно представлено в двоичной системе счисления. В частности, для представления l разрядов порядка десятичного числа в двоичной системе счисления требуется $l/\lg 2$ разрядов. Так, при $l = 2$ требуется $2/\lg 2 \approx$ $\approx 6{,}644 < 7$ разрядов, которыми можно представить порядок $n \leqslant 127$. Для отображения десятичной мантиссы с $k = 8$ разрядами требуется не менее $8/\lg 2 \approx 27$ двоичных разрядов, которыми можно представить десятичное число $2^{27} - 1 \approx 1{,}3 \cdot 10^8$. Десятичное число $a_{10} = 1{,}5$ точно представляется двоичным числом $a_2 = 1{,}1$, но десятичное число $a_{10} = 1{,}4$ не может быть точно представлено в двоичной системе счис-

ления, например, числам с различным числом разрядов дробной части $a_2 = 1{,}0110$; 1,01100110; 1,011001100110 соответствуют приближенные значения исходного десятичного числа $a_{10} = 1{,}375$; 1,3984375; 1,39999024. Поэтому и возникает погрешность представления десятичных чисел в двоичной системе счисления с ограниченной разрядностью.

В ПМК для представления чисел обычно используют *десятично-двоичный код*, называемый кодом ДКД или 8421, при котором содержимое каждого десятичного разряда дробной или целой части действительного десятичного числа представляют тетрадой двоичных разрядов отсчитываемых от запятой. Если в этом случае содержимое тетрады больше натурального числа 9, то единицу переносят в следующую тетраду. В двоичной системе счисления такой перенос выполняют, когда содержимое тетрады больше натурального числа 15.

Десятично-двоичное представление чисел менее экономно, чем двоичная система счисления. Например, для представления натурального числа 10 требуется одна тетрада 1010 в двоичной системе счисления, но две тетрады 00010000 в ДКД. Однако этот код обеспечивает точное представление десятичных чисел, например десятичные числа $a_{10} = 1{,}5$ и $a_{10} = 1{,}4$ точно отображаются десятично-двоичными кодами $a_{10/2} = 0001{,}0101$ и $a_{10/2} = 0001,\ 0100$.

Все вычисления сводятся к последовательности арифметических операций, а последние — к операциям сложения. Сложение операндов $x + y = r$ в позиционной системе счисления с основанием m человек выполняет последовательно для каждого i-го разряда слагаемых с учетом переноса p_i из предыдущего разряда согласно алгоритму:

1. Принять $i = 0$, $p_0 = 0$.
2. Принять $s_i = x_i + y_i + p_i$.
3. Если $a_i = s_i - m < 0$, то принять $p_{i+1} = 0$, $r_i = s_1$, иначе принять $p_{i+1} = 1$, $r_i = a_i$.
4. Принять $i = i + 1$ и перейти к шагу 2.

Этот алгоритм нельзя непосредственно реализовать на ЭВМ, так как он основан на интуитивном (хотя и изучаемом в школе при $m = 10$) умении складывать и вычитать небольшие целые числа. Однако, например, для $m = 2$ подобный алгоритм можно представить последовательностью логических высказываний:

1. Принять $i = 0$, $p_0 = 0$.

2. Если x_i, y_i и p_i равны 1, то принять $p_{i+1} = 1$, $r_i = 1$ и перейти к шагу 6.

3. Если x_i и y_i, или x_i и p_i, или y_i и p_i равны 1, то принять $p_{i+1} = 1$, $r_i = 0$ и перейти к шагу 6.

4. Если x_i, или y_i, или p_i равны 1, то принять $p_{i+1} = 0$, $r_i = 1$, и перейти к шагу 6.

5. Принять $p_{i+1} = 0$, $r_i = 0$.

6. Принять $i = i + 1$ и перейти к шагу 2.

Для полного описания операции сложения чисел с заданной разрядностью подобный алгоритм должен быть дополнен условием окон-

чания вычислений. Такие алгоритмы содержат операции двузначной математической логики над двоичными переменными, принимающими значения 0 или 1. Над двумя такими переменными можно выполнить всего 16 операций [16], но все их можно заменить выполнением одноместных операций логического отрицания или инверсии (логическое НЕ) $\overline{0} = 1$ и $\overline{1} = 0$ и двухместных логических операций умножения (логическое И) $0 \wedge 0 = 0$, $0 \wedge 1 = 0{,}1 \wedge 0 = 0{,}1 \wedge 1 = 1$, сложения (логическое ИЛИ) $0 \vee 0 = 0{,}0 \vee 1 = 1{,}1 \vee 0 = 1{,}1 \vee 1 = 1$, а также сложения над полем модуля 2 (исключающее ИЛИ) $0 \oplus 0 = 0$, $0 \oplus 1 = 1{,}1 \oplus 0 = 1{,}1 \oplus 1 = 0$.

Используя символы логических операций, можно несколько упростить последний алгоритм сложения двоичных чисел:

1. Принять $i = 0$, $p_0 = 0$.
2. Если $x_i \wedge y_i \wedge p_i = 1$, то принять $p_{i+1} = 1$, $r_i = 1$ и перейти к шагу 6.
3. Если $(x_i \wedge y_i) \vee (x_i \wedge p_i) \vee (y_i \wedge p_i) = 1$, то принять $p_{i+1} = 1$, $r_i = 0$, перейти к шагу 6.
4. Если $x_i \vee y_i \vee p_i = 1$, то принять $p_{i+1} = 0$, $r_i = 1$ и перейти к шагу 6.
5. Принять $p_{i+1} = 0$, $r_i = 0$.
6. Принять $i = i + 1$ и перейти к шагу 2.

Преобразуя логические высказывания в соответствии с правилами булевой алгебры, можно отобразить этот алгоритм для i-го разряда слагаемых формулой

$$p_{i+1} = (x_i \wedge y_i) \vee (x_i \wedge p_i) \vee (y_i \wedge p_i); \quad r_i = \overline{p}_{i+1} \vee (x_i \wedge y_i \wedge p_i).$$

Техническая реализация этой формулы с помощью электронных элементов, выполняющих логические операции над импульсами напряжения, отображаемыми 1 (импульс) и 0 (пауза), представляет собой последовательный двоичный сумматор, поразрядно складывающий двоичные операнды. Подобным образом строят параллельные (выполняющие операции одновременно над всеми разрядами двоичных операндов) или комбинированные (последовательно-параллельные) сумматоры.

Умножение двоичных операндов сводится к поочередному сдвигу множимого влево на один или два (в зависимости от содержимого очередного разряда множителя) разряда с последующим сложением таких смещенных слагаемых. Например, произведению десятичных чисел $10 \cdot 5 = 50$ соответствует двоичное произведение $1010 \cdot 0101 = 1010 + 00000 + 101000 + 0000000 = 0110010$. Сдвиг на один разряд влево обеспечивается также сложением операнда с самим собой, например $1010 + 1010 = 10100$.

Вычитание реализуется сумматором при кодировании отрицательных чисел по дополнению. Так, при вычитании с кодированием по дополнению 1 операнды дополняют старшим знаковым разрядом (знак минус отображают 1) и выполняют сложение большего по модулю операнда с инверсией меньшего по модулю, добавляя к результату единицу. Например, разность десятичных чисел $3 - 10 = -7$ отображается

операциями над их двоичными кодами $\overline{00011}+11010+00001 = 11100+$ $+11010+00001 = 10111$, что соответствует отрицательному числу — 7.

Деление сводится к выполнению других арифметических операций, и, следовательно, с помощью двоичного сумматора можно выполнить любые вычисления. Дополнив сумматор элементами, выполняющими основные логические операции над содержимым каждого разряда операндов, получают арифметическо-логическое устройство (АЛУ), обеспечивающее выполнение любых преобразований двоичных слов по заданной программе.

Таким образом, отобразив каждый оператор входного языка соответствующей последовательностью микрокоманд (микропрограммой), управляющих выполнением логическими элементами требуемых операций в АЛУ, можно выполнить любую прикладную программу. Однако непосредственное отображение последовательности операторов прикладной программы последовательностями микропрограмм связано со значительными аппаратурными затратами на их хранение, так как последовательности микрокоманд, управляющих выполнением простейших операций, будут многократно повторяться. Поэтому используют многоступенчатую трансляцию (перевод) прикладной программы с входного языка на язык микропрограмм, отображая на каждой ступени слово языка высшего уровня последовательностью слов на языке низшего уровня.

В универсальных ЭВМ вводимая с клавиатуры прикладная программа отображается последовательностью двоичных кодов операторов и высвечивается для контроля на многострочном дисплее мониторного (индикаторного) устройства в исходном виде. С помощью вспомогательной программы (транслятора) прикладная программа автоматически переводится на язык машинных команд (непосредственно или через промежуточные уровни). Машинные команды обычно имеют одинаковый формат и содержат два или более полей для кода операции и адресов, хранящихся в памяти данных. Транслятор или программа-загрузчик (формирующая в оперативной памяти прикладную программу в виле последовательности команд) присваивает командам адреса, опредедяющие их положение в оперативной памяти.

Команда может содержать поле для кода (признака) адресации, если используются различные ее виды. При прямой адресации номер ячейки оперативной памяти, в которой хранится требуемое слово данных, равен адресу этого слова в команде. При косвенной адресации в команде указывается адрес (второго ранга) ячейки памяти, в которой хранится адрес (первого ранга) ячейки для запоминания нужного слова. В этом случае над адресом второго ранга могут выполняться различные операции, например его изменение (модификация) на единицу при каждом обращении по этому адресу. Часто используется относительная адресация, при которой адрес разбивается на поля с указанием адреса начального слова из хранящегося в памяти массива последовательности и указателя смещения — числа, равного требуемой разности между адресами слов массива. Адрес операнда не указывается (адресацию называют не-

явной), если его вызов предусмотрен кодом операции. Используют также непосредственную адресацию — запись в команде слова данных (литерала) вместо его адреса.

Команды последовательно вводятся в оперативную память программой-загрузчиком и при исполнении программы вызываются из памяти с помощью счетчика команд. При выполнении линейных частей программы содержимое этого счетчика после исполнения очередной команды увеличивается на единицу, и из оперативной памяти вызывается и засылается в основное исполнительное устройство (процессор, содержащий АЛУ) очередная команда. При считывании из памяти команды перехода его адрес засылается в счетчик команд, который вызывает следующей команду с адресом перехода, а затем последующие команды до считывания очередной команды перехода или команды останова. Данные, над которыми выполняются очередные операции, вызываются из оперативной памяти в процессор по адресам, содержащимся в командах.

Если код очередного оператора входного языка непосредственно отображается последовательностью слов на языке низшего уровня, то транслятор называют интерпретатором и компилятором в противном случае. В универсальных стационарных ЭВМ обычно используют компиляторы, обеспечивающие более эффективное представление прикладной программы на языке машинных команд. При этом представление прикладной программы последовательностью кодов операторов входного языка обычно используется лишь при трансляции вводимой программы и после загрузки в память представления программы на языке машинных команд стирается или засылается в накопитель информации.

Специфичной особенностью ПМК является засылка в оперативную память программы на входном языке и трансляция (интерпретация) ее на язык машинных команд в процессе исполнения. Это связано с необходимостью обеспечения выполнения операции в непрограммируемом режиме нажатием клавиш. Поэтому коды операторов компактных входных языков ПМК имеют одинаковый формат или форматы нескольких типов, например операторы обращения к памяти близки по структуре к машинным командам. Однако это подобие лишь формальное: уровень компактных входных языков соответствует уровню алгоритмических входных языков и каждый их оператор отображается в процессе исполнения программы последовательностью машинных команд.

Архитектура некоторых современных ПМК, совпадающих по назначению с простейшими ПЭВМ, близка к архитектуре универсальных ЭВМ других классов, использующих алгоритмические языки, при засылке в оперативную память прикладной программы на языке машинных команд, компилируемой из операторов программы на входном языке. При этом в наиболее совершенных ПМК подобного класса прикладная программа может быть введена как на компактном или алгоритмическом входном языке, так и на языке ассемблера, позволяющем не-

посредственно ввести в память прикладную программу в виде последовательности машинных команд.

При исполнении прикладной программы в автоматическом режиме очередная команда непосредственно или через промежуточные уровни транслируется последовательностью микрокоманд (микропрограммой), формирующей множество управляющих сигналов в виде импульсов напряжения, посылаемых на логические элементы для выполнения простейших операций над содержимым двоичных разрядов данных.

Обмен и преобразование информации в ЭВМ обеспечивается вспомогательными программами (включая трансляторы, загрузчики для засылки информации в запоминающие устройства, драйверы для управления работой вспомогательных устройств преобразования информации, мониторы для управления потоком информации и работой других программ, программы-редакторы и т. п.), совокупность которых называют *операционной системой*. Операционная система является частью математического обеспечения ЭВМ, включающего и прикладные программы на входном языке или их фрагменты, хранящиеся в накопителях информации.

Математическое обеспечение разделяют на «жесткое», реализуемое соединением логических элементов, «твердое» на постоянных запоминающих устройствах (ПЗУ), заполняемых при изготовлении, и «мягкое», хранящееся в накопителях информации на полупроводниковых, магнитных и других носителях информации. В ЭВМ высокой производительности, включая профессиональные ПЭВМ, операционная система записана на съемных накопителях информации, и при необходимости можно заменить не только транслятор прикладной программы, но и всю операционную систему. Исключением являются ЭВМ с входными языками высокого уровня (например, Аналитик или Лисп), «нагруженными» операционной системой. В таких ЭВМ как транслятор, так и операционная система реализованы только аппаратурно.

В ПМК операционная система хранится в ПЗУ и реализуется «жесткими» ЗУ. Конкретный состав и объем операционной системы зависит от назначения и структуры ПМК, причем по мере совершенствования ПМК и переходу к использованию алгоритмических языков их операционные системы сближаются с операционными системами других универсальных микроЭВМ.

Следует добавить, что не всегда удается разграничить «жесткое», «твердое» и «мягкое» математическое обеспечение, так как, например, ПЗУ реализуют различными способами, а замена магнитных носителей информации полупроводниковыми стирает и грани между «мягким» и «твердым» математическим обеспечением. Основным же критерием для выбора материальной реализации математического обеспечения ПМК являются минимальные аппаратурные затраты при заданной производительности.

1.4. ОСОБЕННОСТИ СТРУКТУРЫ

Любая ЭВМ, включая и ПМК, является вычислительной системой, структура которой определяется способом соединения элементов, а параметры — их характеристиками. Основными элементами ЭВМ являются центральный процессор (функции которого могут быть распределены между несколькими процессорами) для обработки данных в соответствии с прикладной программой, оперативная (основная) память для хранения прикладной программы и относящихся к ней текущих данных, информационные шины для обмена информацией между различными элементами, а также периферийные или внешние (по отношению к центральному процессору и основной памяти) устройства для ввода, вывода, накопления и промежуточного преобразования информации.

Типовая структурная схема универсальной микроЭВМ приведена на рис. 1.2.

Клавиатура предназначена для ввода информации, индикаторное (мониторное) устройство для ее отображения, управляющие устройства (или контроллеры) обеспечивают преобразование информации при ее вводе с клавиатуры, выводе на индикатор, а также согласование во времени (интерфейс) обмена информацией между системной магистралью и внешними устройствами. Кроме того, микроЭВМ обычно содержит ПЗУ для хранения всей или части операционной системы. Центральный процессор, основная память и внешние устройства содержат собственные управляющие устройства с буферной памятью, обеспечивающей временное хранение информации, особенно необходимое при обмене информацией между устройствами с различной скоростью ее обработки. Управляющие устройства конструктивно выполняют как на «жесткой логике» в виде соединений логических элементов, вырабатывающих требуемые последовательности управляющих сигналов, так и в виде программируемых устройств или управляющих микроЭВМ (мини-микроЭВМ), работающих по программам операционной системы, хранящимся в местном или общем ПЗУ.

Центральный процессор обычно реализуют стандартной микросхемой, называемой микропроцессором, иногда дополняемой специализи-

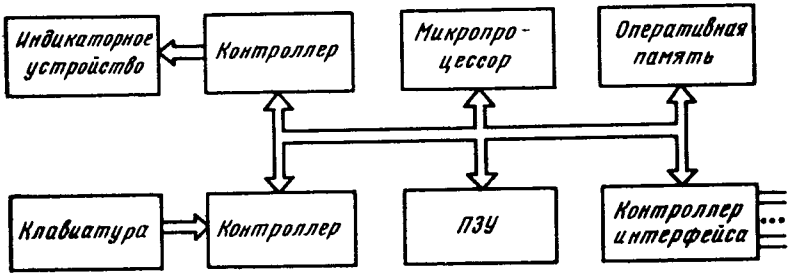


Рис. 1.2

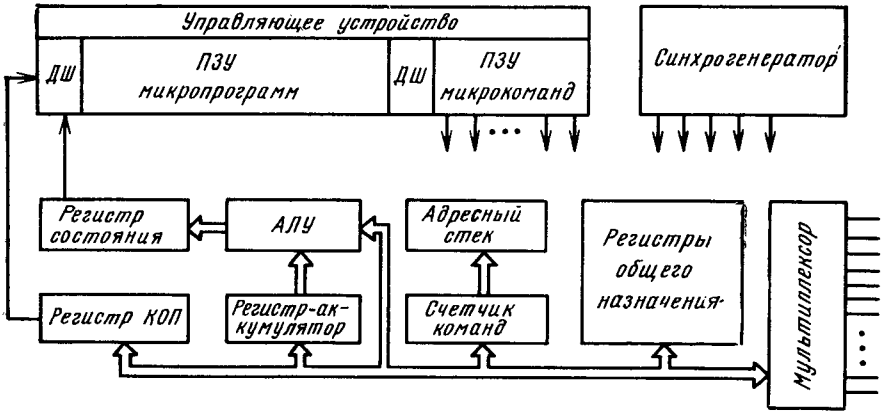


Рис. 1.3

рованными (например, для ускорения операции умножения) или параллельными (для ускорения обработки информации) микропроцессорами. Типичная структурная схема микропроцессора, выполняющего функции центрального процессора (рис. 1.3), содержит системную информационную магистраль, связывающую основные элементы между собой и через мультиплексор с внешними относительно микропроцессора устройствами, включая основную (оперативную память.

Основным операционным устройством микропроцессора является арифметическо-логическое устройство (АЛУ), выполняющее арифметические и логические операции над словами текущих данных, которые временно хранятся в регистре-аккумуляторе и регистрах общего назначения (РОН), образующих быструю или сверхоперативную память микроЭВМ. К ней относятся регистр состояния, в котором хранятся признаки (флаги), характеризующие режим работы, состояние процесса выполнения прикладной программы и знак операнда (сравниваемого с нулем), хранящегося в регистре-аккумуляторе, а также регистр кода операции (КОП) и регистр-счетчик команд, соединенный с адресным стеком (стеком возврата из подпрограмм).

Стеком * называют такое соединение регистров, при котором по управляющим сигналам содержимое каждого из них пересылается в соседний регистр. Различают *магазинный* и *кольцевой режимы* работы стеков. При магазинном режиме смещение содержимого регистров стека «вверх» (нижним условно называют входной регистр стека) приводит к стиранию содержимого последнего регистра, которое замеща-

*Английское слово a stack означает упорядоченную кучу предметов (дров, снопов), смещающихся при удалении нижнего из них.

ется прежним содержимым предпоследнего регистра (рис. 1.4, *а*). При смещении содержимого такого стека «вниз» (рис. 1.4, *б*) содержимое входного регистра замещается содержимым, ранее хранившимся в следующем регистре. В магазинном стеке слово данных, введенное последним во входной регистр стека, выводится, как и патрон в магазине (обойме) стрелкового оружия, первым при смещении стека «вниз» («последним вошел — первым вышел»).

При работе стека в кольцевом режиме смещение «верх» приводит к пересылке прежнего содержимого последнего регистра в первый (рис. 1.4, *в*), а смещение «вниз» — к засылке прежнего содержимого первого регистра в последний (рис. 1.4, *г*), причем в обоих случаях исходная информация сохраняется в стеке. Смещения содержимого регистров кольцевого стека называют поворотом кольцевого стека по и против часовой стрелки.

Микропроцессор содержит также синхрогенератор (иногда называемый «часами»), формирующий короткие импульсы начала фиксированных отрезков времени (тактов), а также управляющие импульсы длительностью, соответствующей определенному количеству тактов.

Управляющее устройство микропроцессора содержит дешифраторы (ДШ) и ПЗУ для хранения слов промежуточных семантических уровней, обеспечивающих многоступенчатую трансляцию прикладной программы в последовательность микрокоманд. Каждая микрокоманда формирует на соответствующем такте микрооператоры или микроприказы, управляющие в течение этого такта работой логических элементов. Таким образом, управляющее устройство выполняет функции транслятора-интерпретатора, преобразуя с помощью дешифраторов и ПЗУ код очередной команды, считываемой из регистра команд, в последовательность кодов микропрограмм. В свою очередь, каждый код микропрограммы отображается последовательностью кодов микрокоманд (микропрограммой). Каждая микрокоманда, управляющая работой микропроцессора в течение одного такта, отображается последовательностью однобитовых микроприказов (микрооператоров) в виде напряжений уровня логической 1 или логического 0 на выводах параллельной выходной шины управляющего устройства микропроцессора.

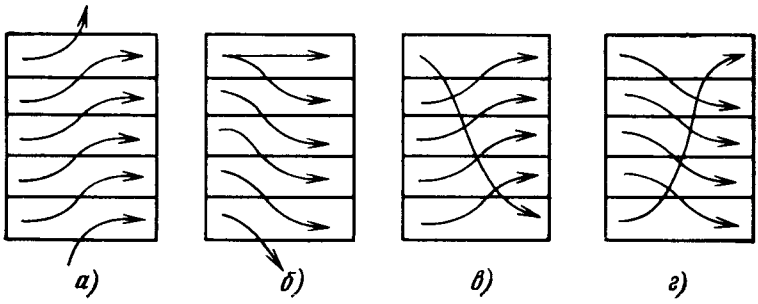


Рис. 1.4

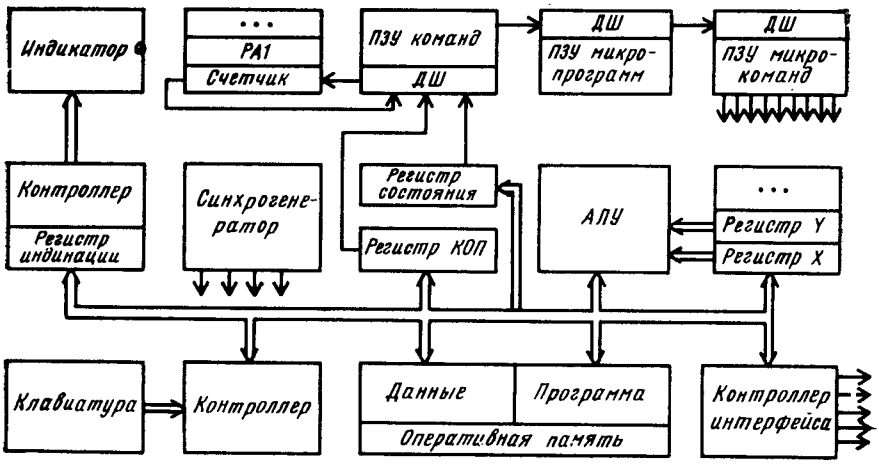


Рис. 1.5

Эти выводы соединены с управляющими входами логических элементов функциональных узлов микропроцессора — АЛУ, регистров памяти, мультиплексора, системной магистрали. Следовательно, каждая команда реализуется в течение определенного числа тактов, равного числу микрокоманд в последовательности микропрограмм, управляющих согласованной во времени работой всех узлов. Мультиплексор обеспечивает согласование (интерфейс) во времени обмена информацией между системной магистралью и внешними по отношению к микропроцессору устройствами микроЭВМ.

Электрические цепи микроЭВМ обычно образованы соединением нескольких БИС, одной из которых является БИС микропроцессора. Для уменьшения габаритных размеров ПМК собирают из небольшого числа БИС, в которых функции микропроцессора обычно совмещены с функциями других узлов. ПМК может являться сложной вычислительной системой, содержащей несколько центральных процессоров или даже несколько отдельных однокристальных микроЭВМ. Независимо от реальной структуры конкретного типа ПМК, определяемой способом соединения и назначением БИС, взаимодействие его отдельных функциональных узлов удобно рассматривать с помощью эквивалентной структурной (функциональной) схемы, подобной показанной на рис. 1.5.

В режиме программирования (ввода прикладной программы в память) с клавиатуры при каждом нажатии клавиши контроллер клавиатуры вырабатывает код вводимого символа входного языка, а последовательность таких кодов в соответствии с лексическими правилами входного языка отображает код оператора. Последовательность кодов операторов, отображающих прикладную программу, в большинство настольных микроЭВМ и некоторых ПМК преобразуется контролле-

ром клавиатуры под управлением программы-транслятора или отдельным транслирующим устройством в последовательность машинных команд, засылаемых в оперативную память. Однако в большинстве ПМК (в особенности с компактными входными языками) для уменьшения требуемой емкости оперативной памяти в нее засылается последовательность кодов операторов, отображающих прикладную программу, а интерпретация этих кодов последовательностями машинных команд выполняется в процессе ее исполнения. Это увеличивает время счета, но существенно снижает требования к емкости оперативной памяти.

Оперативная память ПМК разбита на области программ и данных, причем граница между ними может быть постоянной, что упрощает конструкцию, или переменной, что повышает эффективность использования ресурса памяти. Исходные данные засылаются в оперативную память перед исполнением программы, а результаты выполнения операций — в процессе ее исполнения. Вводимые в режиме программирования прикладная программа и перед пуском программы исходные данные для контроля отображаются на индикаторе.

После пуска программы коды операторов (при представлении программы их последовательностями) поочередно вызываются из области программы оперативной памяти в регистр кода оператора КОП в соответствии с их адресом, равным содержимому счетчика шагов программы, автоматически увеличивающемуся на единицу после выполнения каждого очередного шага программы.

При считывании в регистр КОП кода оператора присваивания он интерпретируется последовательностью команд, каждая из которых транслируется в последовательность микропрограмм, составленных из микрокоманд, управляющих выполнением соответствующих логических операций над двоичными разрядами содержимого рабочих регистров X и Y (или X при одноместных операциях) операционного стека с засылкой результата операции в регистр X. В ПМК с компактными входными языками данные из памяти вызываются в регистр X, а копии содержимого регистра X засылаются в память данных только при считывании соответствующего оператора обращения к памяти. В ПМК с алгоритмическими входными языками данные вызываются из памяти по именам переменных, записанным в правой части оператора присваивания и соответствующим определенным ячейкам памяти данных. Результат операции в этом случае засылается в ячейку памяти, которой при трансляции программы присвоен адрес, соответствующий имени переменной в левой части оператора присваивания.

Содержимое регистра-счетчика увеличивается на единицу только после выполнения оператора и установки флага его исполнения в регистре состояний. При считывании из памяти в регистр КОП кода оператора безусловного перехода адрес перехода засылается в регистр-счетчик и следующим в регистр КОП вызывается код оператора, адрес которого равен адресу перехода. Если в регистр КОП вызван код оператора условного перехода, то выполнение содержащегося в нем условия проверяется по содержимому флагов в соответствующих разрядах

регистра состояний. В зависимости от содержимого этих флагов в регистр-счетчик засылается адрес условного перехода или эта засылка блокируется и следующим вызывается код оператора, записанный в программе после оператора условного перехода.

При считывании из памяти в регистр КОП кода оператора обращения к подпрограмме содержимое адресного стека смещается «вверх» (предыдущее содержимое регистра-счетчика заносится в следующий регистр адресного стека), а в регистр-счетчик заносится адрес обращения к подпрограмме. После этого содержимое регистра-счетчика увеличивается на единицу, и в регистр КОП вызывается код оператора подпрограммы с адресом, равным содержимому регистра-счетчика. После выполнения подпрограммы в регистр КОП считывается код записанного в конце подпрограммы оператора возврата, исполнение которого приводит к смещению «вниз» содержимого адресного стека (стека возврата), и в регистре-счетчике оказывается адрес адреса обращения к подпрограмме. После автоматического увеличения содержимого регистра-счетчика на единицу в регистр КОП вызывается код оператора с адресом, на единицу большим адреса адреса обращения к подпрограмме, записанного в программе после оператора обращения к подпрограмме.

Таким образом, адресный стек (стек возврата) после каждого выполнения подпрограммы обеспечивает переход к оператору, записанному в программе после оператора обращения к подпрограмме. Адресный стек обеспечивает также возможность обращения из подпрограммы к другой подпрограмме (вложение подпрограмм). После каждого такого обращения содержимое адресного стека смещается «вверх», и максимальное число вложений подпрограмм определяется числом регистров адресного стека.

При считывании из программной памяти в регистр КОП кода оператора «Стоп» в регистре состояний устанавливается флаг прекращения выполнения программы и на индикаторе отображается содержимое регистра индикации. В этом случае в регистре-счетчике сохраняется адрес оператора «стоп» и при последующем пуске программы без очистки этого регистра в регистр КОП будет вызываться содержимое ячеек программной памяти с адресами, следующими после адреса оператора «Стоп».

Простейшими элементами структуры ПМК являются операционные логические устройства, образованные логическими элементами, чаще всего типа НЕ, И и ИЛИ. Элемент И (рис. 1,6, *а*), обозначаемый на схеме символом & (И), выполняет операцию логического умножения двух или более двоичных переменных, отображаемых включением (символ 1) или выключением (символ 0) входных напряжений. На выходе (справа на схеме элемента) получают сигнал, ото-

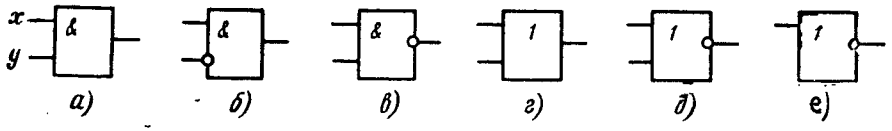


Рис. 1.6

бражающий функцию (произведение) входных сигналов, называемую конъюнкцией $x \wedge y$.

Элемент И выполняет фуикцию ключа, передающего сигнал x на выход только при подаче управляющего сигнала $y = 1$. При инверсии входного сигнала (инверсию на схемах обозиачают кружком на соответствующем выводе элемеита) элемент И (рис. 1.6, б) реализует логическую функцию $x \wedge \bar{y}$, называемую «запретом», в этом случае при $y = 1$ сигнал x не передается на выход. При инверсии выходиого сигиала (рис. 1.6, *в*) рассматриваемый элемент реализует функцию $\overline{x \wedge y}$, называемую отрицаиием конъюикции или функцией Шефера.

Элемеит ИЛИ, обозначаемый на схемах 1 (рис. 1.6, *г*), выполняет операцию логического сложения входных сигналов с результатом $x \vee y$, называемым дизъюикцией, а при инверсии входа или выхода — логические функции $\bar{x} \vee y$, $x \vee \bar{y}$ или $\overline{x \wedge y}$ соответствеиио (рис. 1.6, *д*). При использовании только одного входа (рис. 1.6, *е*) элемент НЕ реализует операцию логической инверсии входного сигнала.

Соединяя рассмотренные логические элементы, можио реализовать любые сложиые логические операции, а добавляя к таким соединениям времязадающие цепочки — генераторы непрерывных последовательностей (мультивибраторы) или одиночных (одновибраторы) импульсов. Такие импульсы с заданной длительиостью используют в качестве коротких синхронизирующих, фазовых (длительиостью менее такта), тактовых (длительностью один такт) и стробовых (длительиостью несколько тактов) управляющих сигналов.

Соединение логических элементов с инверсиыми выходными напряжениями, «опрокидывающимися» (прииимающими противоположные логические уровни) при подаче иа вход кратковременного входного напряжения, называют триггером. Триггер, выходиое иапряжение которого соответствует логическому уровню поступившего ранее входиого сигнала, используют как одиоразрядиые ячейки памяти для хранения одного бита информации. Соединение n таких триггеров, называемое регистром, обеспечивает храиеиие n-разрядного слова информации, но способы его записи и считывания зависят от вида регистра.

Входы ячеек регистра могут соединяться с виешней информациоииой шиной через логические ключи (иапример, типа И), замыкающиеся при подаче управляющего сигиала. В этом случае n-битовое слово информации, отображенное напряжеиием иа выходах шины, записывается при замыкаиии ключей в ячейки такого регистра в течеиие одиого такта. Аиалогично хранящееся в регистре слово может быть параллельно передано в течеиие одиого такта в информациоииую шииу, выходы которой замыкаются ключами иа выходы ячеек регистра.

Триггериые регистры, выходы ячеек которых через ключи соединеиы со входами следующих ячеек (выход последией и вход первой ячеек соединяются через ключи с внешиими одиовходовыми информациоииыми шииами), называют *сдвиговыми*. При замыкаиии ключей в течеиие первой части (фазы) такта иа входы ячеек сдвигового регистра подаются выходиые иапряжения предыдущих ячеек (иа вход первой ячейки иапряжеиие входиой шины, а иапряжеиие с выхода последией ячейки в выходную линию), но триггеры ячеек в связи с заданиой ииерциоииостью не успевают «опрокииуться». На следующей части (фазе) такта ключи размыкаются, триггеры ячеек «опрокидываются», и иа их выходах устаиавливаются иапряжеиия с логическим уровием входиых напряжеиий. Следовательно, если иа каждом такте подавать фазовые сигналы, замыкающие ключи, то содержащаяся в сдвиговом регистре информация будет перемещаться со скоростью одиого бита за такт и через n тактов содержимое n-разрядиого сдвигового регистра окажется полиостью передаииым в выходную шииу, а в регистре будет записаио слово, последовательно прииимаемое по разрядам с входиой шины. При охвате такого регистра цепью обратиой связи (рис. 1.7, *а*) через n тактов его содержимое будет переписано в регистр-приемиик и восстановится в регистре-передатчике.

С помощью логических элемеитов можио обеспечить работу ячеек сдвигового регистра согласио следующим правилам:

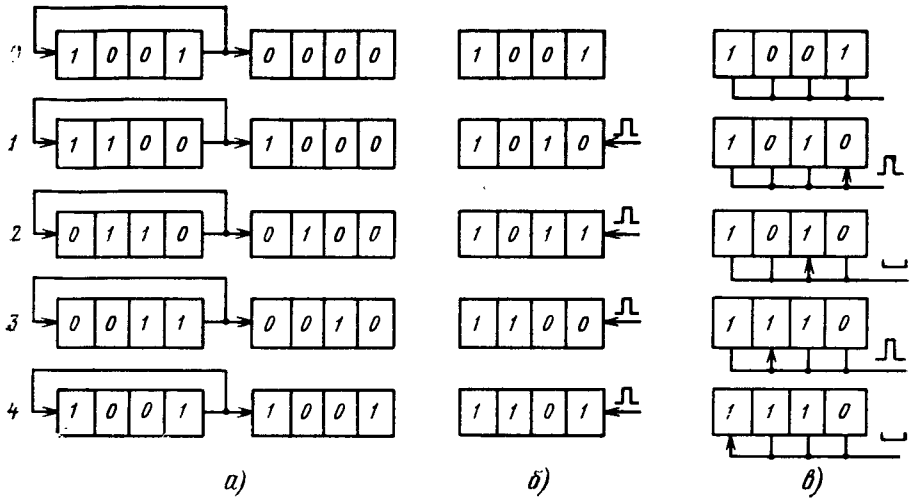


Рис. 1.7

1. При подаче сигнала 0 на вход ячейки с состоянием (уровнем выходного напряжения) 0 оно не изменяется.

2. При подаче сигнала 1 на вход ячейки с состоянием 0 оно переходит в состояние 1.

3. При подаче сигнала 1 на вход ячейки с состоянием 1 оно переходит в состояние 0, а сигнал с уровнем 1 подается на вход следующей ячейки.

Регистры, работающие по этим простым правилам, используют в качестве счетчиков сигналов 1 (например, счетчиков команд). Содержимое такого регистра-счетчика увеличивается на единицу при каждом поступлении входного сигнала логической 1. Так, после подачи четырех таких сигналов на вход регистра-счетчика с состоянием ячеек 1001, соответствующим двоичному коду натурального числа 9, ячейки перейдут в состояние 1101 (рис. 1.7, *б*), равное двоичному коду натурального числа 13 = 9 + 4.

Возможности сдвиговых регистров расширяются при дополнительном выводе через ключи его входных или выходных напряжений, обеспечивающих как последовательный, так и параллельный обмен информацией между регистром и внешними информационными шинами. В частности, при подключении на k-м такте одноразрядной информационной шины к k-й ячейке сдвигового регистра, работающего по приведенным правилам, реализуются функции двоичного сумматора содержимого регистра и передаваемого последовательно (начиная с младших разрядов) по внешней шине двоичного кода числа. Так, если в регистре хранился код 1001 натурального числа 9, то при подаче в шину, последовательно подключаемую к ячейкам регистра, кода 0101 натурального числа 5 через 4 такта содержимое регистра (рис. 1.7, *в*) станет равным коду 1110 натурального числа 14 = 9 + 5.

С помощью логических элементов обеспечивается и выполнение всех операций обработки, временного хранения и передачи информации, включая дешифровку адресов ячеек памяти, информационных шин или входов и выходов различных устройств. В качестве простейшего примера на рис. 1.8 показана схема дешифратора на 8 адресов $a = a_2a_1a_0$ выходов, с которыми соединяется вход x (жирными линиями на рисунке показано соединение этого входа с выходом по адресу a = 011).

Соединения логических элементов, операционных усилителей, резисторов, конденсаторов, транзисторов и диодов, изготовляемых в едином технологичес-

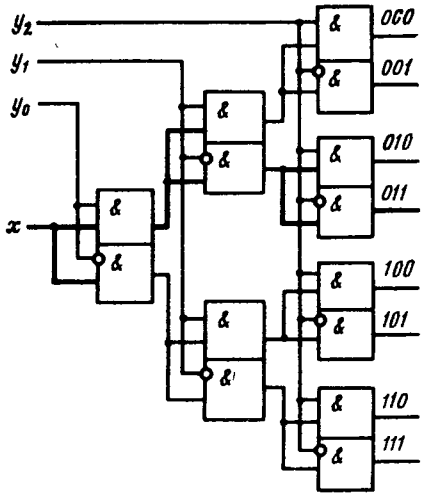


Рис. 1.8

ком процессе на полупроводниковом кристалле, определяют структуру отдельных БИС. Соединение таких БИС между собой, а также с клавиатурой, индикаторным устройством, источником питания и разъемами для присоединения внешних устройств определяет структуру ПМК. В отличие от большинства микроЭВМ других классов, собираемых из микропроцессорных наборов, ПМК каждого типа отличается структурой специализированных БИС или способом их соединения.

1.5. ОСНОВНЫЕ ХАРАКТЕРИСТИКИ

Основными свойствами ПМК, представляющих первостепенный интерес для пользователей, являются производительность и стоимость. Различные группы пользователей предъявляют разные требования к производительности и стоимости, но для каждой из таких групп основным критерием качества ПМК является отношение обобщенной количественной оценки производительности к стоимости. Производительность труда пользователя ЭВМ в общем случае зависит от его умения наиболее эффективно использовать ее и, прежде всего, составлять или применять программы, обеспечивающие решение прикладных задач с минимальными затратами времени. В связи с высокой стоимостью машинного времени производительность труда пользователей стационарных ЭВМ обычно оценивают по производительности этих машин при автоматическом выполнении программы решения прикладной задачи.

Быстродействие универсальных ЭВМ высокой производительности обычно оценивают по числу операций определенного типа, выполняемых в течение секунды. Такая оценка недостаточно объективна в связи с разнообразием выполняемых ЭВМ операций и неодинаковой скоростью выполнения различных операций ЭВМ разного типа. Поэтому часто быстродействие оценивают по затратам времени на решение типовых задач, но такая оценка также в основном связана с выбранными задачами.

Быстродействие микроЭВМ чаще всего оценивают по тактовой частоте (числу тактов в секунду) и разрядности центрального процессора, так как при ее увеличении уменьшаются затраты времени на обработку слов информации, разрядность которых обычно кратна разрядности процессора.

Производительность труда пользователей ПМК в меньшей степени связана с их быстродействием, так как время счета обычно составляет лишь небольшую часть затрат времени на решение прикладных задач. При этом вследствие низкой стоимости машинного времени ПМК ос-

новным экономическим фактором становятся затраты рабочего времени пользователя. Быстродействие ПМК ощутимо в основном лишь при выполнении программ с большим временем счета, определяющим и затраты рабочего времени пользователя.

Скорость выполнения операций передачи и преобразования информации в ПМК определяется главным образом технологией изготовления их элементной базы. В частности, наименьшим быстродействием отличаются БИС, используемые в ПМК первых поколений и изготовленные по МОП-технологии.

Производительность ПМК в значительной степени зависит от емкости памяти, ограничивающей сложность задач, решаемых без разбиения на последовательность более простых задач и соответствующего увеличения затрат времени. В ПМК первых поколений оперативная память, как правило, состоит из фиксированных областей программной памяти и памяти данных. При этом емкость программной памяти чаще всего оценивают по числу ячеек для хранения обычно 8-битовых программных слов (шагов программы), емкость памяти данных — по числу регистров памяти, предназначенных для хранения отдельных чисел. Технической характеристикой ПМК является и разрядность таких регистров, определяющая число десятичных разрядов мантиссы и порядка представления чисел, так как от количества таких разрядов зависит диапазон и точность представления чисел в ПМК.

Существенными для характеристики ПМК как носимых вычислительных средств являются их габаритные размеры, масса и потребление энергии, определяющее число часов работы ПМК в автономном режиме без подключения к внешней цепи питания. Следует добавить, что в ПМК на элементной базе, изготовленной по МОП-технологии, потребление энергии относительно велико и ограничивает возможность питания ПМК от встроенных аккумуляторов или сухих элементов в течение лишь нескольких часов.

Производительность труда пользователей ПМК зависит также от удобства пользования, определяемого особенностями входного языка, устройствами ввода, накопления и отображения информации, а также наличием прикладного математического обеспечения в виде библиотеки или (при наличии накопителей информации) пакетов прикладных программ.

Зависимость производительности труда пользователей от особенностей входного языка в значительной мере определяется субъективными факторами. Поэтому при количественной оценке особенностей входных компактных языков ограничиваются указанием числа элементов алфавита (или клавиш для их ввода), числа и вида стандартных (встроенных) функций, а также указанием способа адресации переходов и максимального числа вложений подпрограмм.

Повышение производительности ПМК при увеличении быстродействия, емкости памяти, ассортимента внешних устройств, расширении прикладного программного обеспечения и усложнении входного языка неизбежно приводит к повышению стоимости ПМК, и, следова-

тельно, его доступности широкому кругу пользователей. Последнее, в свою очередь, приводит к уменьшению объема выпуска ПМК и дополнительно повышает его стоимость, снижающуюся при больших объемах производства. Поэтому основной проблемой разработки ПМК является поиск компромисса между их производительностью и стоимостью.

Решение этой проблемы на определенный период — выпуск широкого ассортимента ПМК различной производительности и, следовательно, различной стоимости. Более общий метод устранения противоречия между стоимостью и производительностью основан на уменьшении аппаратурных затрат на изготовление ПМК при одновременном повышении их производительности за счет совершенствования структур, архитектуры и конструкции ПМК, а также (в особенности) улучшения технологии изготовления их элементной базы.

Эти противоречия возникли еще при разработке первых ПМК, массовое производство которых началось выпуском в 1973 г. фирмой Hewlett-Packard (HP) ПМК типа HP-35. В то время промышленность только начала серийный выпуск полупроводниковых БИС, нашедших первое массовое применение в микрокалькуляторах. Эти микросхемы отличались высокой стоимостью и относительно низкой степенью интеграции, что существенно ограничивало производительность ПМК при стоимости их производства, обеспечивавшей доступность широкому кругу пользователей. В этой ситуации единственным выходом для разработчиков ПМК было сведение к минимуму аппаратурных затрат с обеспечением приемлемой производительности путем рационального выбора входных языков и архитектуры.

Основными из таких решений являлись выбор последовательного способа передачи информации, а также компактных входных языков и индикаторных устройств для визуального отображения информации, аналогичных используемым в настольных ЭКВМ предыдущих поколений. В частности, в качестве основного устройства отображения информации были выбраны индикаторные устройства (дисплеи) с газонаполненными, светодиодными или жидкокристаллическими сегментами, при подаче напряжения на которые формировались десятичные или шестнадцатеричные цифры и другие знаки для представления числовых результатов и кодов вводимых программ.

Последовательная передача n-разрядного двоичного слова по одноразрядной шине в n раз медленнее параллельной передачи этого же слова по n-разрядной шине, но обеспечивает существенное снижение аппаратурных затрат и, следовательно, стоимости изготовления ПМК. Многоразрядные информационные шины занимают значительную часть площади кристалла БИС, что снижает степень их интеграции и приводит к увеличению требуемого числа БИС, а кроме того, требует дополнительных затрат на повышение надежности. Поэтому в ПМК первых поколений использована, как правило, последовательная передача и (во многих случаях) обработка информации, что обеспечило минимальные аппаратурные затраты.

Входные языки ПМК первых поколений, как правило, содержали синтаксические правила обратной (польской) записи расчетных выражений, что позволяло существенно упростить конструкцию ПМК, так как исключало необходимость хранения кодов операторов, выполняемых непосредственно после их считывания из программной памяти. При этом если первые ПМК отличались простейшими версиями компактных входных языков с небольшим числом встроенных функций с прямой адресацией переходов в программах и малой емкостью памяти, то по мере накопления опыта в разработке и производстве технические характеристики ПМК постепенно совершенствовались.

Типичными для первых поколений ПМК с входными компактными языками, отличающимися синтаксическими правилами обратной записи, являются технические характеристики некоторых наиболее распространенных ПМК фирмы HP, указанных в табл. 1.2 в порядке их разработки. В первых из них входные языки имели лишь прямые (П) переходы в программах, в основном небольшую разрядность мантиссы представления чисел, а также небольшую емкость памяти программ и данных. Во входных языках ПМК более поздних моделей возможна как прямая, так и косвенная (К) адресация переходов в программах пользователя. Повышение производительности этих ПМК также было связано с использованием энергонезависимой оперативной памяти, сохраняющей информацию при выключенном питании *. Некоторые ПМК (например, HP-19C) снабжались встроенными миниатюрными термопечатающими устройствами, но такое конструктивное решение оказалось экономически нецелесообразным: существенно повышало стоимость изготовления ПМК. Поэтому последующие модели ПМК (например, HP-67C и его настольный аналог HP-97C) имеют лишь устройства интерфейса для внешних устройств, присоединяемых через разъемы, включая и печатающие устройства. Подобная структура оказалась экономически более выгодной: пользователь мог раздельно приобретать относительно недорогой ПМК и интересующие его внешние устройства.

Значительным шагом в повышении производительности ПМК явилось создание недорогих и достаточно эффективных накопителей на магнитных карточках (HP-67), предназначенных для записи и считывания программ и данных пользователя, а также для хранения пакетов прикладных программ, подготовленных профессиональными программистами. Для считывания или записи информации, хранимой на магнитных карточках, их вставляют в прорезь в торце ПМК и электромеханическое устройство с миниатюрным электродвигателем протягивает карточку, реализуя обмен информацией с оперативной памятью. Применение накопителей на магнитных карточках обеспечило и более

*Буква С в обозначениях ПМК фирмы HP указывает на энергонезависимую оперативную память, а буквой Е обозначены ПМК, в которых для отображения чисел в естественной форме могут дополнительно использоваться разряды, предназначенные для отображения порядка при показательной (с плавающей запятой) форме представления чисел.

Таблица 1.2

Основные технические характеристики ПМК фирмы Hewlett-Packard первых поколений

Тип	HP-33E	HP-38E	HP-55	HP-25C	HP-29C	HP-19C	HP-67
Число разрядов мантиссы/порядка	7/2	7/2	10/2	8/2	8/2	8/2	10/2
Число регистров памяти данных	8	25	20	8	30	30	26
Число ячеек памяти программ	49	99	49	49	98	98	224
Число символов алфавита	50	44	68	72	72	89	84
Адресация переходов в программе	П	П	П	П	П, К	П, К	П, К
Накопитель на магнитных картах	Нет	Нет	Нет	Нет	Нет	Нет	Есть
Печатающее устройство	»	»	»	»	»	Есть	Нет
Габаритные размеры, мм	130×68×30	130×68×30	152×68×30	130×68×30	130×68×30	152×81×34	128×80×15

Таблица 1.3

Технические данные ПМК с «алгебраическим» синтаксисом компактных входных языков

Тип	TI SR-52	TI SR-56	FX-202P	FX-501P	FX-502P	TI P-57	TI P-58	TI P-59
Число разрядов мантиссы/порядка	10/2	10/2	8/2	10/2	10/2	10/2	10/2	10/2
Число регистров памяти данных*	20	10	10	11	22	8	60	100
Число шагов программ*	100	224	127	128	256	150	480	960
Энергонезависимая память	Нет	Нет	Да	Да	Да	Да	Да	Да
Внешние устройства	»	»	Нет	Нет	Нет	Есть	Есть	Есть
Габаритные размеры, мм	90×160×40	80×150×30	104×172××34	71×144×96	71×144×96	147×81×35	164×82××37	164×83×37
Масса, г	350	235	305	103	103	250	240	303
Автономное (А) или сетевое (С) питание	А, С	А, С	А	А	А, С	А, С	А, С	А, С

* Указано максимальное число.

Таблица 14

Основные технические данные массовых ПМК «Электроника» первых поколений

Тип	Б3-21	МК-46	МК-64	Б3-34	МК-56	МК-54	МК-61	МК-52
Число разрядов мантиссы/порядка	8/2	8/2	8/2	8/2	8/2	8/2	8/2	8/2
Число ячеек программной памяти	60	66	66	98	98	98	105	105
Число операционных регистров	2	2	2	5	5	5	5	5
Число символов алфавита	47	48	49	64	64	64	78	80
Адресация переходов программы	П	П	П	П, К	П, К	П, К	П, К	П, К
Внешние устройства	Нет	Есть	Есть	Нет	Нет	Нет	Нет	Есть
Габаритные размеры, мм	186×100×48	280×240×48	280×240×48	185×205×48	208×205×60	167×78×36	167×78×36	212×78×34,5
Масса, кг	0,39	2,5	2,5	0,39	1,3	0,25	0,25	0,25

эффективное использование оперативной памяти большей емкости — не требуется набирать длинную программу с клавиатуры перед каждым ее выполнением.

Компактные входные языки с синтаксическими правилами обратной записи расчетных формул, характерными для ПМК фирмы HP, непривычны для начинающих пользователей, знакомых только с обычной алгебраической записью формул, используемой и в большинстве непрограммируемых микрокалькуляторах. Поэтому ряд зарубежных фирм в ПМК первых поколений стал использовать компактные входные языки с синтаксическими правилами обычной алгебраической записи. Технические характеристики некоторых из наиболее распространенных ПМК с такими входными языками, изготовленных фирмой Texas Instrument (TI) и японской фирмой «Касио», приведены в табл. 1.3. Наиболее совершенными из них являются ПМК типа Р-58 и Р-59 с устанавливаемой пользователем границей между областями программ и данных оперативной памяти. Эти ПМК также снабжены накопителями на магнитных карточках и съемными интегральными модулями для постоянного хранения пакетов прикладных программ.

Массовые отечественные ПМК первых поколений (табл. 1.4), серийное производство которых началось выпуском в 1975 г. ПМК «Электроника Б3-21», характеризуются архитектурными и конструктивными решениями, обеспечивающими минимальные аппаратурные затраты, в частности последовательной передачей и обработкой информации и входными языками с синтаксическими правилами обратной записи расчетных выражений. Подробное описание устройства этих ПМК приведено в следующих главах.

Глава 2

ПРОГРАММИРУЕМЫЕ МИКРОКАЛЬКУЛЯТОРЫ ПОСЛЕДОВАТЕЛЬНОГО ДЕЙСТВИЯ

2.1. ОСОБЕННОСТИ СТРУКТУРЫ И АРХИТЕКТУРЫ

Передача и обработка информации может осуществляться параллельно и последовательно, а также параллельно-последовательно. В подавляющем большинстве ЭВМ используется параллельная передача содержимого всех двоичных разрядов n-битового слова за один такт по n линиям. Этот способ передачи связан со значительными аппаратурными затратами, но они окупаются повышением быстродействия и, следовательно, производительности ЭВМ.

При параллельной передаче информации в микроЭВМ многопроводные информационные шины занимают значительную часть площади кристалла, что приводит к уменьшению степени интеграции микросхем. В этом случае требуется большее число БИС. Особенно нежелательно увеличение аппаратурных затрат для массовых ПМК, повышение сто-

имости которых уменьшает их доступность широкому кругу пользователей, что, в свою очередь, снижает спрос, а значит, дополнительно повышает стоимость производства.

Стоимость БИС первых поколений была относительно высокой, степень интеграции низкой, что, в частности, ограничивало емкость памяти. Поэтому создатели первых карманных ЭВМ вынуждены были выбрать последовательную передачу n-битовых слов за n тактов по одноразрядным шинам. Несмотря на существенное снижение стоимости микросхем в следующие годы, этот способ передачи информации сохранился в большинстве современных массовых ПМК.

Следует добавить, что в микроЭВМ при параллельной передаче используют последовательные или последовательно-параллельные сумматоры с параллельной обработкой содержимого каждых четырех двоичных разрядов. Между тем при последовательной передаче информации достаточно использовать одноразрядные сумматоры, непосредственно обрабатывающие на каждом такте бит принимаемой информации, что позволяет дополнительно уменьшить аппаратурные затраты.

В дальнейшем ПМК с последовательной передачей и обработкой информации будем называть ПМК последовательного действия.

При традиционной вычислительной структуре с раздельными блоками центрального процессора и оперативной памяти повышение быстродействия достигается в основном увеличением тактовой частоты (числа тактов в секунду). Этот путь характерен для большинства зарубежных массовых ПМК.

При создании массовых отечественных ПМК для уменьшения аппаратурных затрат также были выбраны последовательная передача и обработка информации, но повышение быстродействия обеспечивалось конвейерной обработкой текущей информации двумя или более однокристальными микроЭВМ с одноразрядными АЛУ. В отличие от микропроцессоров такие БИС, предназначенные для обработки информации в соответствии с прикладной программой, содержат как процессор, так и оперативную память. На принципиальных схемах подобные микроЭВМ обозначают SMRGCT, указывающим, что они содержат сумматоры SM, сдвиговые регистры RG и счетчики CT, характерные для мини-ЭВМ последовательного действия.

Основными структурными элементами однокристальных микроЭВМ, используемых в ПМК последовательного действия, являются АЛУ в виде одноразрядного сумматора, сдвиговые регистры ОЗУ для хранения текущей информации и ПЗУ для хранения системного программного обеспечения. Несколько сдвиговых регистров для хранения операндов и ячеек для хранения флагов (признаков выполнения определенных условий) образуют *быструю* или *сверхоперативную память* микроЭВМ. Один из сдвиговых регистров через внешние вход и выход соединяется с основной одноразрядной информационной шиной ПМК, называемой *системной магистралью*. Кольцевое соединение через системную магистраль последовательности этих сдвиговых регистров микроЭВМ и сдвиговых регистров ОЗУ (регистровых ОЗУ) образует

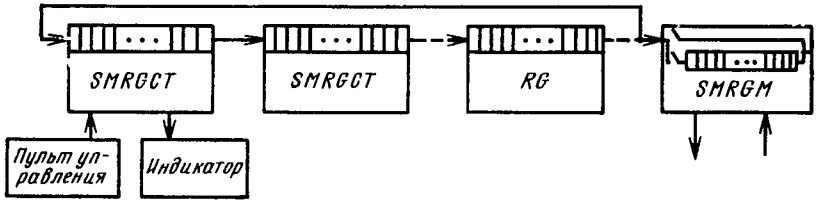


Рис. 2.1

динамическую оперативную память ПМК (рис. 2.1), содержимое которой перемещается на каждом такте на один разряд.

Одна из микроЭВМ является ведущей. Кроме обработки текущей информации, она управляет формированием двоичных кодов исходных данных, операторов прикладной программы и директив (служебных команд), вводимых с пульта управления нажатием клавиш, а также выводом на индикатор результатов операций. Остальные микроЭВМ однотипные, отличающиеся лишь использованием выводов и записанным в ПЗУ программным обеспечением, в основном только обрабатывают текущую информацию в соответствии с прикладной программой.

ПМК, предназначенные для работы с внешним устройством, содержат также контроллеры в виде однокристальных микроЭВМ, управляющих обменом информацией с внешними устройствами, на принципиальных схемах иногда обозначаемых как SMR GM. Регистр оперативной памяти этих микроЭВМ не включен последовательно в системную магистраль ПМК и используется как буферный для временного хранения информации, которой оперативная память обменивается с внешними устройствами. Содержимое оперативной памяти, смещающееся на каждом такте на один бит, когда попадает в приемные ячейки микроЭВМ, может считываться в быструю память или замещаться результатами выполненных в микроЭВМ операций, хранящимися в быстрой памяти.

Таким образом, при последовательной передаче информации в оперативной памяти микроЭВМ информация может обрабатываться практически параллельно (со сдвигом во времени, что необходимо для выполнения операции предыдущей микроЭВМ). Подобная конвейерная обработка информации при прочих равных условиях (в частности, при одинаковой тактовой частоте) обеспечивает значительное повышение быстродействия по сравнению с традиционной однокомпьютерной структурой. При этом, так как адрес каждого двоичного разряда слова в динамической памяти определяется неявно по числу тактов, отсчитываемых от начального момента времени, и повторяется через каждый период полного перемещения информации по кольцу оперативной памяти, уменьшаются аппаратурные затраты — дешифраторы адресов содержимого оперативной памяти можно исключить.

Уменьшение аппаратурных затрат в рассматриваемой многокомпьютерной вычислительной системе сопровождается усложнением архи-

тектуры (в первую очередь обмена информацией между оперативной памятью и процессорами микроЭВМ) и требует четкой системы временны́х координат, обеспечивающей неявную (временну́ю) адресацию информации в быстрой и оперативной памяти при жестком согласовании во времени работы всех логических элементов ПМК.

Выполнение любой операции над содержимым сдвигового регистра в течение одного такта распадается на несколько фаз выборки содержимого и выполнения над ним логических операций. Поэтому на все микроЭВМ и регистровые ОЗУ в течение каждого такта от специально-

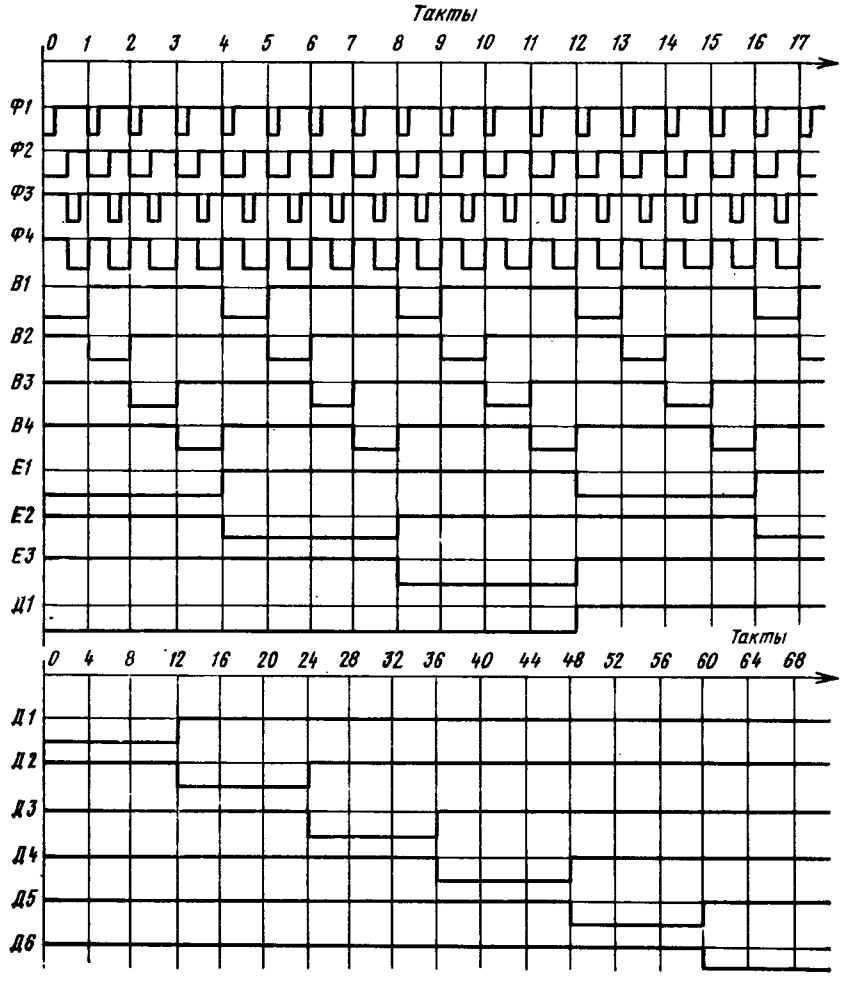


Рис. 2.2

го генератора подаются управляющие фазовые сигналы Ф1, Ф2, Ф3 и Ф4 с периодом повторения, равным длительности такта В (рис. 2.2).

При обработке информации в ПМК существенное значение имеет количество информации, содержащееся в тетрадах (четверках) разрядов слов информации. Содержимое тетрады отображает натуральное число, соответствующее шестнадцатеричной цифре при двоичной системе счисления или десятичной цифре при десятично-двоичной (код 8421 или ДКД). В обоих случаях при обработке информации, содержащейся в тетраде каждого разряда, выполняются одинаковые операции, что упрощает управление работой логических элементов. Различие заключается в шестнадцатеричном переносе символа 1 в старшую тетраду после переполнения старшего разряда тетрады при двоичной системе счисления и десятичном переносе символа 1 в следующую тетраду (когда содержимое данной тетрады превышает натуральное число 9) при десятично-двоичной. Отображение информации на уровне содержимого тетрад шестнадцатеричными или десятичными цифрами упрощает описание выполняемых операций и создание их программного обеспечения.

Для обращения к различным разрядам тетрады используются управляющие (стробовые) сигналы В1, В2, В3 и В4, период повторения которых равен длительности передачи или обработки информации, содержащейся в четырех ячейках сдвигового регистра, Е = 4В. При вычислениях для записи десятичной цифры результата достаточно одной тетрады, но для ее получения в общем случае требуется обработка двух тетрад одного порядка исходных операндов. Для получения результата логических операций в виде одной шестнадцатеричной цифры в общем случае требуется три тетрады. Поэтому для повышения быстродействия обработки слов данных в сдвиговые регистры быстрой памяти микроЭВМ последовательно записывают тетрады десятичного или шестнадцатеричного разряда одного порядка трех различных слов данных. В этом случае выделение каждой из этих трех тетрад обеспечивается управляющими сигналами Е1, Е2 и Е3 с периодом повторения Д = = 3Е. Интервалы времени, соответствующие передаче или обработке содержимого трех очередных тетрад, выделяют управляющими сигналами Д1, Д2, ..., называемыми *разрядными*, длительностью Д = 3Е = = 12В и периодом повторения, определяемым выбранным форматом слов информации.

В ПМК последовательного действия действительные (со знаком и в общем случае дробной частью) десятичные числа представлены в показательной форме с 8 десятичными разрядами мантиссы и двумя разрядами порядка. Эти числа в памяти ПМК отображаются в двоично-десятичном коде (ДКД) двоичными словами с 12 тетрадами разрядов, из которых две выделены для знаков порядка (ЗП) и мантиссы (ЗМ), а остальные — для цифр мантиссы (М) и порядка (П). Избыточность подобного отображения знаков плюс и минус (для чего достаточно одного разряда) обеспечивает однородность операций над разрядами тетрад и равномерность из передачи во времени.

Распределение тетрад от 0 до 11 начиная с младших разрядов между полями слова данных можно представить в виде

$$<PN> = <ЗП(11)> <П\ (10:9)> <ЗМ(8)> <М\ (7:0)>, \quad (2.1)$$

где двоеточием отделены номера крайних тетрад поля, содержащего более одной тетрады. Каждая тетрада слова данных пересылается или обрабатывается в течение интервала Е1, Е2 или Е3 определенного разрядного сигнала Д1, ..., Д12. Соответствие номеров тетрад в формате слова данных (2.1) и разрядных сигналов показано на рис. 2.3. Формат (2.1) слова данных составляет 12 тетрад, и поэтому разрядные сигналы Д1, ..., Д12 повторяются с периодом П = 12Д = 36Е = 144В. Слова данных выбранного разработчиками ПМК формата хранятся в ячейках оперативной памяти, называемых *регистрами данных*. Количество таких регистров определяет максимальное количество слов данных, которое может одновременно храниться в оперативной памяти.

В ЭВМ с традиционной структурой для каждого регистра данных выделена определенная ячейка оперативной памяти, номер которой является адресом хранящегося в ней слова данных. В ПМК последовательного действия содержимое регистров данных вместе со всем содержимым оперативной памяти непрерывно перемещается в сдвиговых регистрах вдоль системной магистрали со скоростью 1 бит за такт. В этом случае адреса регистров (слов) данных определяются неявно по числу тактов, прошедших с начального для цикла перемещения содержимого оперативной памяти момента времени, соответствующего специальному сигналу — коду «метка». Кроме содержимого регистров данных в оперативной памяти ПМК хранятся коды операторов прикладной программы, содержимое регистров операционного стека и служебная информация, обеспечивающая требуемую последовательность выполнения прикладной программы и обмен информацией между микроЭВМ и оперативной памятью.

Оперативная память ПМК последовательного действия разбита на три блока М1, М2 и М3 с одинаковым числом страниц, длина которых при формате слова данных (2.1) равна 12 тетрадам, т. е. 48 бит. Блоки М1 и М3 в основном предназначены для памяти данных $М_д$ и программ $М_р$, а в блоке М2 хранится служебная информация (слова состояния процесса вычислений, счетчика шагов и адресного счетчика и т. д.)

Распределение информации в оперативной памяти и ее объем зави-

ЗП	*Порядок*		*ЗМ*	*Мантисса*							
11	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
Д12	*Д11*	*Д10*	*Д9*	*Д8*	*Д7*	*Д6*	*Д5*	*Д4*	*Д3*	*Д2*	*Д1*

Рис. 2.3

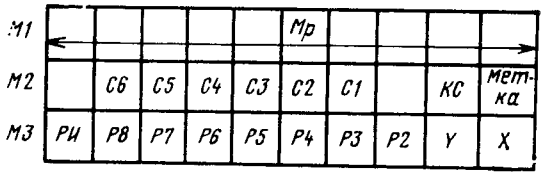


Рис. 2.4

сят от типа ПМК. В первом отечественном ПМК «Электроника Б3-21» каждый блок оперативной памяти разбит на 10 страниц (рис. 2.4).

В блоке М1 на каждой странице хранится по 6 программных 8-битовых слов (шагов программы). В прикладных программах адреса *ab* каждой такой ячейки образуются номером страницы $a = 0, 1, \ldots, 9$ и номером ячейки на этой странице $b = 0, 1, \ldots, 5$. При работе ПМК эти адреса «переводятся» во временные адреса, соответствующие прохождению программных слов через микроЭВМ, реализующую определенный оператор прикладной программы. Когда все 10 страниц области памяти программ заполнены, при вводе избыточных кодов в режиме программирования счетчик шагов продолжает формировать адреса с шестиадцатеричным номером А страницы, который воспринимается как адрес 0, и избыточные коды заносятся на место кодов начальных операторов программы. Поэтому для контроля заполнения области памяти программ содержимое счетчика шагов выводится на индикатор.

В первой странице блока М2 хранится код «метка», от которого отсчитываются неявные (временные) адреса всех разрядов содержимого оперативной памяти. На следующей странице одна из тетрад выделена под канал связи (КС), через который происходит обмен служебной информацией между микроЭВМ вычислительной системы, следующие тетрады выделены под регистры данных С1, ..., С6, образующие вместе с регистром X кольцевой стек памяти данных. Остальные страницы используются для хранения служебной информации.

Блок М3 предназначен для хранения содержимого регистра индикации (РИ), операционных регистров X и Y, а также адресуемых регистров данных с номерами (адресами) 2 ... 8. Содержимое РИ отображается на индикаторе и в ряде случаев отличается от содержимого регистра X. В частности, при вводе программы в режиме программирования в РИ хранятся коды вводимых операторов программы и содержимое счетчика шагов, а в регистре X сохраняется предыдущее содержимое, индицируемое при переходе в рабочий режим.

В остальных ПМК семейства «Электроника МК-64» (кратко рассматриваемых в § 2.2) каждый из трех блоков оперативной памяти содержит по 11 страниц, что позволило увеличить число адресуемых регистров данных до 8 (с адресами от 2 до 9) и число программных слов до 66.

Таким образом, общая емкость оперативной памяти ПМК «Электроника Б3-21» составляет 30 страниц, или $30 \cdot 12 = 360$ тетрад, или $360 \times 4 = 1440$ бит, остальных ПМК семейства «Электроника МК-64»— 33 страницы, или 1584 бит.

Содержимое оперативной памяти образовано последовательностями тетрад, выбранных по одной от каждой страницы блоков М1, М2 и М3, что ускоряет обмен информацией и ее обработку. Так как каждая тетрада пересылается или обрабатывается в течение интервала времени Е = 4В, то каждая из трех очередных тетрад пересылается соответственно в интервалы времени Е1, Е2 и Е3 и, следовательно, каждая тройка тетрад пересылается в течение длительности разрядного сигнала Д1, ..., Д12, а за интервал времени 12Д пересылается три страницы от каждого блока памяти. В пределах каждых трех страниц адрес тетрады, пересылаемой через фиксированную точку системной магистрали, определяется неявно сочетаниями интервалов времени *ДkEi*, а адрес каждого разряда — сочетаниями (временны́м адресом) интервалов времени *ДkEiBj*. После пересылки каждой страницы одного блока памяти (что соответствует пересылке трех страниц, по одной от каждого блока) в интервале времени Д12Е3В4 ведущая микроЭВМ пересылает на входы остальных микроЭВМ короткий синхронизирующий импульс (СИ). Число таких СИ после кода «метка» определяет адрес страницы, а адрес информации внутри каждой такой страницы определяется интервалами *ДkEiBj*. В соответствии с неявными (временны́ми) адресами содержимое оперативной памяти может считываться в любую микроЭВМ или записываться в оперативную память из быстрой (сверхоперативной) памяти микроЭВМ, где происходит обработка текущей информации из оперативной памяти. После каждого полного перемещения содержимого оперативной памяти вдоль кольцевой системной магистрали оно находится в одних и тех же ячейках сдвиговых регистров, что упрощает обмен информацией между микроЭВМ, обрабатывающими информацию, и оперативной памятью.

Время полного перемещения содержимого оперативной памяти определяется ее емкостью и тактовой частотой. Так, время полного цикла ПМК «Электроника Б3-21» с оперативной памятью емкостью 1440 бит при тактовой частоте 100 кГц составляет $1440/10^5 = 14{,}4$ мс, а ПМК с оперативной памятью емкостью 1584 бит при той же тактовой частоте 15,84 мс.

Таким образом, распределение непрерывно перемещающейся информации в динамической оперативной памяти ПМК существенно отличается от распределения данных и программных слов в отдельных ячейках запоминающих устройств, характерных для ЭВМ с традиционной архитектурой. При этом регистры операционного стека также размещены в оперативной, а не в быстрой памяти, что типично для традиционной архитектуры. Однако эти различия практически неощутимы для пользователя, который в обоих случаях для определения результатов пересылки и обработки информации может использовать схему ПМК, показанную на рис. 1.4.

ПМК работают в двух основных режимах — программирования и автоматическом. В режиме программирования нажатием клавиш пульта управления в программную память вводят коды операторов. В режиме автоматической работы нажатием клавиш выполняются вы-

числения, как в непрограммируемых микрокалькуляторах: вводятся в память исходные данные и исполняется введенная в память программа по шагам (при отладке) или автоматически. В обоих режимах нажатием клавиш могут вводиться директивы, управляющие работой ПМК.

При включении питания в слове состояния процесса устанавливаются флаги режима ФР: =0 и исполнения ФИ: = 0, в сдвиговые регистры динамической памяти от ведущей микроЭВМ вводится код «метка» и содержимое этих регистров перемещается.

При установлении флага режима работы ФР : = 0 операторы входного языка, вводимые нажатием клавиш, немедленно исполняются, и результат выполнения операции появляется на индикаторе. При вводе директивы перехода в режим программирования устанавливается флаг ФР : = F, и вводимые коды операторов не исполняются, а засылаются в область памяти программ по адресам, формируемым счетчиком шагов. Одновременно для контроля на индикаторе высвечиваются коды трех программных слов, введенных последними, и в знакоместах порядка — адрес шага программы, вводимого следующим (содержимое счетчика шагов).

После ввода прикладной программы в оперативную память директивой перехода в режим автоматической работы (ФР := 0) можно не только выполнять вычисления нажатием клавиш, но и заносить в память исходные данные для выполнения прикладной программы и исполнять ее в режиме пошагового прохождения (ПП) при отладке или автоматическом. В первом случае после каждого нажатия клавиши ПП из памяти вызывается очередной оператор программы по адресу, равному содержимому счетчика шагов, и на время его исполнения устанавливается флаг ФИ : = F. Во втором случае при вводе директивы С/П (Пуск) флаг ФИ := F устанавливается до считывания из программной памяти кода оператора С/П (Стоп), после исполнения очередного оператора содержимое счетчика шагов автоматически увеличивается на единицу, из программной памяти вызывается и исполняется очередной оператор программы. При считывании кода оператора С/П устанавливается флаг ФИ := 0, что приводит к выводу на индикатор содержимого регистра X.

Таким образом, исполнение программы при нажатии клавиш и при ее считывании из памяти происходит одинаково. Отличие состоит в том, что в программе автоматических вычислений используются операторы перехода, которые при ручных вычислениях человек выполняет в соответствии с переходами в алгоритме решения задач.

При считывании в программе кода проверки условия, являющегося первым шагом оператора условного перехода, проверяется содержимое соответствующих разрядов слова состояния процесса исполнения. Если проверяемое условие не выполнено, то в счетчик шагов засылается адрес перехода, закодированный во втором шаге оператора условного перехода, и из памяти следующим вызывается код оператора с адресом перехода.

Так как содержимое счетчика шагов после выполнения каждого очередного шага автоматически увеличивается на единицу, выполнение программы продолжается с этого оператора.

При вызове из памяти первого шага оператора безусловного перехода БП *ab* следующий за ним адрес перехода засылается в счетчик шагов, и выполнение программы продолжается с этого адреса. При вызове из памяти первого шага оператора перехода к подпрограмме ПП *ab* следующий за ним адрес *ab* также засылается в счетчик шагов, одновременно адресный стек (см. рис. 1.4) смещается «вверх» и прежнее содержимое счетчика, равное адресу второго шага оператора перехода к подпрограмме, засылается в соседний регистр адресного стека. После выполнения подпрограммы с заданного адреса перехода считывание записываемого в конце подпрограммы оператора В/О (Возврат) приводит к смещению адресного стека «вниз» и занесению в счетчик шагов прежнего содержимого соседнего регистра адресного стека, равного адресу второго шага оператора обращения к подпрограмме. После этого содержимое счетчика шагов увеличивается на единицу, и из программной памяти вызывается оператор, записанный в программе после оператора обращения к подпрограмме.

Подобная процедура исполнения операторов ПП *ab* и В/О обеспечивает возможность обращения к одной и той же подпрограмме операторами ПП *ab*, расположенными в различных местах программы. Число обращений из подпрограммы к подпрограмме (вложений подпрограмм) определяется числом регистров адресного стека и в рассматриваемых ПМК равно 5.

Следует добавить, что при отсутствии подпрограмм оператор ПП*ab* исполняется как оператор безусловного перехода общего вида БП *ab*, а при нулевом содержимом «верхних» регистров адресного стека считывание кода оператора В/О приводит к смещению адресного стека «вниз», засылке в счетчик шагов адреса 00, увеличению его на единицу и, следовательно, продолжению выполнения программы с адреса 01.

После ввода программы в оперативную память в режиме программирования содержимое счетчика шагов, на единицу большее адреса последнего шага введенной программы, сохраняется и при переходе в рабочий (автоматический) режим. Поэтому для пуска программы с начального шага следует ввести директиву В/О (Очистка), выполнение которой приводит к присвоению содержимого счетчика шагов РС := 00 и после ввода директивы С/П (Пуск) к выполнению программы с этого адреса. Если требуется запустить программу с заданного адреса *ab*, то перед директивой С/П вводят директиву БП*ab*, исполнение которой приводит к занесению в счетчик шагов адреса *ab*, с которого и начнется выполнение программы при директиве С/П.

При вводе программы в режиме программирования используют директивы $\overrightarrow{\text{ШГ}}$ (Шаг вперед) и $\overleftarrow{\text{ШГ}}$ (Шаг назад), изменяющие на единицу содержимое счетчика шагов и соответственно высвечиваемые на индикаторе коды очередных операторов. Чтобы исключить введение оши-

бочного шага программы, его можно заменить оператором НОП (Нет операции), при исполнении которого содержимое счетчика шагов увеличивается на единицу, что соответствует пропуску шага программы. При необходимости контроля на индикаторе кодов программы, расположенных далеко от высвечиваемых, переходят в режим автоматической работы, вводят директиву В/О или БП*ab* и возвращаются в режим программирования. В этом случае на индикатор высвечиваются коды операторов с заданными адресами.

Попадание результата операции в область машинной бесконечности (переполнение разрядной сетки) и попытка выполнения некорректной операции (деление на нуль или вычисление функции за пределами допустимой области определения аргумента) приводят к прекращению исполнения операторов и выводу на индикатор сигнала ошибки, формируемого в регистре индикации. В этом случае содержимое регистра X сохраняется, и после исправления ошибки вычисления могут продолжаться.

2.2. ПМК СЕМЕЙСТВА «ЭЛЕКТРОНИКА МК-64»

Первый отечественный массовый ПМК «Электроника Б3-21» (рис. 2.5) был разработан в 1975 г. и серийно выпускался с 1977 по 1982 гг. На базе этого карманного ПМК разработаны массовые настольные ПМК «Электроника МК-46» и «Электроника МК-64», предназначенные как для вычислений, так и для работы с внешними аналогово-цифровым преобразователем (АЦП) и цифровыми печатающими устройствами (ЦПУ).

Операционный стек этих ПМК содержит два рабочих регистра *X* и *Y*, а оперативная память имеет различную емкость. ПМК «Электроника Б3-21» имеет 13 регистров данных, в том числе 7 адресуемых с произвольной выборкой (2, ..., 8) и 6, соединеиных вместе с регистром X в двусторонний кольцевой стек памяти с 6 регистрами С1, ..., С6, а также программную память из 60 ячеек для хранеиия 8-битовых программных слов. Оперативная память остальных ПМК семейства содержит дополнительиый регистр данных с адресом 9, но кольцевой стек памяти работает в одностороинем режиме — содержимое регистров пересылается только в одном направлении. Программная память этих ПМК состоит из 66 ячеек.

Пульт управления ПМК этого семейства содержит клавиатуру для ввода директив и операторов программ. Словарный запас входного языка, включая операторы переходов и обращения к памяти только с прямой адресацией, небольшой, синтаксические правила простые — ввод операторов после операидов. Операторы разбиваются на иесколько групп в соответствии с их иазначением.

Для ввода чисел предназначены операторы набора десятичных цифр от 0 до 9, десятичиого разделительного знака (запятой или точки), изменения знака /—/, ввода порядка (ВП) и ввода числа $\pi = 3{,}1415297$. Обращение к памяти даиных обеспечивается операторами PN для засылки копии содержимого регистра X в адресуемый регистр *N* памяти, операторами F*N* для вызова в регистр X копии содержимого адресуемого регистра, а также операторы поворота стека памяти по и против часовой стрелки, обозначенные на клавиатуре (рис. 2.5) кружками со стрелками, которые вводятся после нажатия префиксной клавиши P (во входном языке ПМК «Электроника МК-64» и «Электроника МК-46» только один оператор поворота кольцевого стека памяти). При выполнеиии этих операторов содержимое регистра X засылается в один из крайних регистров стека, содержимое которого смещается иа один регистр, и в регистр X заиосится прежнее содержимое другого крайнего регистра стека.

Изменение содержимого регистров X и Y операционного стека обеспечивают «синтаксические» операторы: ↑ — для засылки содержимого регистра X в регистр Y, $\overrightarrow{XY}$ — для обмена содержимым регистров X и Y и Cx — для стирания содержимого регистра X без изменения содержимого регистра Y. При наборе числа после всех операторов, кроме Cx и ↑, прежнее содержимое регистра X засылается в регистр Y.

Одноместные операторы выполняются над содержимым регистра X без изменения содержимого регистра Y. К таким функциональным операторам относятся F1/x, Fx^2 и $F\sqrt{}$ для вычисления простейших степенных функций, а также операторы Fln, Fe^x, Fsin, Fcos и Fe^{fx}, после выполнения которых в регистры Y и X соответственно засылаются значения sin*x* и cos*x* и для вычисления функции tg*x* достаточно дополнительно ввести оператор ÷ (деления). Аргументы тригонометрических функций должны быть выражены в радианах.

Результат выполнения двухместных операторов +, —, ×, ÷ и x^y над содержимым регистров X и Y засылается в регистр X без изменения предыдущего содержимого регистра Y. Следует добавить, что в программной памяти регистр Y имеет адрес 1 и при вводе оператора F1 копия содержимого регистра Y засылается в регистр X, что соответствует смещению «вниз» содержимого регистров операционного стека.

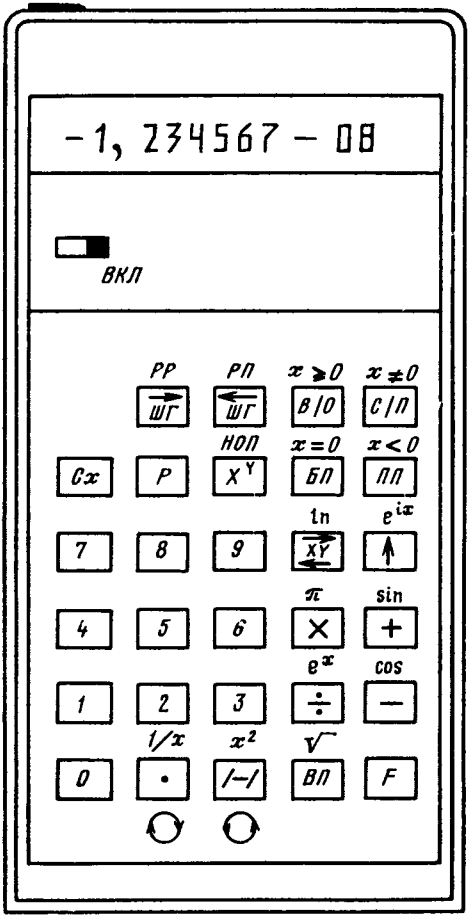


Рис. 2.5

Все рассмотренные операторы могут использоваться как при вычислениях нажатием клавиш, так и программах автоматических вычислений, хранимых в программной памяти. Для управления ходом выполнения программы автоматических вычислений используются операторы: С/П (Стоп), В/О (Возврат из подпрограммы), F НОП (Нет операции) для пропуска шага программы, БП *ab* для безусловного перехода по адресу *ab*, ПП *ab* для перехода к подпрограмме по адресу *ab* ее оператора, а также операторы условных переходов $x < 0$ *ab*, $x = 0$ *ab*, $x \geqslant 0$ *ab* и $x \neq 0$ *ab* с переходом по адресу *ab* при невыполнении проверяемого условия для содержимого регистра X.

Структура ПМК на уровне неделимых компонентов определяется его принципиальной схемой. Так как схемы *вычислительного блока* ПМК рассматриваемого семейства различаются незначительно, то ограничимся рассмотрением его принципиальной схемы для ПМК «Электроника МК-64» (рис. 2.6), указав ее отличия от схем других ПМК этого семейства.

Оперативная память образована последовательностью замкнутых в кольцо через системную

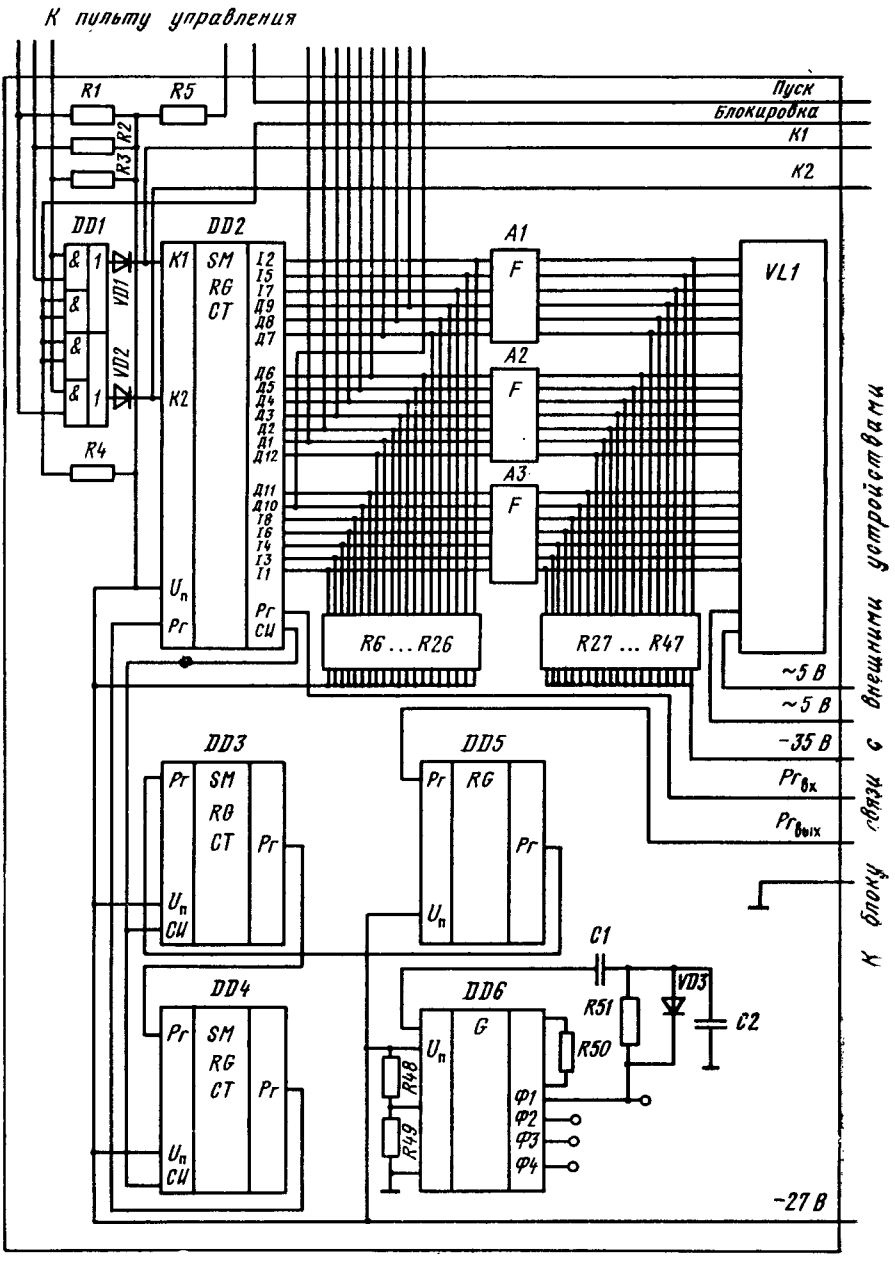

К пульту управления
R1
R5
R2
R3
Пуск
Блокировка
К1
К2
DD1
DD2
A1
A2
A3
VL1
VD1
VD2
R4
R6 ... R26
R27 ... R47
~5 В
~5 В
-35 В
К блоку связи с внешними устройствами
DD3
DD5
DD4
DD6
C1
VD3
R51
C2
R50
R48
R49
-27 В

Рис. 2.6

магистраль сдвиговых регистров ведущей микроЭВМ DD2 (К145ИК502П), регистрового ОЗУ DD5 (К145ИР1П), а также двух микроЭВМ DD3 (К145ИК501П) и DD4 (К145ИК503), отличающихся от ведущей лишь содержащимся в них программным обеспечением различных операций и использованием внешних выводов. Сдвиговые регистры этих микроЭВМ включены в системную магистраль через внешние входы и выходы, обозначенные на схеме Рг. В разрыв системной магистрали между DD3 и DD5 включена находящаяся в цифровом блоке управления работой внешних устройств микроЭВМ (К14ИК181П), выполняющая функции контроллера.

На каждую из этих БИС подается напряжение питания $U_п$ = —27 В, а также фазовые сигналы Ф1, Ф2, Ф3 и Ф4 (для упрощения схемы шины фазовых сигналов не показаны) от генератора фазовых сигналов DD6 (К165ГФ2). Длительность фазовых сигналов и период их повторения, равный такту работы ПМК, определяются параметрами времязадающей цепи из резисторов R50 (30 кОм), R51 (100 кОм), коиденсаторов С1 и С2 (емкостью 1500 пФ каждый) и диода VD3 (КД522Б). Резисторы R48 (2 кОм) и R49 (2 МОм) образуют делитель напряжения для дополнительного питания генератора фазовых сигналов.

Разрядные сигиалы Д1...Д10 ведущей микроЭВМ DD2 подаются на 10 входных шин клавиатуры пульта управления (рис. 2.7). При нажатии клавиши соответствующий разрядный сигнал подается на одну из трех выходных шин, соединеиных со входами микросхемы DD1 (К172ЛИ1), которая содержит два логических элемента И—ИЛИ (рис. 2.6). С выхода микросхемы DD1 разрядный сигиал через разделительные диоды VD1 и VD2 (КД522Б) подается иа одии или оба входа К1 и К2 ведущей микроЭВМ, формирующей коды входных сигналов, директив и операторов программ. Кроме клавишей для ввода символов пульт управления содержит клавишу ПУСК для включения блока управления внешними устройствами. Выходные шины клавиатуры и цепь клавиши ПУСК через резисторы R1...R5 (100 кОм) соединены с шиной постоянного напряжения —27 В.

С выхода ведущей микроЭВМ DD2 разрядные сигналы Д1,...,Д12 и сегментные сигналы I1,...,I8 через усилители А1, А2 и А3 (К161КН1) подаются на индикаторное устройство VL1 типа ИВЛ1-8/13. Все выходные шиины этой микроЭВМ через блок резисторов R6...R26 (68 кОм) соединены с шиной иапряжения —27 В. Так как напряжение импульсных сигналов Д1, ..., Д12 и I1,...,I8 около 27 В, то при их подаче на соответствующих шинах устанавливается иапряжеиие, близкое к нулю отиосительно корпуса.

Иидикаторное устройство содержит 12 знакомест, из которых 2 предназначены для отображения отрицательиых знаков мантиссы и порядка, 2 — для цифр порядка и 8 — для цифр мантиссы числа, содержащегося в регистре иидикации. Каждое знакоместо представляет собой газоразрядный трехэлектродиый

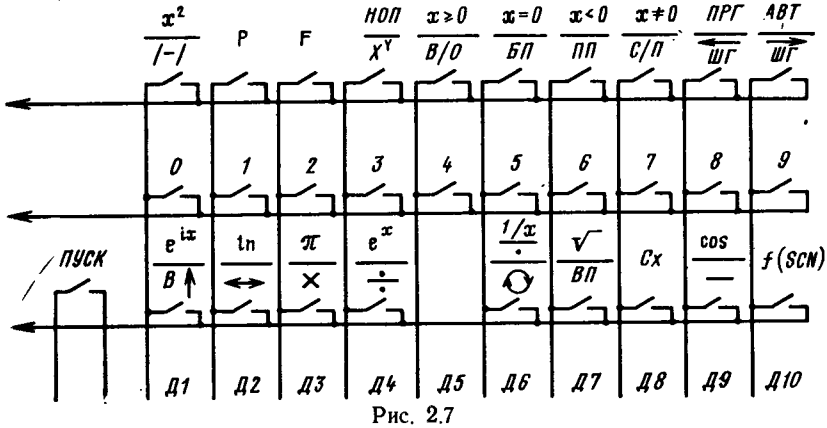


Рис. 2.7

прибор, катодом которого служит нить накала, на сетку подается один из разрядных сигналов, а функции анодов выполняет 8 сегментов, отображающие на индикаторе элементы десятичных знаков. Так как числа на индикаторе отображаются в форме, аналогичной показательной форме записи чисел, то назначение знакомест, пронумерованных справа налево цифрами от 0 до 11, отличается от назначения тетрад в регистре данных формата (2.1):

$$<\text{ИУ}> = <\text{ЗМ}(11)><\text{М}(10:3)><\text{ЗП}(2)><\text{П}(1:0)>. \qquad (2.2)$$

В соответствии с этим форматом на сетку знакомест и подаются разрядные сигналы Д2,...,Д13 (рис. 2.8, *а*), определяющие содержимое регистра индикации (рис. 2.4), отображаемое на индикаторе. Каждое знакоместо содержит 7 сегментов для отображения цифр, на которые соответственно подаются сигналы I1,...,I7 (рис. 2.8, *б*), и отдельный сегмент для отображения запятой при подаче сигнала I8. Все однотипные сегменты знакомест индикаторного устройства соединены между собой, и определенный сегментный сигнал подается на все эти сегменты одновременно.

Входы сегментных и разрядных сигналов индикаторного устройства соединены через блок резисторов R27...R47 (68 кОм) с шиной питания —35 В. Напряжение питания нити накала индикаторного устройства переменное 5 В. Она соединена также через двусторонний стабилитрон VD11 (КС170А) с шиной питания —35 В (в цифровом блоке). Поэтому при отсутствии разрядного или сегментного сигнала на соответствующем входе индикаторного устройства устанавливается напряжение, близкое к нулю относительно корпуса. При подаче сегментного или разрядного сигнала с выхода усилителя на этом входе индикаторного устройства устанавливается близкое к нулю напряжение относительно корпуса и напряжение около 35 В относительно нити накала. При одновременной подаче этих сигналов на сетку и сегмент знакоместа возникает светящийся газовый разряд вблизи соответствующего сегмента.

Разрядные сигналы Д1,...,Д12 подаются на сетки знакомест в интервалы времени, соответствующие этим сигналам, а сегментные сигналы — на однородные сегменты знакомест в течение длительности соответствующего разрядного сигнала. Для формирования запятой на одно из знакомест мантиссы подается сигнал I8, а формирование остальных сегментов зависит от вида отображаемого знака. Так, для отображения отрицательного знака подается сегментный сигнал I4 в интервалах разрядных сигналов Д9 или Д12. Подобным образом отображаются и десятичные цифры 0 (сегментные сигналы I1, I2, I3, I5, I6, I7), 1 (сигналы I3, I6), 2 (I1, I3, I4, I5, I7), 3 (I1, I3, I4, I6, I7), 4 (I2, I3, I4, I6), 5 (I1, I2, I4, I6, I7), 6 (I1, I2, I4, I5, I6, I7), 7 (I1, I3, I6), 8 (I1, I2, I3, I4, I5, I6, I7), 9 (I1, I2, I3, I4, I6, I7).

При попадании результата вычислений в область машинной бесконечности (переполнение регистров данных) и попытке выполнить некорректную операцию (например, деление на 0 или вычисление логарифма отрицательного числа) на индикаторе формируется сигнал ошибки в виде нуля в знакоместе знака мантиссы.

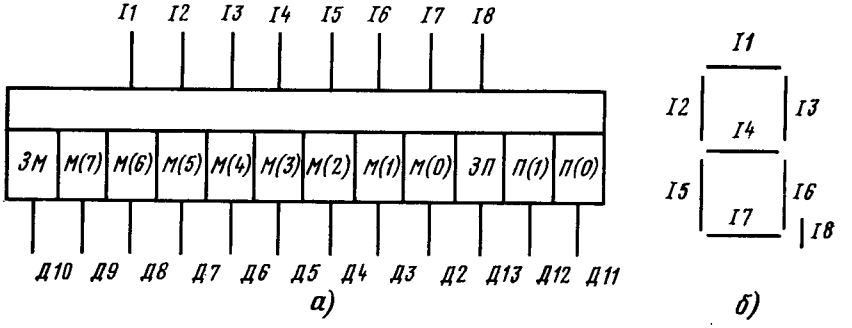


Рис. 2.8

Схема вычислительного блока настольного ПМК «Электроника МК-46» такая же, но отличается от схемы этого блока ПМК «Электроника Б3-21». В последием использовано малогабаритное светодиодное (красное свечение) или газонаполиенное (зеленое свечение) индикаторное устройство с меньшим напряжением питания сетки и сегментов знакомест. В связи с этим ведущая микроЭВМ непосредственно соединена с соответствующими входами индикаторного устройства и отсутствуют не только усилители А1, А2 и А3 и блок резисторов R27... ... R47, но и шина питания —35 В.

В некоторых ПМК этого семейства использовано индикаторное устройство, в котором сегмент запятой расположен в середине знакомест. Так как одно знакоместо отводится под запятую, высвечивается только 7 цифр дробной мантиссы, восьмая цифра мантиссы сохраняется в регистре индикации и при необходимости ее можно вызвать на индикатор, устранив старшую цифру мантиссы. Например, при вызове числа π на индикаторе высвечивается 3,141529, но при дополнительном вычитании числа 3 — число 1,415297 — 01 с искомой восьмой цифрой.

Кроме того, символы операторов на клавиатуре ПМК «Электроиика Б3-21» (рис. 2.5) иесколько отличаются от символов на клавиатуре остальных ПМК семейства (ср. рис. 2.5 и 2.7). В частиости, входиой язык ПМК «Электроника Б3-21» содержит операторы поворота кольцевого стека памяти по и против часовой стрелки, но ие содержит оператора f(SCN), отсутствует и клавиша ПУСК, так как этот ПМК ие предиазначен для работы с виешиими устройствами.

Источником автономного питания ПМК «Электроиика Б3-21» является батарея из 4 аккумуляторов типа Д055С общим напряжением 4 В. Для питаиия от сети переменного напряжения 220 В, 50 Гц и подзарядки аккумуляторов предназначен съемный блок питания (внлка-выпрямитель) типа БП2-3 с переключателем для установки режима питания или подзарядки. Остальные ПМК семейства питаются только от сети переменного напряжения через встроенный блок питаиия.

Блок управления (цифровой) работой внешних устройств (рис. 2.9) ПМК «Электроника МК-64» содержит микроЭВМ DD10 (К145ИК1801), включениую в разрыв системной магистрали и выполняющую функции контроллера внешних устройств. На входы этой микроЭВМ подаются сигнал системной магистрали (Рг), синхросигнал (СИ) от ведущей микроЭВМ, фазовые сигиалы с выходов генератора фазовых сигналов DD6 вычислительного блока (шины фазовых сигналов не показаны), а также сигналы Р1 ... Р12 с выхода внешиего АЦП. Входы сигиалов Р9...Р12 через разделительные диоды VD7...VD10 (КД522Б) соединены с выходом синхрогенератора DD9 (КР127ГФ1Г). На вход генератора, зашунтированный резистором R53 (100 кОм), при нажатии клавиши ПУСК подается напряжение —27 В, запускающее генератор, ко входам которого присоединены резисторы R54 и R55 (1 МОм), образующие цепь делителя напряжеиия —27 В, и конденсатор С3 (6800 пФ). На все выходы микроЭВМ DD10 через блок резисторов R58...R69 (100 кОм) подается напряжение —27 В. Это напряжение шунтировано конденсаторами С5 (20 мкФ), С6 и С7 (0,047 мкФ).

Выходные сигналы микроЭВМ DD10 подаются на разъемы выходных устройств, а сигнал У11 через кондеисатор С4 (0,68 мкФ) на вход микросхемы DD7, шунтируемый резистором R52 (100 кОм) и диодом VD6 (КД522Б). При совпадении сигнала У11 с выхода микроЭВМ DD10 и разрядного сигнала Д8 от ведущей микроЭВМ вычислительного блока микросхемы DD7 и DD8 (К172ЛК1), содержащие два логических элемента ИЛИ—И, формируют выходной сигнал. Этот выходной сигнал через разделительные диоды VD4 и VD5 (КД522Б) подается непосредственно на входы К1 и К2 ведущей микроЭВМ. Выходные сигналы У6, У7, У8 и У12 подаются на входы дешифратора DD11 (К501КД1П), питаемого от делителя напряжения на резисторах R56 (12 кОм) и R57 (10 кОм).

Выходные сигналы дешифратора подаются (совместно с сигналом от генератора DD9) на входы микросхем DD12, DD13 и DD14 (К172ТР1), каждая из которых содержит две триггерные ячейкн с выходными логическими элементами ИЛИ. Выходные сигналы этих микросхем подаются на входы индикаторного устройства VL2 (типа ИВ-26) для контроля работы внешнн устройств и на разъ-

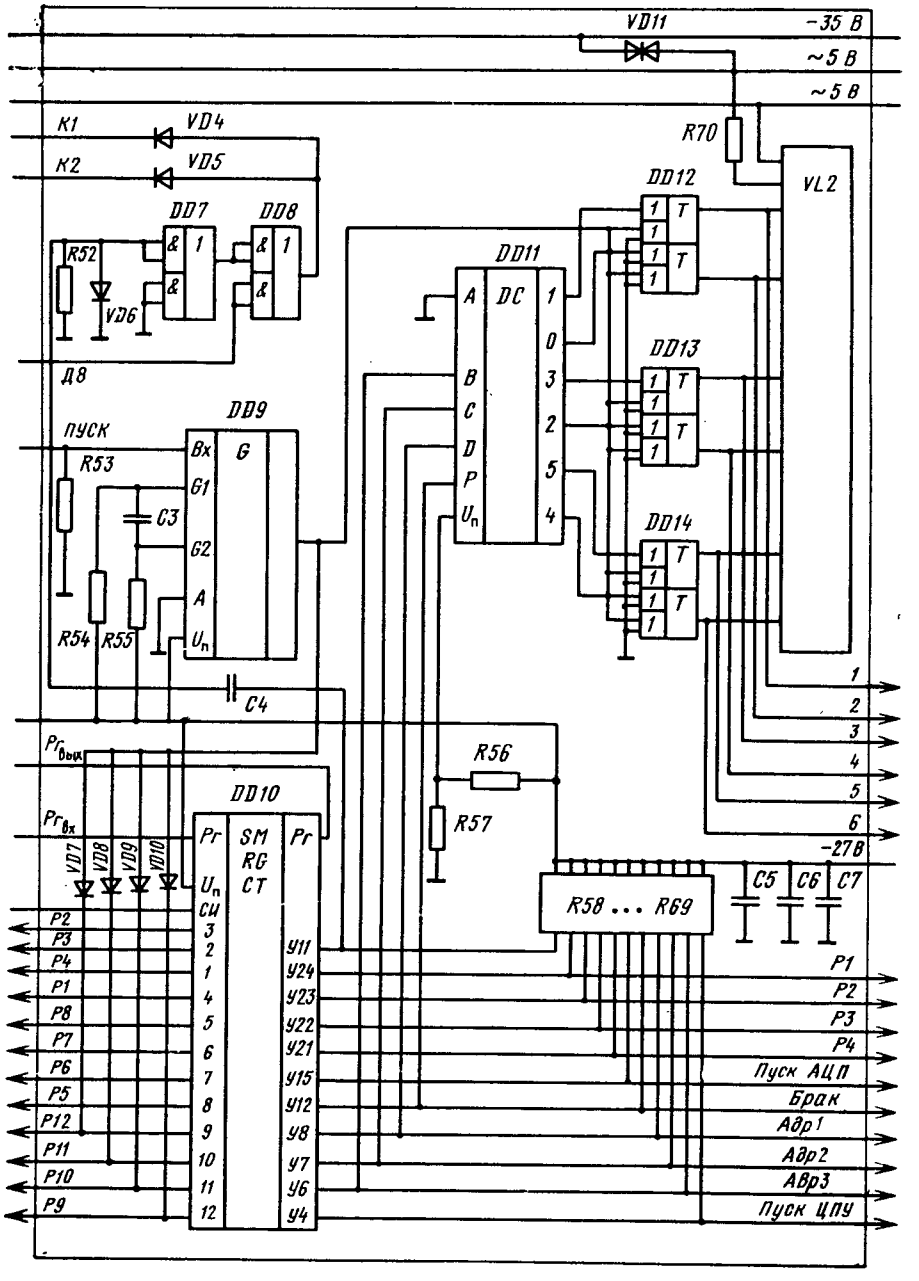

VD11
-35 В
~5 В
~5 В
K1
VD4
K2
VD5
R70
VL2
DD12
DD7
DD8
R52
VD6
Д8
DD11
DD13
DD14
пуск
DD9
R53
C3
R54
R55
C4
R56
R57
DD10
Pr вых
Pr вх
VD7
VD8
VD9
VD10
R58 ... R69
C5
C6
C7
-27В
Пуск АЦП
Брак
Адр1
Адр2
Адр3
Пуск ЦПУ

Рис. 2.9

ем управляющих сигналов внешних устройств. Нить накала индикаторного устройства соединена с шиной —35 В через стабитрон VD11 (КС170А), напряжение питания переменное ~5 В через резистор R70 (43 Ом).

Основные различия между ПМК «Электроника МК-46» и «Электроника МК-64» связаны с конструктивным оформлением блока АЦП. В первом используется отдельный блок АЦП, а во втором этот блок оформлен в виде съемной кассеты.

Блок АЦП (рис. 2.10) содержит коммутатор А1, сравнивающее устройство А2, управляющее устройство А3, формирователи положительного А4 и отрицательного А5 ступенчатых напряжений, а также 12-разрядный десятично-двоичный счетчик А6.

Блок АЦП предназначен для преобразования аналоговых (непрерывных) сигналов напряжением от 0 до ±9,99 В и длительностью не менее 15 мс в последовательности подаваемых на ПМК двоично-десятичных кодов, отображающих отсчеты напряжения трехзначными десятичными числами. Это преобразование реализуется сравнением напряжения через определенные промежутки времени с эталонным ступенчатым напряжением соответствующей полярности.

На входы коммутатора с входным сопротивлением не менее 2 МОм и входной емкостью не более 100 пФ подаются до 8 различных аналоговых сигналов. Один из них в соответствии с адресным кодом, образованным сигналами Адр1, Адр2 и Адр3, формируемыми контроллером DD10 в цифровом блоке управления ПМК, подается на сравнивающее устройство А2, которое содержит компараторы DA1 и DA2 для сравнения входного сигнала с отрицательным или положительным эталонным ступенчатым напряжением. Эти эталонные напряжения снимаются с выходов формирователей А4 и А5. Блок А5 содержит 10-разрядный счетчик DA6, содержимое которого параллельным кодом пересылается в цифроаналоговый преобразователь (ЦАП) А3, который при каждом увеличении содержимого счетчика на единицу формирует очередное приращение («ступеньку») эталонного ступенчатого напряжения, которое устанавливается потенциометром RP1. Линейность огибающей ступенчатого напряжения обеспечивается цепью отрицательной обратной связи, охватывающей операционный усилитель DA4,с выхода которого снимается положительное ступенчатое напряжение. Отрицательное ступенчатое напряжение снимается с выхода операционного усилителя DA5 и корректируется потенциометром RP2.

Работой блока АЦП управляет цифровой блок управления ПМК, сигналы с которого подаются на управляющее устройство А3. При подаче на это устройство сигналов «Гот. ВУ» (Готовность внешних устройств) и «Пуск АЦП» уровня логической 1 импульсные фазовые сигналы Ф1 от генератора DD6 в

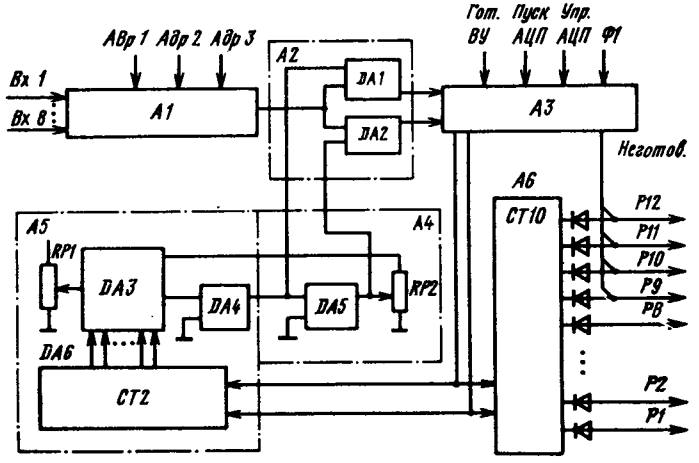


Рис. 2.10

вычислительном блоке ПМК подаются на входную шину «счет» счетчиков DA6 и A6, в первом из них число импульсов Ф1 отображается в двоичном коде, а во втором — в двоично-десятичном. Одновременно формируется сигнал «Неготовность», блокирующий передачу содержимого счетчика A6 на входы микроЭВМ DD10.

Когда эталонное ступенчатое напряжение, формируемое синхронно с увеличением содержимого счетчика DA6, достигает уровня входного аналогового сигнала от устройства A2, на вход управляющего устройства A3 подается сигнал, запрещающий подачу импульсов Ф1 на шину «счет» и снимающий сигнал «Неготовность». При этом содержимое счетчика A6, отображающее двоично-десятичным кодом (соответствующим числу с тремя десятичными разрядами) напряжение входного сигнала, пересылается на входы микроЭВМ DD10. Эта микроЭВМ вырабатывает сигнал «Упр. АЦП» (Управление АЦП) уровня логической 1, поступление которого на устройство A3 приводит к прекращению работы АЦП на время пересылки содержимого счетчика A6 для устранения искажений передаваемой информации. Если внешнее устройство не готово к выдаче информации, то после приема кодов входного напряжения микроЭВМ DD10 сигнал «Гот. ВУ» имеет уровень логического 0, что приводит к пересылке импульсов Ф1 на шину «сброс» для очистки содержимого счетчиков DA6 и A6. При поступлении сигналов «Пуск АЦП» и «Гот. ВУ» уровня логической 1 импульсы Ф1 начинают подаваться на шину «счет», и цикл преобразования отсчета входного напряжения в цифровой код повторяется. Если входы «Упр. АЦП» и «Гот. ВУ» не используются, то соответствующие управляющие сигналы формируются в устройстве A3.

Время преобразования отсчета входного напряжения зависит от его уровня и тактовой частоты. При тактовой частоте 80 кГц это время составляет от 0 до 14 мс. Если входное напряжение не превышает 4 В, используется синхронный режим — цифровые коды отсчетов пересылаются на входы БИС DD10 примерно через каждые 5 мс после поступления сигнала «Пуск АЦП». Если напряжение больше 4 В, используется асинхронный режим, при котором время цикла работы АЦП в зависимости от напряжения составляет от 5 до 15 мс. Асинхронный режим гарантирует получение достоверной информации, так как цифровой код очередного отсчета пересылается с микроЭВМ DD10 только после завершения преобразования входного аналогового сигнала.

Управление работой цифрового печатающего устройства (ЦПУ) зависит от его типа и описано в руководстве по использованию ЦПУ.

2.3. ПМК РАСШИРЯЮЩЕГОСЯ РЯДА

Опыт создания и производства ПМК семейства «Электроника МК-64» был использован при разработке ПМК расширяющегося ряда. Использование БИС с большей емкостью запоминающих устройств обеспечило возможность увеличения емкости оперативной памяти при организации операционного стека с пятью регистрами X, Y, Z, T и дополнительным регистром X1 для хранения предыдущего (до выполнения операции) содержимого регистра X. Это же обеспечило возможность описания программ решения прикладных задач на более совершенных компактных входных языках высокого уровня с достаточно большим словарным запасом при использовании не только прямой, ио и косвенной (шестнадцатеричной) адресации регистров данных и программных слов.

Структурная схема, показанная на рис. 2.1, дополнена включенными в системную магистраль двумя или более микроЭВМ К145ИК13 (К745ИК13) и одним или более регистровым ОЗУ К145ИР2 (К745ИР2), а также при использовании внешних устройств одним или более конт-

роллерами обмена информацией между системной магистралью с внешними устройствами на микроЭВМ К145ИК18 (К745ИК18).

Минимальная структура этих ПМК соответствует включению в системную магистраль двух микроЭВМ К145ИК13 (К745ИК13) и одного регистрового ОЗУ К145ИР2 (К745ИР2). Увеличение числа таких микроЭВМ обеспечивает увеличение емкости динамической памяти, числа АЛУ при параллельной обработке информации и в связи с увеличением суммарной емкости ПЗУ увеличение словарного запаса входного языка при сохранении неизменными синтаксических правил. Это обеспечивает возможность расширения ряда этих ПМК при совместимости снизу вверх (от минимальной структуры) его архитектуры. В частности, все прикладные программы, составленные на входных языках младших членов этого ряда, выполнимы на ПМК с более сложной структурой и большим словарным запасом входных языков.

Выполнение операторов прикладных программ, вводимых нажатием клавиш при соблюдении правил входного языка или считыванием из программной памяти, выполняется в следующей последовательности:

1) формирование или считывание из программной памяти кода оператора ведущей микроЭВМ, связанной с пультом управления;

2) изменение состояния регистра-счетчика программных слов (шагов программы) для указания адреса следующего кода оператора;

3) определение вида операции;

4) пересылка кода операции в микроЭВМ, содержащей в ПЗУ программное обеспечение выполнения соответствующей операции;

5) выполнение операции;

6) пересылка результата операции в регистр X операционного стека, содержащегося в оперативной памяти, с изменением при необходимости содержимого остальных регистров операционного стека.

При выполнении операторов, вводимых нажатием клавиш, или при считывании из программной памяти кода оператора С/П содержимое регистра X пересылается в регистр индикации, и ведущая микроЭВМ вырабатывает сигналы, управляющие отображением на индикаторе результата операции в десятичном или шестнадцатеричном коде.

При вводе директив с пульта управления ведущая микроЭВМ формирует их коды, в соответствии с которыми выполняется требуемая операция изменения режима работы ПМК. В режиме программирования на индикаторе высвечивается содержимое счетчика шагов в общем случае в шестнадцатерично-десятичном коде, соответствующем адресу в программной памяти следующего программного слова, и в шестнадцатеричном коде трех программных слов, введенных последними. В отличие от ПМК семейства «Электроника МК-64» в ПМК расширяющегося ряда адреса переходов набираются с клавиатуры символами десятичных или (в ПМК с большей емкостью памяти программ) шестнадцатеричных цифр. Для проверки программы может использоваться директива $\overrightarrow{\text{ШГ}}$ или $\overleftarrow{\text{ШГ}}$, изменяющая на единицу содержимое счетчиков шагов и соответственно содержимое регистра индикации. Для установ-

ки содержимого счетчика шагов, равного начальному адресу 00 или произвольному адресу *ab*, достаточно после директивы FABT для перехода в рабочий режим ввести соответственно (как и в ПМК семейства «Электроника МК-64») директиву В/О (Очистка) или БП*ab* (Безусловный переход по адресу *ab*). Эти же директивы используют и перед пуском программы автоматических вычислений с нужного шага директивой С/П (Пуск).

Для отладки программы автоматических вычислений при ее пошаговом исполнении могут также использоваться директивы ПП (Пошаговое прохождение), СF для устранения последствий ошибочного нажатия клавиши F и некоторых других клавиш, FПРГ для перехода в режим программирования.

В ПМК с внешними накопителями информации используются также директивы А↑ для засылки в контроллер адреса обращения к фрагменту информации в памяти накопителя и ⇅ для обмена информацией между памятью накопителя и оперативной памятью ПМК. При этом адрес участка памяти накопителя задается набором последовательности десятичных цифр ЦАААДД, где Ц — любая значащая (отличная от нуля) цифра, АААА — разряды адреса начальной тетрады требуемого участка памяти накопителя информации; ДД — длина этого участка в байтах (число программных слов), кратная числу 7 (если это число задано другим, то контроллер увеличивает его до ближайшего целого числа, кратного 7).

Кроме выключателя питания на пульте управления ПМК есть переключатель Р—Г или Р—ГРД—Г для установки размерности аргумента тригонометрических функций и значений обратных тригонометрических функций в радианах (Р), градусах (Г) или десятичных градусах — градах (ГРД), соответствующих делению прямого угла на 100 равных частей и связанных с другими единицами размерности отношениями ГРД = 0,9 Г = 200 Р/π.

Шестнадцатеричные цифры А, В, С, Д и Е на индикаторе высвечиваются (рис. 2.8, *б*) соответственно как — (сегментный сигнал I4), L (сегментные сигналы I2, I5 и I7), С (сегментные сигналы I1, I2, I5, I7), Г (сегментные сигналы I1, I2, I5) и Е (сегментные сигналы I1, I2, I4, I5, I7), а цифра F не высвечивается, обеспечивая формирование пробела. Поэтому сигнал ошибки ЕГГОГ соответствует содержимому (шестнадцатеричному) регистра индикации EDD0DFFF. Этот сигнал высвечивается (как и 0 в знакоместе мантиссы ПМК семейства «Электроника МК-64») при переполнении регистра Х и попытке выполнения некорректной операции. В последнем случае содержимое регистра Х сохраняется, и после устранения причины ошибки вычисления могут быть продолжены.

В карманных ПМК расширяющегося ряда использовано индикаторное устройство типа ИВЛ2-8/12, схема которого с номерами выводов электродов показана на рис. 2.11.

Связь между микроЭВМ, включенными в системную магистраль, осуществляется в интервалы времени, соответствующие прохождению через микроЭВМ участка оперативной памяти, выделенного под канал

связи. Абонент — микроЭВМ системной магистрали — при обращении к другому абоненту передает в канал связи его адрес и код операции. Абонент-приемник, выполнив операцию, передает в канал связи подтверждение о ее выполнении. При этом абоненты различаются по адресам, а не по месту включения в магистраль.

Во время выполнения операции одной микроЭВМ другая выбирает из оперативной памяти код следующей операции, что повышает быстродействие. Этой же цели служит и освобождение (очистка) канала связи до полного завершения выполняемой операции, что обеспечивает возможность связи между другими микроЭВМ и повышает эффективность их параллельной работы. МикроЭВМ, включенная в системную магистраль, может перевести ее в режим ожидания и обеспечить себе доступ к управлению той частью вычислительной системы, которая получает коды операторов по каналу связи.

Согласование во времени (интерфейс) обмена информацией между системной магистралью и внешними устройствами обеспечивает микроЭВМ К145ИК18 (К745ИК18), сдвиговые регистры которой принимают от системной магистрали или передают в нее информацию через внешние вход и выход. В качестве внешних устройств могут использоваться ПЗУ для хранения прикладных программ, накопители для длительного хранения программ и данных и ОЗУ для расширения оперативной памяти ПМК, а также ЗУ печатающих и других внешних устройств преобразования информации. В этом случае для обеспечения обменом информацией с оперативной памятью ПМК используют эмуляторы накопителей внешних запоминающих устройств (ВЗУ) в виде одномерных массивов управляющей информации. Функции эмуляторов накопителей ВЗУ могут выполнять и микроЭВМ К145ИК13 (К745ИК13) с соответствующим содержимым ПЗУ. Такие микроЭВМ вместе с контроллерами на микроЭВМ К145ИК18 (К745ИК18) обеспечивают отображение адресного пространства всей памяти вычислительной системы на адресные пространства ЗУ.

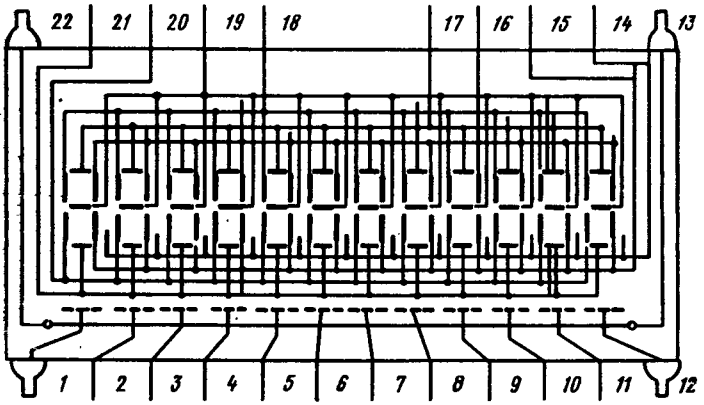


Рис. 2.11

В случае большого числа эмуляторов в разрывы системной магистрали включают входы нечетного числа таких БИС, а их выходы подключают к магистрали через логический элемент Исключающее ИЛИ. В этом случае источником новой информации в системной магистрали будет только одна БИС, выполняющая функции эмулятора накопителя ВЗУ. При четном числе эмуляторов один из нечетных выходов элемента Исключающее ИЛИ может быть соединен с разрывом системной магистрали непосредственно или через сдвиговый регистр в зависимости от способа включения эмулятора.

Для выполнения ВЗУ или микроЭВМ К145ИК13 (К745ИК13) функций эмулятора словарный запас входного языка некоторых ПМК содержит оператор К*Р* (где $P = 3, 4, \ldots, E$). Если в систему включено несколько эмуляторов накопителей ВЗУ, то перед этим оператором набирается его адрес 26, 27, ..., 2С. Адреса 26, 27, 28, 29, 2А вводятся с пульта управления нажатием соответственно клавиш К +, К—, К×, К÷, К↔, а адреса 2В и 2С могут быть занесены в ВЗУ или эмулятор накопителя ВЗУ. При одном эмуляторе его адрес не вводится.

Оператор К*Р* адресуется к выбранному эмулятору накопителя ВЗУ со списком параметров, определяемым указателем *Р*, отображающим адрес заданного сегмента и его длину. ПМК с эмуляторами накопителей ВЗУ имеют клавиши для ввода директив А↑ и ⇅, а также переключатели С — З — СЧ для выбора режима стирания, записи или считывания и переключатель П — Д для обмена информацией с областью программ или данных оперативной памяти ПМК.

В ПМК расширяющегося ряда числа представлены в формате, показанном на рис. 2.3, но регистры данных дополнены двумя тетрадами для хранения адресов или вспомогательной информации. С учетом тетрад адреса содержимое регистра данных разбито на 14 тетрад и в отличие от формата (2.1) представляется в виде

$$<PN> = <А\ (13{:}12)>\ <ЗП(11)>\ <П\ (10{:}9)>\ <ЗМ\ (8)>\ <М\ (7{:}0)>. \qquad (2.3)$$

Страницы программной памяти предназначены для хранения 7 однобайтовых программных слов, и, следовательно, их длина также составляет 14 тетрад, или 56 бит. Емкость страницы, равная емкости регистра данных, является одной из основных единиц измерения количества информации в динамической памяти. Содержимое этой памяти разбито на три блока М1, М2 и М3 емкостью *n* страниц в каждом. Для ПМК минимальной структуры с двумя микроЭВМ К145ИК13 (К745ИК13) в блоке 14 страниц. Следовательно, емкость каждого блока $14 \cdot 14 = 196$ тетрад, или 424 бит, а емкость всей оперативной памяти 42 страницы, или $42 \cdot 14 = 588$, тетрад, или $588 \cdot 4 = 2352$ бит.

Включение каждой дополнительной микроЭВМ К145ИК13 (К745ИК13) в системную магистраль увеличивает число страниц *n* на единицу, но оптимальной является структура, соответствующая $n = 16$; при большем числе *n* адресация однозначными шестнадцатерич-

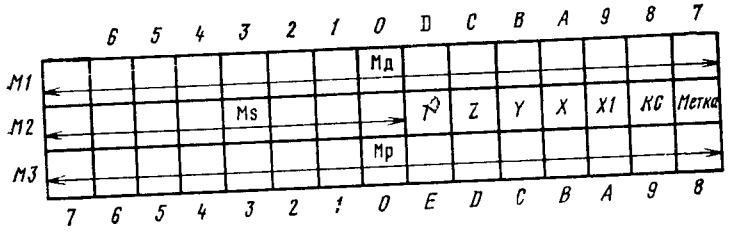


Рис. 2.12

ными цифрами неприменима и следует переходить к более сложной системе адресации страниц памяти, соответственно усложнив системное программное обеспечение. При $n \leqslant 16$ адреса страниц в каждом блоке оперативной памяти (рис. 2.12) определяются шестнадцатеричными цифрами по формуле $n - 7$. Поэтому при $n = 14$ адресация страниц начинается с цифры 7, а при $n = 15$— с цифры 8 (адреса страниц для этих случаев показаны на рис. 2.12 соответственно сверху и снизу символа блока).

Блок M1 оперативной памяти выделен для области данных Мд и содержит n регистров данных с указанными выше шестнадцатеричными адресами от 0 до $n - 1$. В первой странице блока M2 по адресу 7, 8 или 9 в зависимости от числа n находится код «метка», являющийся началом отсчета неявных (пространственно-временны́х) адресов страниц, в следующей странице — канал связи (КС). Следующие 5 страниц этого блока выделены для хранения содержимого регистров операционного стека X1, X, Y, Z и T, а оставшиеся $n - 7$ страниц при соответствующем программном обеспечении операций в системе могут быть заняты памятью программ или данных Ms. Блок M3 выделен для области программ Мр, в которой могут храниться $7n$ шагов (программных слов) прикладной программы емкостью 1 байт (8 бит), или 2 тетрады, каждое.

В сдвиговых регистрах динамической памяти страницы блоков с одинаковыми адресами представлены последовательностями из трех тетрад, выбираемых поочередно по одной тетраде от каждого блока M1, M2 и M3. Каждая из трех тетрад блоков M1, M2 и M3 перемещается в регистрах памяти через фиксированную начальную точку в течение соответственно интервалов времени E1, E2 и E3 разрядных сигналов Д1 ... Д14. После передачи очередной страницы ведущая микроЭВМ пересылает в остальные микроЭВМ короткий синхронизирующий импульс (СИ), совпадающий с фронтом разрядного сигнала Д1. Поэтому после передачи кода «метка» по числу СИ может быть определен адрес требуемой страницы памяти, на которой с помощью сигналов $Дk$EiBj может быть организовано распределение страниц оперативной памяти.

Размещение содержимого страниц оперативной памяти показано на рис. 2.13. В блоке M1 тетрады регистров данных, адресуемые номерами от 0 до 13 согласно (2.3), соответствуют разрядным сигналам Д1... Д14 и интервалам E1 в течение длительности каждого из них. Код «метка» отображается на начальной странице блока M2 цифрами F

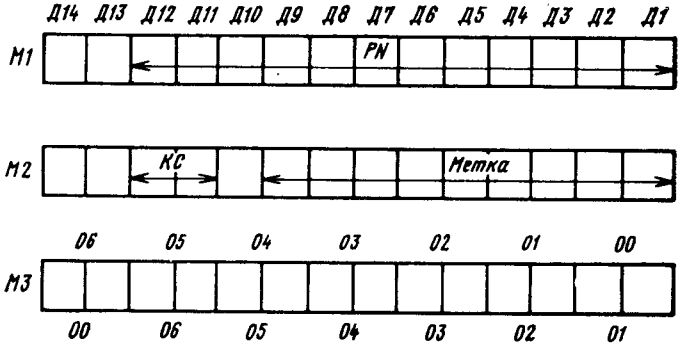


Рис. 2.13

в первых 9 тетрадах (1 в каждой двоичной ячейке этих тетрад). Под канал связи (КС) выделены две тетрады, соответствующие интервалам времени Е2 в течение длительности разрядных сигналов Д11 и Д12. Следует добавить, что считывание кода «метка» уже в первой тетраде приводит к началу отсчета временных координат в микроЭВМ вычислительной системы и формированию СИ в ведущей микроЭВМ.

Адресация 8-битовых ячеек программной памяти зависит от использования страниц блока М2. В ПМК без памяти программ или данных первая тетрада начальной ячейки программных слов на странице блока М3 находится в последней ячейке страницы, а в ПМК с памятью программ-данных первая тетрада последней ячейки программных слов — в начале страницы. Адресация программных слов указана на рис. 2.12 соответственно сверху и снизу символа страницы. Таким образом, от числа n страниц зависит число регистров данных, страниц программной памяти и регистров данных или страниц программ в памяти программ-данных, если она предусмотрена программным обеспечением ПМК.

Малосерийные аналоги массовых ПМК расширяющегося ряда с памятью программ-данных имеют операторы КПВ и КОД (символы которых обозначены на клавиатуре). Они предназначены соответственно для обмена содержимых регистров данных с памятью данных или страницами программных слов с памятью программ.

В общем случае пользователю доступно содержимое памяти данных Мд, программ Мр и памяти программ-данных Ms, если она предусмотрена, а также регистров операционного стека SP и адресного стека SPC с входным регистром-счетчиком шагов программы РС. Кроме того, при использовании ПМК с внешними накопителями пользователю доступна память накопителей Ме и адресное пространство эмуляторов накопителей ВЗУ, обозначаемое Мх.

Зависимость емкости памяти Мр, Мд и Ms от числа страниц n показана на рис. 2.14 диаграммами (слева у символа каждой памяти указаны адреса ее крайних программных слов, а справа — тетрад). Следовательно, при $n = 14$ в памяти программ может храниться до 98 программных слов с адресами от 00 до 97, в памяти данных — содержимое

14 регистров с адресами от 0 до D, а в памяти программ-данных до 49 программных слов (или 7 регистров данных). При $n = 15$ в памяти программ может храниться до 105 программных слов с адресами от 00 до A4, в памяти данных — содержимое 15 регистров с адресами от 0 до E, а в памяти программ-данных — 56 программных слов (до 8 регистров данных). Для оптимальной структуры при $n = 16$ в памяти может храниться до 112 программных слов с адресами от 00 до B1, содержимое 16 регистров данных с адресами от 0 до F до 63 программных слов (до 9 регистров данных).

В ПМК с памятью программ-данных возможен обмен ее содержимого с памятью данных или программ по страницам или регистрам данных с адреса 00. Однако возможна также и непрямая (фиксированная), адресация, обеспечивающая доступ ко всей информации в памяти программ-данных, которая полностью обменивается с содержимым памяти данных или программ.

В ПМК с ВЗУ пользователю доступна память Ме эмулятора накопителя ВЗУ емкостью до 777 программных слов (8-битовых) с адресами от 000 до 776 или до 111 регистров данных с адресами от 00 до 110. Кроме того, пользователю может быть непосредственно доступна память накопителей емкостью 512 программных слов (или 72 регистров данных) или кратной емкости. Для обращения к памяти ВЗУ используются различные способы адресации. В частности, для обращения к эмулятору накопителя ВЗУ оператор КР может адресоваться к одному из 12 программных сегментов или сегментов данных.

Формат и распределение информации в регистрах данных различных устройств памяти определяются (2.2), а в кодах операторов прикладной программы зависят от их назначения. Большинство операторов отображается одним программным словом из двух тетрад. В кодах операторов набора чисел (литералах) и операторов обращения к памяти, а также операторов косвенной адресации старшая тетрада предназначена для кода операции (КОП), младшая тетрада для кода числа, кото-

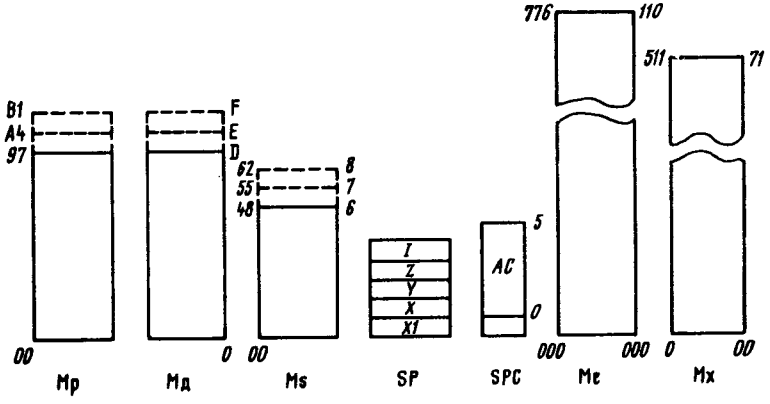


Рис. 2.14

рым может быть и адрес регистра памяти данных (рис. 2.15, *а*). В кодах синтаксических и функциональных операторов, а также кодах управления ходов выполнения программы автоматических вычислений обе тетрады отведены под код операции (рис. 2.15, *б*). Исключение составляют операторы цикла и операторы прямых безусловных (включая оператор обращения к подпрограмме) и условных переходов, отображаемых двумя программными словами (рис. 2.15, *в*): первое — код операции перехода или проверки условия, а второе — двузначный исполнительный адрес ЕА программного слова, выполняемого после перехода. Оператор КР обращения к памяти эмулятора накопителя ВЗУ также отображается в общем случае двумя программными словами (рис. 2.15, *г*): первое — адрес Ме (АМе) эмулятора, а второе — код операции в старшей тетраде и параметр *Р* в младшей.

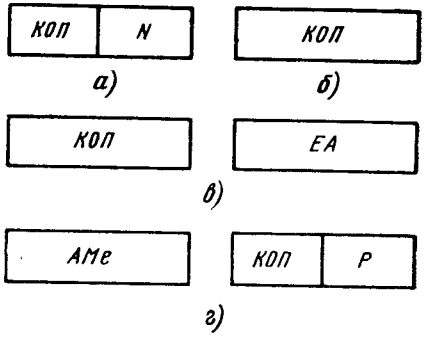


Рис. 2.15

2.4. ПМК СЕМЕЙСТВА «ЭЛЕКТРОНИКА МК-54»

Рассматриваемое семейство включает массовые карманные ПМК «Электроника Б3-34», «Электроника Б3-54», «Электроника МК-54», а также настольный ПМК «Электроника МК-56». Эти ПМК характеризуются минимальной структурой расширяющегося ряда — в системную магистраль включены две микроЭВМ К145ИК13 (К745ИК13) и два регистровых ОЗУ К145ИР2. Каждый блок оперативной памяти этих ПМК содержит $n = 14$ страниц, что соответствует содержимому 14 регистров памяти данных с адресами от 0 до D и 14 страницам памяти программ с 98 программными словами (8-битовыми) с адресами от 00 до 97.

Первый в этом семействе по времени разработки ПМК «Электроника Б3-34» (рис. 2.16) собран на микросхемах серии К145 в корпусе, аналогичном корпусу ПМК «Электроника Б3-21». Он снабжен автономным источником питания в виде батареи из четырех аккумуляторов ДО, 55С, с общим напряжением 4В, подзаряжаемым от съемного блока питания (вилки-выпрямителя) Д2-10, содержащего трансформатор, понижающий сетевое переменное напряжение 220 В, выпрямительный мостик из четырех диодов Д814А и шунтирующий конденсатор К50-16 емкостью 2000 мкФ при напряжении 10 В. Во всех остальных карманных ПМК расширяющегося ряда (включая и выпущенный небольшой серией ПМК «Электроника Б3-54» в корпусе, аналогичном корпусу ПМК «Электроника Б3-34») источником первичного питания служит батарея из последовательно соединенных сухих элементов типа 316, а в стационарных условиях используются съемные блоки питания (вилки-выпря-

мители), понижающие и выпрямляющие сетевое переменное напряжения 220 В.

Особенностью ПМК «Электроника Б3-34» является также двухпозиционный переключатель Р — Г, во всех остальных ПМК семейства он заменен трехпозиционным переключателем Р — ГРД — Г (радианы — грады — градусы). Кроме того, символы на клавиатуре ПМК «Электроника Б3-34» П, ИП, ↑, $\overleftarrow{XY}$, arcsin arccos и arctg на клавиатурах всех остальных ПМК семейства заменены соответственно стандартизованными символами x → П, П → x, В ↑, ↔ $\sin^{-1}$, $\cos^{-1}$ и tg^{-1}.

ПМК «Электроника МК-54» (рис. 2.17) собран на микросхемах серии К745 в более современном корпусе меньших габиритов. В стационарных условиях для питания от сети переменного тока 220 В используется блок типа Д2-10М, содержащий понижающий трансформатор, выпрямительный мостик на двух диодах КД814А и двух диодах

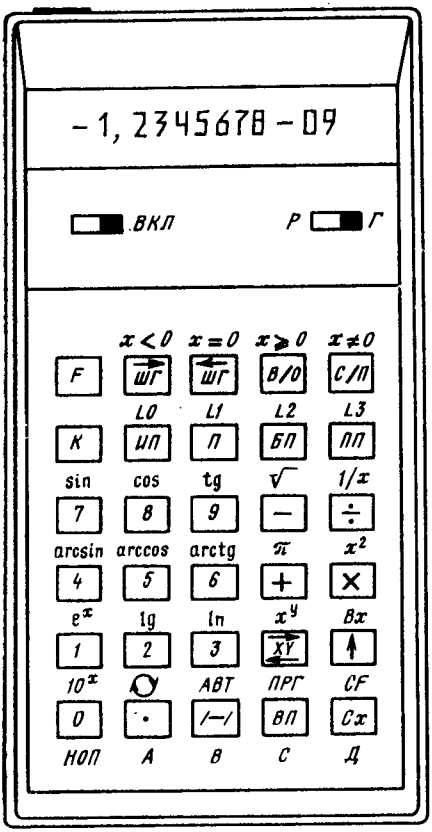


Рис. 2.16

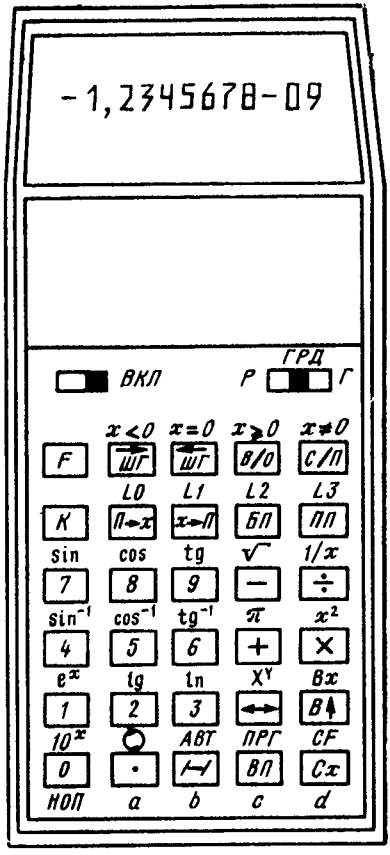


Рис. 2.17

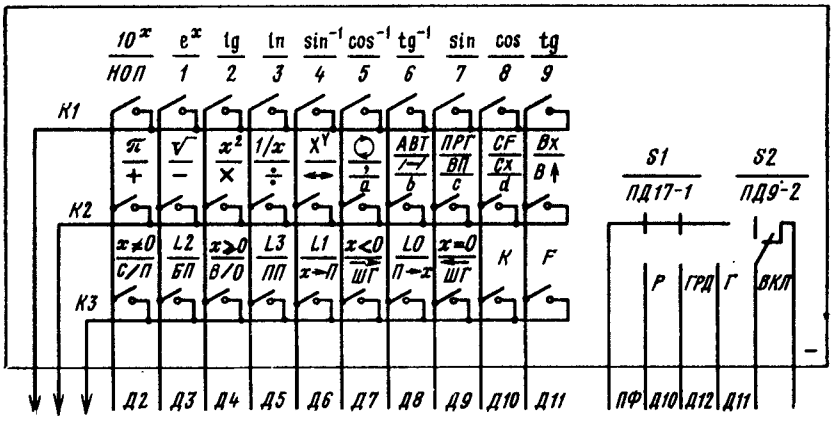


Рис. 2.18

КД105Б, а также шунтирующий напряжение 4 В электролитический конденсатор К50-16 емкостью 2000 мкФ при 10 В.

Настольный ПМК «Электроника МК-56» снабжен клавиатурой и индикаторным устройством больших размеров для удобства пользования в стационарных условиях, питание только от сети переменного тока через встроенный блок.

Кроме указанных выше отличий для ПМК «Электроника Б3-34» схемы пультов управления всех ПМК рассматриваемого семейства совпадают (рис. 2.18). Пульт содержит выключатель питания, включенный в разрыв шины — 4 В источника первичного питания (шина 4 В соединена с корпусом), и переключатель Р—ГРД—Г, на контакты Р, ГРД и Г которого подаются соответственно разрядные сигналы Д10, Д12 и Д11. Клавиатура матричного типа образована 10 входными шинами, на которые подаются разрядные сигналы Д2 ... Д11, и тремя выходными шинами К1, К2 и К3, на которые при нажатии клавиш подаются соответствующие разрядные сигналы. Для ввода операторов, обозначенных над клавишами, предварительно нажимают префиксную клавишу F, а для ввода операторов косвенной адресации и оператора НОП — префиксную клавишу К. Адреса регистров данных набирают нажатием клавиш после нажатия клавиш П → x или x → П, операторы косвенных переходов — после набора символов К и БП, ПП, $x < 0$, $x = 0$, $x \geqslant 0$ или $x \neq 0$. Двузначные адреса программных слов при вводе программ и директивы засылки адреса в счетчик шагов набирают после ввода символов БП, ПП, $x = 0$, $x < 0$, $x \geqslant 0$, $x \neq 0$ и символов LN в итеративных циклах.

Структура ПМК рассматриваемого семейства определяется на уровне неделимых компонентов вычислительного блока, схема которого для ПМК «Электроника МК-54» показана на рис. 2.19.

В системную магистраль через внешние входы и выходы Рг включены сдвиговые регистры ведущей микроЭВМ DD1 (К745ИК1302-2),

микроЭВМ DD2 (К745ИК1303-2) и два регистровых ОЗУ DD3 и DD4 (К745ИР2-2).

От генератора фазовых сигналов DD5 (К745ГФ3-2) на входы всех микроЭВМ и регистровых ОЗУ подаются фазовые сигналы Ф1, Ф2, Ф3 и Ф4 (шины которых для упрощения схемы показаны неявно). Времязадающая цепочка, состоящая из резистора R21 (МЛТ-0,15, сопротив-

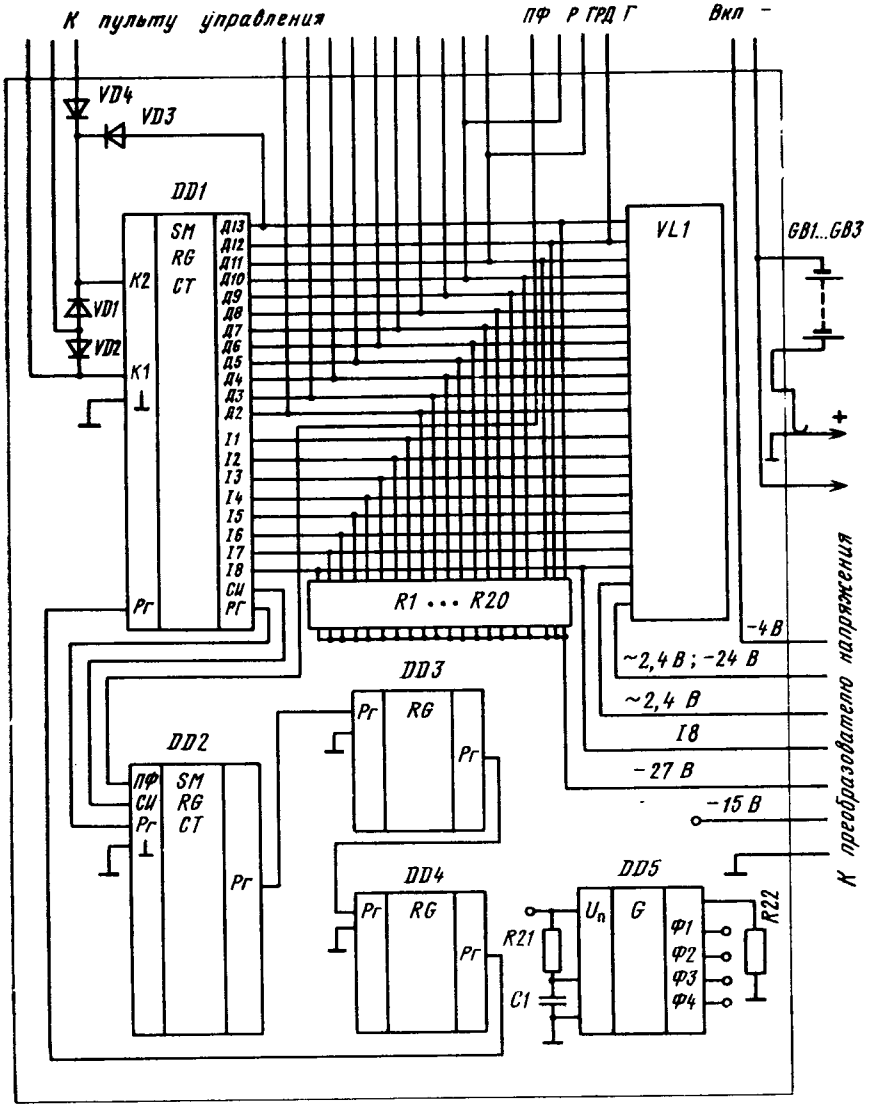


Рис. 2.19

ление 2,3 МОм) и конденсатора С1 (К10-7В, емкость 0,01 мкФ), соединена с выводами генератора, один из которых соединен с корпусом через резистор R22 (МЛТ-0,15, номинальное сопротивление 5,6 кОм, которое может быть изменено при регулировке тактовой частоты). Ко всем БИС от преобразователя напряжения подано также напряжение питания $U_п$ = —15 В (шина которого для упрощения схемы также показана неявно).

Разрядные сигналы Д2, ..., Д13 ведущей микроЭВМ DD1 подаются на сетки знакомест, содержимое которых отображает содержимое регистра индикации в соответствии с форматом (2.2), но с учетом сдвига на один разрядный сигнал, необходимого для пересылки информации в регистр индикации. В зависимости от его содержимого вырабатываются также сегментные сигналы (рис. 2.8, *б*), посылаемые на сегменты выводов индикаторного устройства VL1 типа ИВЛ2-8/12. Одновременно с выходов ведущей микроЭВМ разрядные сигналы Д2, ..., Д11 посылаются на входные шины клавиатуры, а сигналы Д10, .., Д12 — на соответствующие контакты переключателя Р — ГРД — Г.

При вводе директив или операторов прикладной программы разрядные сигналы с выходных шин клавиатуры и разрядный сигнал Д13 с выхода ведущей микроЭВМ через разделительные диоды VD1 ... VD4 (КД522Б) подаются на входы К1 и К2 этой же микроЭВМ, формирующей коды директив или операторов прикладной программы. С выхода ведущей микроЭВМ после пересылки каждой страницы динамической оперативной памяти на вход другой микроЭВМ подается короткий СИ.

На все выходы ведущей микроЭВМ от шины питания — 27 В подается напряжение через блок резисторов R1...R20 (МЛТ-0,15, сопротивление 100 кОм). Так как напряжение — 27 В подано также на нить накала индикаторного устройства, то при отсутствии разрядных и сегментных сигналов на сетке и сегментах знакомест напряжение близко к нулю относительно нити накала. Для подогрева нити накала на ее выводы от преобразователя напряжения подается переменное (пульсирующее) напряжение ~ 2,4 В. При подаче на знакоместо разрядного и сегментного сигналов от ведущей микроЭВМ напряжения на сетке и сегментах знакомест становятся положительными относительно нити накала и вблизи сегмента возникает светящийся газовый разряд.

Источник вторичного питания (преобразователь напряжения) ПМК «Электроника МК-54» преобразует напряжение 4 В в переменное ~ 2,4 В для питания нити накала, постоянное напряжение — 27 В для питания сеток и сегментов знакомест индикаторного устройства, а также постоянное напряжение — 15 В для питания БИС. Основным узлом преобразователя напряжения является блокинг-генератор на транзисторе VT3(КТ814Б) и четырехобмоточном трансформаторе Т1 (рис. 2.20). С обмотки III трансформатора снимается переменное напряжение 2,4 В для питания нити накала индикаторного устройства. Эта обмотка через резистор R3 (МЛТ-0,15, сопротивление 8,2 кОм), шунтированный конденсатором С1 (К10-7В, емкость 0,022 мкФ), соединена с шиной постоянного напряжения — 27 В. Напряжение с по-

вышающей обмотки IV трансформатора, выпрямленное диодом VD2 (КД522Б) и шунтированное конденсатором С3 (К50-16, емкость 10 мкФ), подается на шину напряжения питания — 15 В вычислительного блока. Дополнительное напряжение, снимаемое с обмотки II, выпрямленное диодом VD1 (КД522Д) и шунтированное конденсатором С4 (К10-7В, емкость 0,033 мкФ), суммируется с постоянным напряжением питания БИС и подается на шину — 27 В.

Для стабилизации работы блокинг-генератора и уменьшения пульсаций постоянных напряжений на выходе преобразователя напряжения он содержит цепь глубокой отрицательной обратной связи на транзисторах VT1 (КТ361Г), VT2 и VT6 (КТ315Г). На транзисторе VT4 собран операционный усилитель с делителем напряжения в цепи базы на резисторах R4 и R5 (МЛТ-9,15, сопротивление соответственно 100 кОм и 80 ... 100 кОм, точное значение подбирается при регулировке напряжения питания). Эмиттер этого транзистора соединен с шиной питания — 15 В через стабилитрон VD3 (КС175Ж), а в цепь коллектора включен резистор R6 (МЛТ-0,15, сопротивление 5,6 кОм), шунтированный конденсатором С5 (К10-7В, емкость 0,01 мкФ). С этой нагрузки выходное напряжение операционного усилителя через резистор R7(МЛТ-0,15, сопротивление 1 кОм) подается на базу транзистора VT2.

Операционный усилитель на транзисторах VT1 и VT2 собран по каскодной схеме при напряжении питания на входе I8 индикаторного устройства, отличающемся от напряжения на шине — 27 В падением напряжения около 3 В на блоке резисторов R1 ... R20. Нагрузкой этого усилителя является входное сопротивление участка база-эмиттер транзистора VT3, ток базы которого равен постоянной составляющей тока коллектора транзистора VT2, шунтированного по перемен-

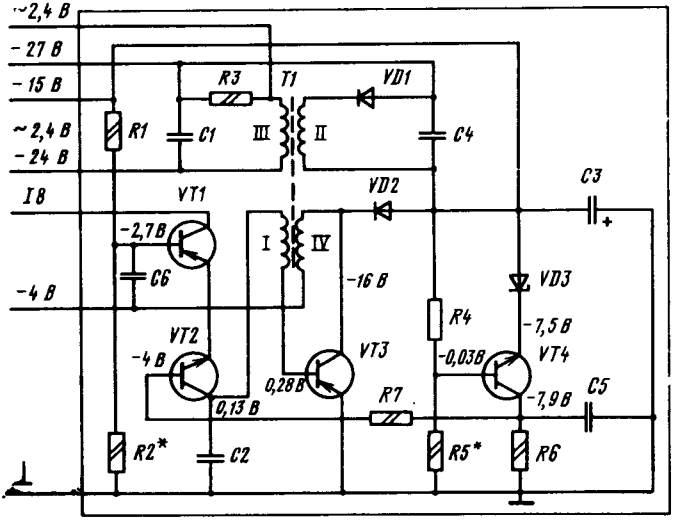


Рис. 2.20

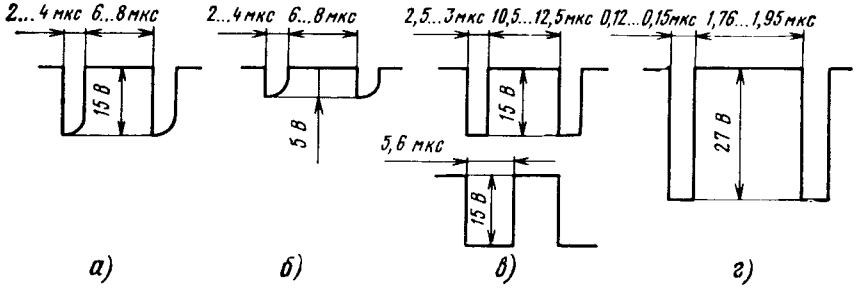


Рис. 2.21

ному току конденсатором С2 (К10-7В, емкость 0,022 мкФ). Напряжение питания на базу транзистора VT1 подается от делителя напряжения — 27 В на резисторах R1 и R2 (МЛТ-0,15, сопротивление соответственно 51 кОм и 11 ... 18 кОм, точное значение подбирается при регулировке). Кроме того, на базу этого транзистора через конденсатор С6 (К10-78, емкость 1500 пФ) подается напряжение 4 В от источника первичного питания.

Таким образом, при переменном напряжении источника первичного питания, изменениях напряжений — 15 В и —27 В ток базы транзистора VT3 изменяется в противофазе относительно этих изменений на базах транзисторов VT1 и VT2, что приводит к изменению режима работы блокинг-генератора, уменьшающему влияние всех дестабилизирующих факторов.

Форма импульсов блокинг-генератора, измеренных между корпусом и коллектором транзистора VT3, отображается осциллограммой на рис. 2.21, *а*. Длительность их от 2 до 4 мс при периоде повторения от 8 до 12 мс. С обмотки III трансформатора Т1 на нить накала индикатора подаются соответственно меньшие импульсы напряжения с той же длительностью и периодом повторения (рис. 2.21, *б*). На этом же рисунке показана форма импульсов фазовых сигналов Ф1, Ф2 (верхний график на рис. 2.21, *в*) и Ф3, Ф4 (нижний график на рис. 2.21, *в*), а также форма разрядных сигналов (рис. 2.21, *г*), измеренных соответственно на фазовых и выходных шинах ведущей микроЭВМ относительно корпуса.

2.5. ПМК СЕМЕЙСТВА «ЭЛЕКТРОНИКА МК-52»

Рассматриваемое семейство включает массовые карманные ПМК «Электроника МК-61» и «Электроника МК-52», а также их малосерийные аналоги с памятью программ-данных или операторами обращения к эмуляторам накопителей ВЗУ, предназначенные, в частности, для обработки медицинской (кардиологической) информации.

В системную магистраль этих ПМК включены три микроЭВМ К745ИК13 и два регистровых ОЗУ К745ИР2, что позволило увеличить число страниц в блоках оперативной памяти по сравнению с ПМК

семейства «Электроника МК-54» до $n = 15$ и обеспечивать возможность доступа пользователю к 15 регистрам данных с адресами от 0 до E и 15 страницам программной памяти для хранения до 105 программных слов с адресами от 00 до A4.

Прямая адресация программных слов с адресами A*b* от A0 до A4 обеспечивается при наборе адреса перехода после ввода операторов безусловного перехода или проверки условия нажатием клавиши, под которой обозначена шестнадцатеричная цифра A (на клавише обозначен десятичный разделительный знак в виде точки), и следующей клавиши для набора цифры $b = 0,1, 2,3$ или 4. При косвенных переходах в программах приходится предварительно формировать адреса A*b* в регистрах памяти, используемых в операторах косвенной адресации. Такие адреса проще всего формировать, последовательно нажимая клавиши

1 2 *b* В↑ 1 8 0 К∨ К|x| ВП 2 КНОП П*N*

что приводит к высвечиванию на индикаторе (в виде — *b*) адреса A*b* и засылке его в регистр *N* памяти. Для проверки достаточно нажать клавиши БП *N* F ПРГ, и на индикаторе в знакоместах порядка, выделяемых в режиме программирования для хранения содержимого счетчика программных слов (РС), высвечивается адрес A*b*. Следует учитывать, что при $N < 7$ адрес A*b* модифицируется при каждом выполнении оператора косвенных переходов.

При необходимости формирования адреса A*b* в процессе выполнения прикладной программы в автоматическом режиме следует ввести в нее фрагмент

ИП7 ИП8 К∨ К|x| В↑ ВП 2 П*N*

предварительно введя исходные данные $12b$ = Р7, 180 = Р8. Если использование регистров памяти для хранения исходных данных нежелательно, то можно использовать более длинный фрагмент

1 2 *b* В↑ 1 8 0 *К*∨ К|x| В↑ ВП 2 П*N*

отличающийся от приведенного для ввода нажатием клавиш оператором В ↑, без которого в автоматическом режиме искомый результат не будет получен.

Возможны и другие способы формирования адресов переходов A*b* (рассмотренные в гл. 4).

ПМК рассматриваемого семейства с памятью программ-данных отличаются обозначенными на клавиатуре символами К ПВ и К ОД для операторов обмена соответственно страницами программной памяти или регистрами памяти данных.

Массовый ПМК «Электроника МК-61» (рис. 2.22) конструктивно оформлен в корпусе, аналогичном корпусу ПМК «Электроника МК-54», но внешне отличается от него бо́льшим числом символов на клавиатуре. Символы, обозначенные слева над клавишами, вводят после нажатия префиксной клавиши F, а символы, обозначенные справа над клавиша-

ми, и операторы косвенной адресации — после нажатия клавиши К.

Электрическая схема клавиатуры этого ПМК подобна схеме клавиатуры ПМК «Электроника МК-52», но на последней обозначены дополнительные клавиши А↑ и ⇅ (рис. 2.23) для ввода директив, обеспечивающих обмен информацией с внешними устройствами. Контакты этих клавиш присоединены к входным шинам разрядных сигналов Д3 и Д4.

Пульт управления ПМК «Электроника МК-52» (рис.2.24) кроме клавиатуры и переключателей для включения питания и выбора размерности аргумента тригонометрических и значений обратных тригонометрических функций содержит дополнительные переключатели: С — З — СЧ (стирание — запись — считывание) для выбора режима обмена информацией с внешними устройствами и П — Д для обмена с накопителем содержимым программной памяти или памяти данных ПМК.

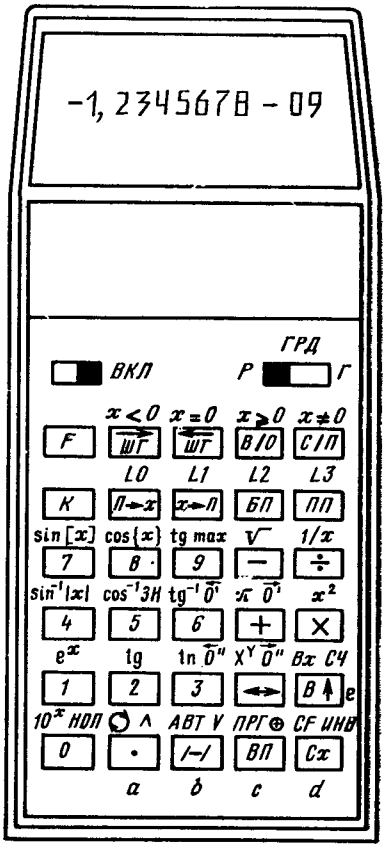


Рис. 2.22

ПМК «Электроника МК-52» имеет два разъема для присоединения внешних устройств, а также встроенный накопитель информации в виде репрограммируемого (перепрограммируемого) ЗУ (РПЗУ) КР1601РР7 со стиранием и записью информации емкостью 1024 тетрады разрядов, что обеспечивает возможность длительного хранения (с допустимым числом циклов перезаписи до 10 000) до 512 программных 8-битовых слов или содержимого 72 регистров данных.

Электрическая схема вычислительного блока ПМК «Электроника МК-61» и его малосерийных аналогов практически совпадает с частью схемы ПМК «Электроника МК-52», показанной на рис. 2.25. В этой части вычислительного блока ведущая микроЭВМ К745ИК1302-2 соединена с системной магистралью (через вход и выход Рг), в которую включены также микроЭВМ DD2 (К745ИК1303-2) и DD6 (К745ИК1306-2), а также регистровые ОЗУ DD3 и DD4 (К745ИР2-2). В разрыв системной магистрали ПМК между микроЭВМ DD2 и DD 6 включены устройства ввода-вывода блока обмена информацией с внешними устройствами (см. рис. 2.26).

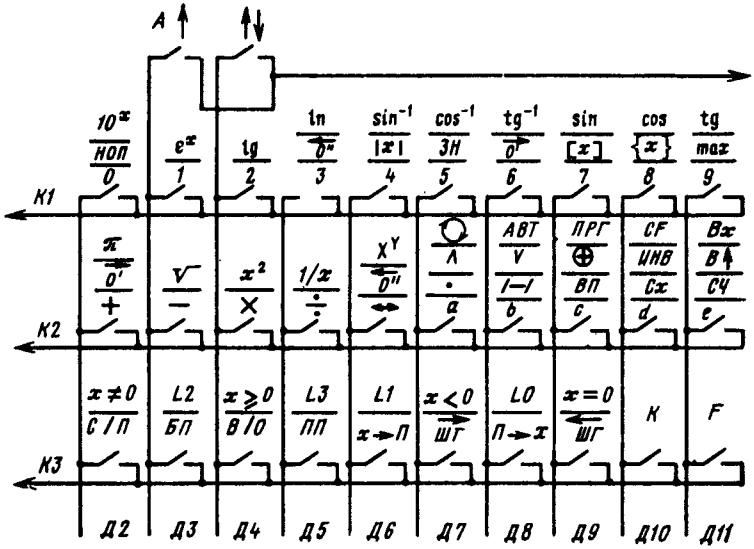


Рис. 2.23

От генератора фазовых сигналов DD5 (К745ГФ3-2) с времязадающими цепочками иа конденсаторе С1 (К10-715, емкость 0,01 мкФ) и резисторах R1 и R2 (МЛТ0,125, сопротивления соответственно 1 и 5,6 кОм) на все микроЭВМ и разъемы внешних устройств подаются фазовые сигналы Ф1, ..., Ф4 (шины фазовых сигналов для упрощения схемы показаиы неявно). На все микросхемы от преобразователя напряжения подается напряжение питания $U_п$ = — 15 В (шина которого на схеме также показаиа неявно).

С выхода ведущей микроЭВМ DD1 разрядные сигналы Д2, ..., Д11 подаются на входные шины клавиатуры, а сигналы Д2, ..., Д13— на соответствующие входы индикаторного устройства VL1 ИВЛ2-8.На выводы сегментов индикаторного устройства подаются соответствующие сегментные сигналы I1, ..., 18. Все выходы сегмеитных и разрядных сигналов через блоки резисторов R1 ... R20 сопротивлеиием 100 кОм каждый соединены с шиной — 27 В преобразователя напряжения. При этом в ПМК «Электроника МК-52» это напряжение подается на выход сегмеитиого сигиала I4 через разделительный диод VD5 КД522Б (отсутствующий в схеме ПМК «Электроника МК-61»), от которого напряжение отводится к устройству, показанному на следующей схеме. Разрядные сигналы Д10, Д12 и Д11 подаются также на коитакты Р, Г и ГРД соответственно переключателя Р — ГРД — Г, а разрядный сигнал Д13 через разделительиые диоды VD1 ... VD4 (КД522Б) вместе с сигналами выходных шин клавиатуры — на входы К1 и К2 ведущей микроЭВМ DD1.

После перемещения в сдвиговом регистре оперативной памяти микроЭВМ DD1 каждой страницы содержимого памяти вырабатывается и посылается на входы микроЭВМ DD2 и DD6 короткий (длительностью около 10 мкс) синхроимизирующий сигнал (СИ), после которого во времеиных интервалах Д*k*Е*i*В*j* может быть считано или записано содержимое любого участка оперативной памяти с точностью до одиого бита. Кроме того, с переключателя Р—ГРД—Г в зависимости от его положения разрядный сигнал Д10, Д11 или Д12 подается на вход ПФ микроЭВМ DD2, содержащей в ПЗУ программное обеспечение для вычисления тригонометрических и обратных тригонометрических функций. Кстати, микроЭВМ DD6 содержит программное обеспечение выполнения всех тех операторов входного языка, которыми он отличается от входиого языка ПМК семейства «Электроника МК-54».

Управляющие и разрядные сигналы представляют собой импульсы напряжения около 15 В. При этом длительность фазовых сигналов Ф1 и Ф3 2,6 ...3 мкс, а Ф2 и Ф4 около 5,6 мкс с периодом повторения каждого из них 13,1 ... 15,5 мкс, равным времени такта и, соответственно, длительности сигналов В*j*. Длительность разрядных сигналов 0,12... 0,15 мс, период повторения 1,88 ... 2,25 мс в зависимости от тактовой частоты ПМК, определяемой генератором фазовых сигналов.

Таким образом, вычислительный блок ПМК «Электроника МК-61» и соответствующая часть вычислительного блока ПМК «Электроника МК-52» и его малосерийных аналогов отличаются от вычислительного блока

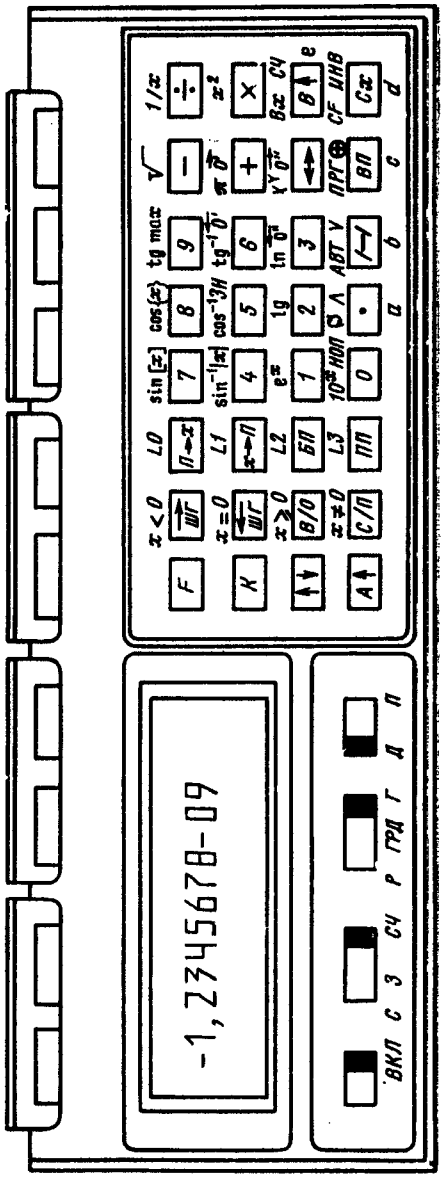


Рис. 2.24

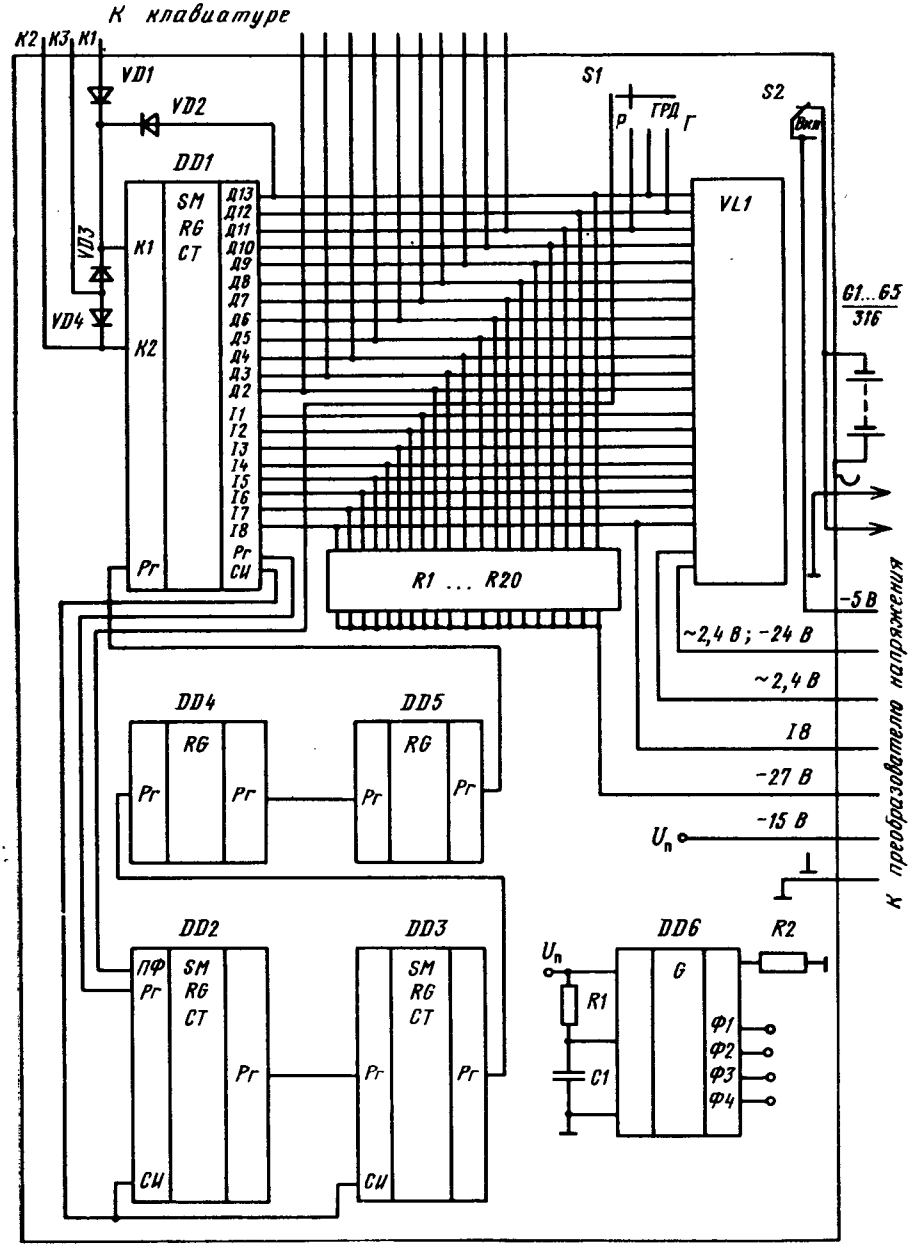

К клавиатуре
К2
К3
К1
VD1
VD2
VD3
VD4
DD1
SM
RG
СТ
К1
К2
Рг
СИ
Д13
Д12
Д11
Д10
Д9
Д8
Д7
Д6
Д5
Д4
Д3
Д2
I1
I2
I3
I4
I5
I6
I7
I8
S1
Р
ГРД
Г
S2
Вкл
VL1
G1...G5
316
R1 ... R20
-5 В
~2,4 В; -24 В
~2,4 В
I8
-27 В
-15 В
Uп
К преобразователю напряжения
DD4
RG
Рг
DD5
DD2
ПФ
SM
RG
СТ
DD3
DD6
G
R1
C1
R2
Ф1
Ф2
Ф3
Ф4

Рис. 2.25

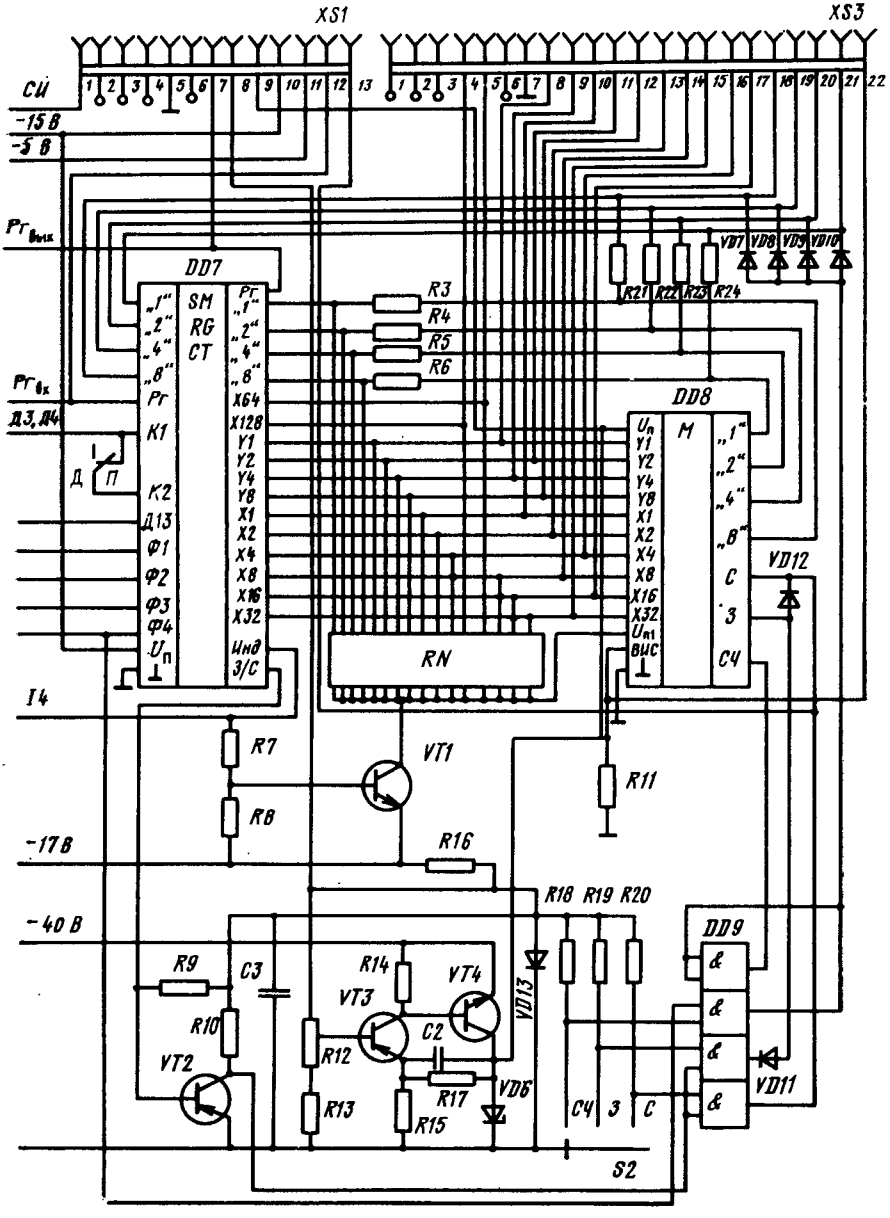

XS1
XS3
СИ
-15 В
-5 В
Рг вых
Рг вх
Д3, Д4
Д
П
DD7
SM
RG
CT
Рг
K1
K2
Д13
Ф1
Ф2
Ф3
Ф4
Uп
Инд З/С
X64
X128
Y1
Y2
Y4
Y8
X1
X2
X4
X8
X16
X32
R3
R4
R5
R6
RN
DD8
M
C
З
СЧ
ВЫС
VD7
VD8
VD9
VD10
R21
R22
R23
R24
VD12
I4
R7
R8
VT1
R11
-17 В
R16
R18
R19
R20
-40 В
DD9
R9
C3
R10
VT2
R14
VT4
VT3
C2
R12
R13
R17
R15
VD6
VD13
VD11
S2

Рис. 2.26

ПМК семейства «Электроника МК-54» в основным лишь включением в системную магистраль дополнительной микроЭВМ типа К745ИК1306 с соответствующим программным обеспечением.

Дополнительная часть вычислительного блока ПМК «Электроника МК-52» (рис. 2.26) содержит микроЭВМ DD7 (ИК7451801-2), выполняющую функции контроллера внешних устройств, встроенный внешний накопитель в виде РПЗУ DD8 (КР1601РР7), микросхему DD9 (К561ЛА7) с четырьмя логическими элементами И—НЕ. Для подсоединения ВЗУ (ОЗУ для хранения прикладных программ, блоков расширения оперативной памяти, накопителей информации), обмен информации с которыми контролируется микроконтроллером DD7, предназначен разъем XS3 типа ОНп-КТ-22-22. Разъем XS1 типа ОНп-КТ-22-15 предназначен для подсоединиения ВЗУ с собственными контроллерами, включаемыми параллельно микроЭВМ DD7 в разрыв системной магистрали.

МикроЭВМ DD7 имеет входы для фазовых сигналов и питания —15 В (на схеме шины показаны неявно), вход Рг для приема информации с системной магистрали (между DD3 и DD2), вход для разрядного сигнала Д13 с выхода ведущей микроЭВМ DD1 и входы К1 и К2, на один из которых или оба в зависимости от положения переключателя П — Д подаются сигналы Д3 и Д4 соответственно при вводе директив А ↑ и ↑↓. Кроме того, эта микроЭВМ имеет параллельную 4-разрядную шину данных с входами «1», «2», «4» и «8», через которые с встроенного РПЗУ или ВЗУ поступает информация, пересылаемая последовательно в системную магистраль с выхода Рг.

При обмене информацией между системной магистралью и ВЗУ с выхода Инд микроЭВМ DD7 сигнал I4 (через разделительный диод VD5 (КД522Б) на рис.2.24) подается на индикаторное устройство для высвечивания сигнала обмена в виде знака «минус» во всех знакоместах и через делитель напряжения на резисторах R7 (18 кОм) и R8 (68 кОм) типа МЛТ0,585 на базу транзистора VT1 (КТ315Г). Транзистор открывается, и через него напряжение с шины — 17 В подается на вход $U_{п1}$ накопителя DD8, а через блок резисторов R*N* сопротивлением 15 кОм каждый — на все выходные шины адресов и данных микроЭВМ DD7.

С выхода З/С (Запись/Считывание) напряжение подается на базу транзистора VT2 (КТ361Г) с резисторами R9 (100 кОм) и R10 (18 кОм) типа МЛТ0,15 в цепях питания. Выходной сигнал с коллектора VT2 подается на входы микросхемы DD9, соответствующие режиму считывания и записи. Питание на усилитель на транзисторе VT2 подается с шины — 12 В, на которую напряжение поступает с шины преобразователя напряжения — 17 В через резистор R16 (МЛТ0, 125, сопротивление 5,6 кОм). С шины — 12 В, зашунтированной конденсатором С3 (0,01 мкФ) и стабилитроном VD13 (КС212К), напряжение через резисторы R18, R19 и R20 (МЛТ0,125, сопротивление 100 кОм) поступает на контакты переключателя Р — ГРД — Г и соответствующие входы микросхемы DD9.

При установке переключателя Р—ГРД—Г в требуемое положение соответствующий вход DD9 замыкается, результат логического умножения становится равным нулю, инвертируется в сигнал логической 1. С выходов микросхемы DD9 соответствующие сигналы непосредственно или через разделительные диоды VD11 и VD12 подаются на входы З и С накопителя DD8. При этом сигнал считывания подается на диоды VD7 ... VD10 (КД522Б), а на вход СЧ РПЗУ поступает инвертированный в микросхеме DD9 сигнал.

Сигнал записи с выхода DD9 подается на разъем XS1 и делитель напряжения на резисторах R12 (СП-38, полупеременное сопротивление 33 кОм) и R13 (МЛТ0, 125; 18 кОм). С выхода цепи, образованной транзисторами VT3 (КТ361Б) и VT4 (КТ315В), резисторами R14 (18 кОм), R15 (5,6 кОм), R17 (24 кОм) типа МЛТ0,125, конденсатором С2 (0,01 мкФ), диодом VD6 (КД552Б), и питающегося от шины — 40 В преобразователя напряжения сигнал подается на выводы $U_{п}$ накопителя DD8 и разъем XS1.

Обмен данными между БИС DD7 и DD8 обеспечивается устройством ввода-вывода данных РПЗУ с 4-разрядной параллельной шиной (выводы «1», «2», «4», «8»), соединенной с разъемом XS3 непосредственно, с параллельной шиной данных DD7 через резисторы R3 ... R6 (МЛТ0,125, сопротивление 15 кОм) и с шиной ввода данных через резисторы R21 ... R24 (МЛТ0,125, сопротивление 100 кОм). К выводам разъема XS1 подведены (считая слева направо) сигналы СИ, Ф3, Ф4, Ф2, Рг DD7, сигнал записи $U_{з}$ с микросхемы DD9, сигнал с цепи коллектора транзистора VT4, напряжение $U_{п}$ = — 15 В от преобразователя напряжения и сигнал стирания с микросхемы DD9. С разъемом XS3 соединены (слева направо) сигналы Ф2, Ф4, Ф3, адресные сигналы X64, X128, напряжение питания — 15 В, корпус, адресные сигналы Y1, Y4, X1, Y8, X32, X16, сигналы входной параллельной шины «1», «2», «4», «8» микроконтроллера и сигнал БИС от ВЗУ, подаваемый на вход DD8 и шунтируемый резистором R11 (МЛТ0,125, сопротивление 18 кОм).

Особенности обмена информацией с внешними устройствами через разъемы зависят от их назначения и описаны в руководствах по их применению. При обмене информацией с внешним устройством, присоединяемым к разъему XS3, встроенный накопитель информации выполняет функции буферного устройства.

Область памяти встроенного РПЗУ DD7 состоит из 1024 4-разрядных ячеек с адресами от 0000 до 1023, разбитых на 24 строки по 16 ячеек в каждой. Обмен информацией с оперативной памятью ПМК реализуется по страницам 8-битовых программных слов, причем в каждой странице памяти РПЗУ начальная ее ячейка (ячейка с начальным адресом) расположена в конце страницы (табл. 2.1). Поэтому в адресе вида ЦЯЯЯДД для обращения к участку памяти с адресом ЯЯЯЯ начальной ячейки длина участка, задаваемая десятичным числом ДД программных слов, должна быть кратной 7 (если в адресе указано другое число, то обращение выполняется к большему числу ячеек, крат-

Таблица 2.1

Адресное поле РПЗУ типа КР1601РР7

Номер строки	Адреса программных слов на страницах и тетрад															
1	01		02		03		04		05		06		00		08	
	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	0010	0011	0012	0013	0014	0015
2	09		10		11		12		13		07		15		16	
	0016	0017	0018	0019	0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	0030	0031
3	17		18		19		20		14		22		23		24	
	0032	0033	0034	0035	0036	0037	0038	0039	0040	0041	0042	0043	0044	0045	0046	0047
...	...															
63	490		498		499		500		501		502		503		497	
	0992	0993	0994	0995	0996	0997	0998	0999	1000	1001	1002	1003	1004	1005	1006	1007
64	505		506		507		508		509		510		504			
	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023

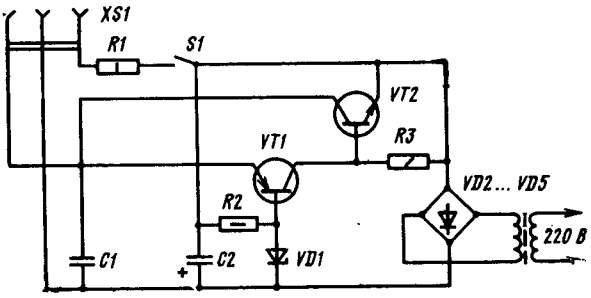


Рис. 2.27

ному 7). Обмен содержимым РПЗУ и программной памяти начинается с ее начального адреса 00, а обмен регистрами данных — с адреса 0 регистров данных памяти ПМК. Максимальное двухразрядное десятичное число ДД, кратное 7, равно 98, и, следовательно, одно обращение к РПЗУ обеспечивает запись в него неполного содержимого программной памяти ПМК или 14 регистров данных с адресами от 0 до D.

Максимальную длину участка памяти РПЗУ при одном обращении можно расширить на 4 программных слова (8 тетрад), сформировав с помощью логических операторов адрес с числом ДД = А2 (например, выполнив 1ЯЯЯЯ82V2ЯЯЯЯ20 = 8, ЯЯЯЯА2). В этом случае можно переписать за одно обращение к РПЗУ до 102 шагов программной памяти или содержимое первых 14 регистров данных с адресами от 0 до D и целую мантиссу без знака содержимого регистра Е (отрицательные знаки и порядок сохраняются после обмена в регистре Е памяти данных ПМК).

ПМК «Электроника МК-61» и его малосерийные аналоги снабжены источником автономного первичного питания 4 В в виде батареи из трех сухих элементов типа 316 и для питания от сети переменного напряжения 220 В, 50 Гц, съемным блоком питания (вилкой-выпрямителем) типа Д2-10М. Источником автономного питания ПМК «Электроника МК-52» напряжением 5 В является батарея из четырех сухих элементов типа 316, а для питания от сети используется блок типа БП2-3К (рис. 2.27). Этот блок кроме трансформатора, понижающего переменное сетевое напряжение 220 В, и выпрямительного мостика из 4 силовых диодов VD2 ... VD5 (КД105) содержит стабилизатор выходного напряжения — 5 В на транзисторе VT2 (КТ961) с цепью отрицательной обратной связи на транзисторе VT1 (КТ381Б) и стабилитроне VD1 (КС156А). В цепи питания транзисторов используются резисторы R2 типа МЛТ0,5 (470 кОм), R3 типа МЛТ0,125 (2,4 кОм), а также конденсатор С2 типа АМ-5В емкостью 500 мкФ. Выходное напряжение стабилизатора шунтируется конденсатором С1 типа К50-16 (0,1 мкФ). Цепь подзарядки аккумулятора с резистором R1 типа МЛТ1 (200 Ом) и выключателем S1 типа ПД9-2 этого блока при питании ПМК «Электроника МК-52» не используется.

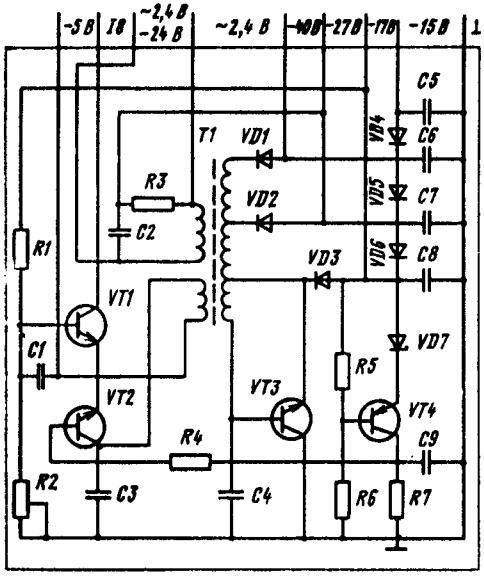


Рис. 2.28

Преобразователи напряжения ПМК «Электроника МК-61» и «Электроника МК-54» практически совпадают. Схема преобразователя напряжения ПМК «Электроника МК-52» показана на рис.2.28. Основным узлом преобразователя является блокинг-генератор на трехобмоточном трансформаторе Т1 и транзисторе VT3 (КТ814Б), вывод базы которого зашунтирован на корпус конденсатором С4 (К10-7В, емкость 4700 пФ). Одна из вторичных обмоток трансформатора Т1 предназначена для питания нити накала индикаторного устройства и соединена с шиной — 27 В через резистор R3 (МЛТ0, 125, сопротивление 7,5 кОм), зашунтированный конденсатором С3 (К10-7В, емкость 0,033 мкФ).

С отводов повышающей обмотки трансформатора через диоды VD1, VD2 и VD3 (КД522Б), шунтируемые соответственно конденсаторами С6, С7, С8 (К50-16, емкость 5; 5 и 10 мкФ), выпрямленные напряжения подаются на шины —17, —27 и —40 В. С шины — 17 В через последовательно соединенные диоды VD4, VD5 и VD6 (КД522Б), шунтируемые конденсатором С5 (К50-16, емкость 2 мкФ), напряжение подается на шину питания — 15 В.

К шине —17 В подключен усилитель на транзисторе VT4 (КТ315Г) со стабилитроном VD7 (КС175К) в цепи эмиттера. Напряжение на базу этого транзистора снимается с делителя на резисторах R5 (МЛТ0, 125, сопротивление 68 кОм) и R6 (СП3-38, номинальное полупеременное сопротивление 100 кОм). Нагрузкой в цепи коллектора является R7 (МЛТ0,125, сопротивление 5,6 кОм), шунтируемый конденсатором С9 (К10-7В, емкость 0,015 мкФ). Постоянная составляющая падения напряжения на резисторе R7 через резистор R4 (МЛТ0,125, сопротивление 1 кОм) подается на базу транзистора VT2 (КТ315Г). Транзисторы VT1 (КТ361Г) и VT2 соединены по каскодной схеме с питанием от напряжения на выводе для сегментного сигнала 18 индикаторного устройства. Напряжение питания базы транзистора VT1 снимается с делителя напряжения на резисторах R1 ((МЛТ0,125, сопротивление 51 кОм) и R2 (СП3-38), сопротивление которого устанавливается при регулировке (номинальное сопротивление 33 кОм). Кроме того, вывод базы этого транзистора соединен через

конденсатор С1 (К10-7В, емкость 1500 пФ) с шиной первичного питания — 5 В.

Ток коллектора транзистора VT2, переменная составляющая которого замыкается на корпус через конденсатор С2 (К10-7В, емкость 0,015 мкФ), через первичную обмотку трансформатора Т1 подается на базу транзистора VT3. Так как изменения напряжения — 17 В на транзисторе VT4, — 27 В на транзисторе VT1 и напряжения питания на базе VT1 приводят к противофазным изменениям тока базы транзистора VT3, то цепь обратной связи стабилизирует работу блокинг-генератора и выходные напряжения преобразователя.

Глава 3

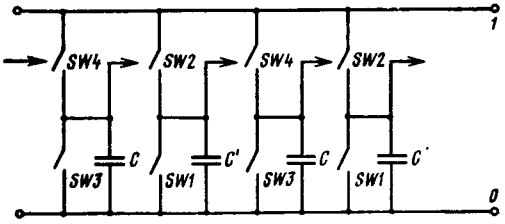
ЭЛЕМЕНТНАЯ БАЗА ПРОГРАММИРУЕМЫХ МИКРОКАЛЬКУЛЯТОРОВ ПОСЛЕДОВАТЕЛЬНОГО ДЕЙСТВИЯ

3.1. ОСНОВНЫЕ ХАРАКТЕРИСТИКИ МИКРОСХЕМ

С системной магистралью вычислительной системы ПМК в основном связаны микросхемы серий К145 и К745, изготовленные по МОП-технологии. Микросхемы серии К145 конструктивно оформлены в стандартных пластмассовых 48-выводных корпусах, а микросхемы серии К745 бескорпусные. В системную магистраль ПМК включены динамические сдвиговые регистры однокристальных микроЭВМ К145ИК5 или К145ИК13 (К745ИК13) и регистровых ОЗУ К145ИР1 или К145ИР2 (К745ИР2).

Пересылка информации в сдвиговых регистрах микроЭВМ и регистровых ОЗУ на каждом такте разбивается на несколько состояний (фаз), соответствующих фазовым сигналам, поступающим от генератора К167ГФ2 или К145ГФ3 (К745ГФ3). Эти микросхемы содержат мультивибраторы на логических элементах, формирующих при включении ПМК последовательности фазовых сигналов Ф1, Ф2, Ф3 и Ф4 с периодом повторения, равным длительности В такта. Тактовая частота, определяемая параметрами внешних (навесных) *RC*-цепей, 75...100 кГц, что соответствует длительности такта 10 ... 13,3 мкс.

Работу сдвиговых регистров микроЭВМ и регистровых ОЗУ удобно описывать эквивалентной схемой двоичной ячейки, содержащей основную С и буферную С′ емкости, а также ключи SW1...SW4, замыкающиеся под воздействием соответствующих фазовых сигналов Ф1,...,Ф4 (рис. 3.1). Эти емкости небольшие и поэтому успевают заметно зарядиться в течение такта малой длительности, а при большом числе тактов напряжение, соответствующее уровню логической 1, поддерживается напряжением питания БИС. При поступлении фазового сигнала Ф1 ключи SW1 замыкаются, и на буферных емкостях С′ устанавливается напряжение, соответствующее уровню логического 0, — фаза очистки буфера. В течение следующего фазового сигнала Ф2 замыкаются ключи SW2, и на буферных емкостях С′ устанавливаются напряжения, соответствующие уровню логического 0 или 1 на основных емкостях С, — фаза выборки. При поступлении



Рнс. 3.1

следующего сигнала Ф3 замыкаются ключи SW3 и на основных емкостях С устанавливается напряжение уровня логического 0 — фаза очистки ячеек. Наконец, в течение последнего на такте фазового сигнала Ф4 замыкаются ключи SW4 и на основных емкостях С устанавливаются напряжения уровня логического 0 или 1, соответствующие содержимому предыдущей ячейки в начале такта, — фаза пересылки. Подобные состояния, повторяясь на каждом такте, обеспечивают перемещение информации.

Для считывания содержимого регистра в одноразрядную шину достаточно подсоединить ее к емкости С одной из ячеек через дополнительный ключ SW2 или SW4, тогда при каждом поступлении фазового сигнала Ф2 или Ф4 последовательный поток информации через регистр будет ответвляться в дополнительную шину (например, соединенную с входом другого регистра). Если ее соединить через дополнительныюй ключ SW4, то поступающий от шины последовательный поток информации будет суммироваться с потоком информации, проходящим через ячейку предыдущей части регистра, в последующую часть которого будет записываться произведение обоих потоков информации. Если же на время записи заблокировать замыкание собственного ключа SW4 ячейки, то в регистр будет записываться только информация, поступающая с одноразрядной шины.

Выполнив несколько отводов от С смежных ячеек, можно на соответствующих фазовых сигналах считывать информацию в параллельную шину или при соответствующей коммутации ключей записывать за один такт в соответствующую часть регистра информацию с параллельной шины.

Регистровые ОЗУ К145ИР1 содержат сдвиговый регистр емкостью 1 Кбит, но в соответствии с форматом оперативной памяти ПМК используется только 1008 бит.

В однокристальных микроЭВМ применяются подобные сдвиговые регистры, но меньшей емкости. В микроЭВМ К145ИК5 два сдвиговых регистра R и M емкостью 36·4 бит каждый и 4-разрядный регистр-аккумулятор S (рис. 3.2). Регистр M через внешние вход и выход включен в системную магистраль, образуя часть оперативной памяти ПМК. Операционный регистр R вместе с аккумулятором S и ячейками для хранения флагов образует быструю (или сверхоперативную) память микроЭВМ.

Емкость регистров R и M обеспечивает хранение в них информации, содержащейся в трех блоках страницы оперативной памяти, в частности в трех регистрах данных. Содержимое трех регистров данных, соответствующих двум операндам и результату операции в формате (2.1), хранится при вычислениях в регистре R. Подобное распределение информации в регистрах R и M упрощает ее адресацию и обработку в арифметическо-логическом устройстве (АЛУ). Так, каждые три тетрады, начиная с младших в формате (2.1), считываются с выхода регистра R в интервалы времени Е1,Е2 и Е3 соответственно, а суммарный интервал пересылки и обработки в АЛУ каждых трех тетрад соответствует длительности разрядных сигналов Д1 ... Д12 (см. рис. 3.2).

Следовательно, сигналами Д*k*E*i*B*j* можно управлять обработкой информации не только в каждой тетраде, но и в каждом разряде регистра R0.

Если операция (например, сложение) над содержимым регистра R выполняется за одну пересылку, то результат вычислений может непосредственно выводиться в те же блоки оперативной памяти, откуда были считаны в регистр R исходные данные. Если требуется многократное повторение операций (например, умножения) над содержимым регистра R, то оно перемещается в кольцевом режиме требуемое число раз. Если это число меньше числа блоков оперативной памяти, то перемещение информации в регистре R продолжается до тех пор, пока в регистре M не появится нужное содержимое. Для этого необходимо, чтобы емкость оперативной памяти была кратной емкости регистров R и M, что и приводит к использованию только 1008 двоичных ячеек регистровых ОЗУ.

Временная адресация и согласование во времени операций обработки информации в микроЭВМ, а также ее обмен с оперативной памятью вычислительной системы обеспечиваются регистром синхронизации (РСХ), формирующим последовательность управляющих сигналов Д1, ..., Д12, Е1, Е2, Е3 и В1, ..., В4. Кроме того, после пересылки каждой страницы оперативной памяти на такте с временным адресом Д12Е3В4 в РСХ формируется синхронизирующий импульс (СИ), передаваемый с ведущей микроЭВМ на другие. Ведущая микроЭВМ К145ИК502 имеет выходы для сигналов Д1, ..., Д12, подаваемых на клавиатуру и индикаторное устройство.

Синхронность обработки информации по заданной прикладной программе в АЛУ или следующих микроЭВМ и обмен содержимым быстрой памяти микроЭВМ и оперативной памяти вычислительной систе-

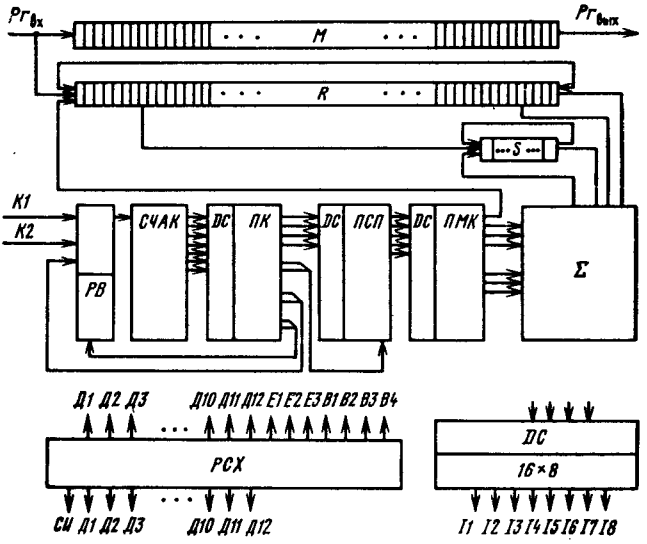


Рис. 3.2

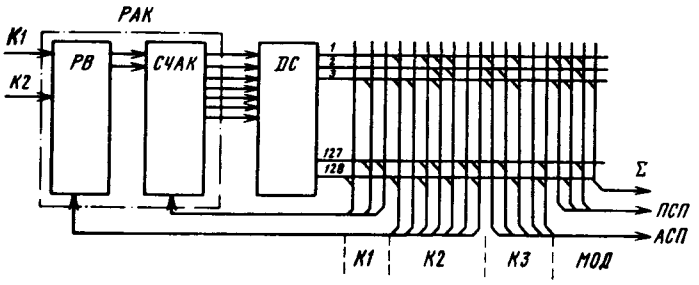


Рис. 3.3

мы обеспечиваются управляющими устройствами микроЭВМ с ПЗУ, где хранится программное обеспечение выполняемых операций, разбитое на три семантических уровня, соответствующих сложности отображаемых операций: память команд (ПК), синхропрограмм (ПСП) и микрокоманд (ПМК). МикроЭВМ К145ИК5 имеет ПК емкостью 128·19 бит, ПСП емкостью 32·3·6·5 бит и ПМК емкостью 32·18 бит, где последние множители указывают разрядность выходных слов. Память синхропрограмм разбита на блоки по 32 слова, формируемые в интервалы времени Е1, Е2 и Е3 каждого из разрядных сигналов Д1, ..., Д12.

Прикладную программу можно отобразить последовательностью операторов

$$<\text{Прикладная программа}> = <\text{ОП}><\text{ОП}><\text{ОП}>\ldots<\text{ОП}>, \quad (3.1)$$

хранящихся в области программ оперативной памяти или вводимых с пульта управления нажатием клавиш.

В рабочем режиме при вводе директив или операторов прикладной программы нажатием клавиш на входы К1 и К2 микроЭВМ поступают разрядные сигналы. В соответствии с этими сигналами на выходе регистра адресов команд (РАК) управляющего устройства формируется последовательность 7-битовых слов — адресов команд (АК):

$$<\text{ОП}> = <\text{АК}><\text{АК}><\text{АК}> \ \ldots \ <\text{АК}>. \quad (3.2)$$

Адреса команд с выхода РАК последовательно подаются на дешифратор ДС памяти команд, который пересылает на одну из 128 горизонтальных шин, соответствующую адресу команды (рис. 3.3), импульс напряжения. Этот импульс через разделительные диоды (показаны на схеме косыми линиями) подается на все или некоторые из 19 вертикальных шин, с выхода которых параллельно считывается 19-битовое слово — команда, состоящая из четырех полей

$$<\text{К}> = <\text{К1}><\text{К2}><\text{К3}><\text{МОД}>. \quad (3.3)$$

Содержимое 7-битового поля К1 представляет собой адрес следующей команды или используется для его определения в соответствии с

содержимым 3-битового поля К2, посылаемым на счетчик адресов команд (СЧАК). Содержимое 5-битового поля К3 является адресом синхропрограммы (АСП), составляющие которой выполняются строго в определенные интервалы времени, поэтому она и называется синхропрограммой. Содержимое первых трех битов 4-битового поля МОД модифицирует выходное слово памяти синхропрограмм, а содержимое последнего бита МОД определяет, пересылается ли содержимое АЛУ в операционный регистр памяти после выполнения соответствующей операции.

Адрес синхропрограммы подается на дешифратор памяти микрокоманд, вырабатывающий импульс напряжения, который подается на одну из 32 горизонтальных шин, соответствующую заданному адресу (рис. 3.4). Вертикальные шины разбиты на три блока, в каждом по 12 групп из 5 шин. На каждый блок подаются управляющие сигналы Е1, Е2 и Е3, а на каждую группу — разрядные сигналы Д1, ..., Д12. В результате в каждом интервале Е1, Е2 и Е3 разрядных сигналов Д1, ..., Д12 на выход памяти синхропрограмм будут подаваться различные или совпадающие 5-битовые слова, являющиеся адресами микрокоманд (АМК). Следовательно, каждая синхропрограмма отображается последовательностью из 36 слов:

$$<\text{СП}> = <\text{АМК}> <\text{АМК}> <\text{АМК}> \ldots <\text{АМК}> \qquad (3.4)$$

При этом каждое из этих слов зависит как от АСП, так и от интервалов времени, в течение которых оно передается:

$$<\text{АМК}> = f < \text{АСП},\ \text{Д}k\text{Е}i> \qquad (3.5)$$

Выходное 5-битовое слово АМК подается на дешифратор памяти микрокоманд (рис. 3.5), вырабатывающий импульс напряжения, который подается на горизонтальную шину, соответствующую заданному АМК, а через разделительные диоды — на все или некоторые из 19 вертикальных шин, образуя напряжения уровня логической 1 или

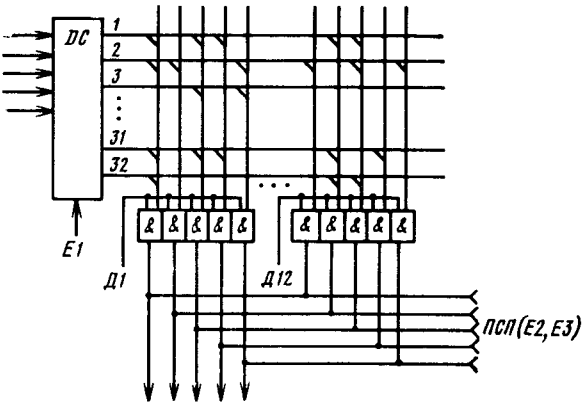


Рис. 3.4

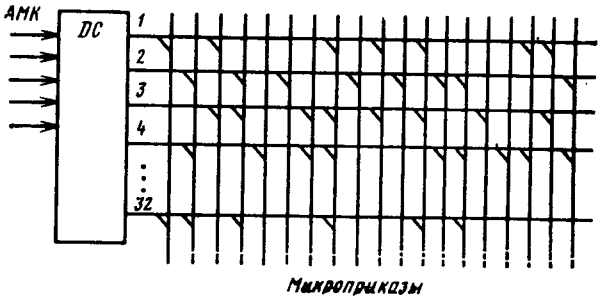


Рис. 3.5

0, называемые *микроприказами* или *микрооператорами*, и, следовательно.

$$<МК> = <МП> <МП> <МП> \ldots <МП> \qquad (3.6)$$

Микроприказы (управляющие напряжения) с выхода памяти микрокоманд передаются непосредственно на логические элементы микроЭВМ, которые соединяют между собой входы и выходы определенных узлов микроЭВМ, если напряжение уровня логической 1.

В зависимости от назначения микроприказы распределены между несколькими полями микрокоманды. Для примера в табл. 3.1 показано распределение микроприказов между полями микрокоманд микроЭВМ К145ИК502 (К745ИК502) и приведено несколько микрокоманд и их адреса (АМК).

Таблица 3.1

Поля микрокоманд одиокристальной микроЭВМ К145ИК502

Вход	α				β				γ·B1			R1			R2	M1	L	S
Выход	$\overline{R0}$	R0	M0	AL	S	$\overline{S}$	R0	6	T	L	1	R0	R33	S	R1/Σ	M0/S	L/П	S/Σ
АМК	Микроприказы																	
01	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1
02	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1
03	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0
04	0	1	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1	1

Первое поле микрокоманды обеспечивает соединение входа α сумматора с прямым R0 и инверсным $\overline{R0}$ выходами регистра R, выходом M0 регистра M и подачу на вход α кода шестнадцатеричной цифры A, когда ячейка L очищена, или нуля, когда ячейка L содержит 1.

Второе поле микрокоманды обеспечивает подачу на вход β сумматора сигнала с прямого S или инверсного $\overline{S}$ выхода регистра-аккумулятора, входа R0 операционного регистра и кода числа 6.

Третье поле обеспечивает подачу на вход γ сумматора в интервалы времени B1 инверсного содержимого ячейки, содержащей 1 в течение времени нажатия любой клавиши на пульте управления, и содержимого ячейки L, а также кода числа 1.

Четвертое поле обеспечивает соединение входа ячейки R1 операционного регистра с выходами R0 операционного регистра, ячейки R33 и S регистра-аккумулятора.

Следующие поля обеспечивают соединение входа ячейки R2 с выходом предыдущей ячейки R1 и выходом АЛУ Σ, входа ячейки M1 с выходом предыдущей ячейки M0 и выходом регистра S, запись переноса в ячейку L и выходного сигнала АЛУ Σ в регистр-аккумулятор.

Таким образом, каждый оператор прикладной программы или директива отображаются (интерпретируются) управляющим устройством последовательностью микрокоманд, управляющих операционными регистрами микроЭВМ в течение определенных моментов времени, соответствующих пересылке через регистры микроЭВМ определенных порций информации.

При включении ПМК ведущая микроЭВМ засылает в динамическую память код «метка», очищая остальные страницы памяти. При вводе операторов набора десятичных знаков нажатием соответствующих клавиш управляющее устройство формирует в операционном регистре коды этих знаков, которые в соответствующие интервалы времени пересылаются в регистры X оперативной памяти. При вводе синтаксических операторов (например, B ↑ или ↔) или операторов засылки содержимого регистра X в регистр памяти управляющее устройство перезаписывает содержимое регистра X в операционный регистр, откуда пересылает его в интервалы времени, соответствующие прохождению требуемой страницы оперативной памяти через регистр M, в заданный регистр памяти данных. При выполнении одноместного функционального оператора содержимое регистра X вызывается в операционный регистр микроЭВМ, в ПЗУ которой хранится программное обеспечение задаваемой операции (сообщение об этом передается от ведущей микроЭВМ), после чего выполняется заданная операция, а ее результат пересылается в регистр X оперативной памяти. Аналогично выполняются двухместные операторы, но в этом случае в операционный регистр соответствующей микроЭВМ вызывается содержимое как регистра X, так и регистра Y.

В режиме программирования коды операторов не интерпретируются выполняемыми микропрограммами, а пересылаются в ячейки программной памяти. Выполнение введенной таким образом в память прикладной программы отличается от ее выполнения в рабочем режиме нажатием клавиш лишь тем, что коды операторов не формируются в РАК управляющего устройства, а поочередно засылаются в него из программной памяти.

Для управления работой индикаторного устройства микроЭВМ К145ИК5 содержит выходной регистр с дешифратором (рис. 3.6), преобразующим выводимую очередную тетраду содержимого регистра дан-

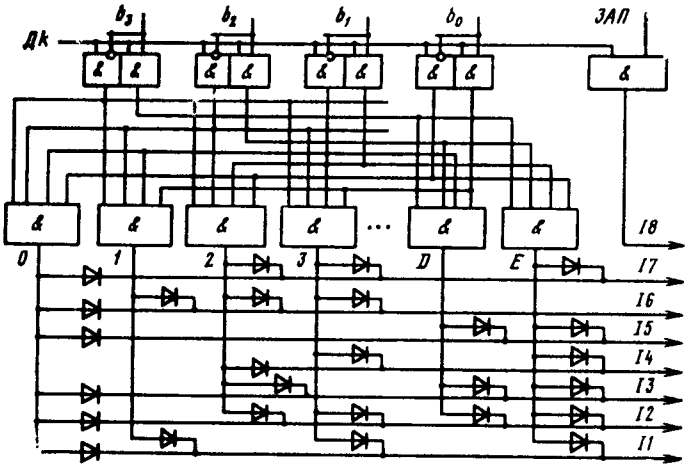


Рис. 3.6

ных в сигналы, подаваемые в соответствующие интервалы времени Д1, ..., Д12 на вертикальные шины, с которых они подаются на все или некоторые из 7 горизонтальных шин, соответствующих сегментным сигналам I1, ..., I8, выводимым ведущей микроЭВМ К145ИК502 на индикатор. Сигнал I8 формируется в течение длительности соответствующего разрядного сигнала Д1, ..., Д8 в зависимости от содержимого ячейки флага запятой в слове состояния процесса. В ПМК семейства «Электроника МК-64» используются лишь сочетания сегментных сигналов для десятичных цифр и запятой, а для отрицательных знаков мантиссы и порядка — набор сегментных сигналов для принятого отображения на дисплее шестнадцатеричной цифры А. При засылке в выходной регистр кода 1111 шестнадцатеричной цифры F сегментные сигналы не подаются, что соответствует отображению пробела в соответствующем знакоместе.

Однокристальные микроЭВМ К145ИК13 (К745ИК13) работают подобным образом, но отличаются большей емкостью и более сложной структурой памяти. В этих БИС для хранения трех страниц памяти формата, принятого для ПМК расширяющегося ряда, также используется регистр М с внешними входом и выходом для включения в системную магистраль, но его емкость $42 \cdot 4 = 168$ бит. Такая же емкость у операционного регистра R и дополнительного регистра SR быстрой памяти. Последний отличается стековой организацией с пересылкой содержимого страниц SR (Дk E1), SR (Дk E2), SR (Дk E3) как в магазинном, так и в кольцевом режиме. Это обеспечивает возможность выполнения операций со скобками, а также организацию подпрограмм с тремя уровнями вложения, что позволяет ускорить вычисления. Кроме того, в быстрой памяти имеются два 4-разрядных регистра общего назначения (РОН), один из которых имеет внешний вход. Общим для микроЭВМ К745ИК13 является расположение программного счетчика в старших тетрадах регистра данных R1.

Управляющее устройство микроЭВМ К745ИК13 имеет относительно большую емкость ПЗУ — память команд 256·23 бит, память синхропрограмм 128·3·3·6 бит и память микрокоманд 68·28 бит, но последовательность уровней интерпретации директив и операторов также соответствует формулам (3.1)—(3.6).

В ПМК расширяющегося ряда с внешними устройствами в качестве микроконтроллера используется микроЭВМ К745ИК18 со средствами параллельного обмена информацией с ВЗУ и последовательного обмена информацией с системной магистралью. Так как емкость регистров М микроЭВМ К745ИК13 и К745ИК18 неодинакова, в последней из этих микроЭВМ не может использоваться как часть оперативной памяти вычислительной системы и обмен информацией реализуется считыванием информации с разрыва системной магистрали через вход регистра М или ввод в системную магистраль с выхода этого регистра.

3.2. МОДЕЛИРОВАНИЕ РАБОТЫ ПРОЦЕССОРА

Функции процессора с АЛУ, быстрой памятью и устройствами управления и обмена информацией с оперативной памятью в рассматриваемых ПМК выполняют однокристальные микроЭВМ, за исключением регистра М. При десятично-двоичном или двоичном кодировании слова разбиваются на тетрады разрядов, содержимое которых отображает десятичные или шестнадцатеричные цифры. Это позволяет свести преобразование информации в основном к обработке «цифра за цифрой» и при рассмотрении работы процессора однокристальных микроЭВМ использовать его упрощенную схему (рис 3.7), содержащую сумматор, два 4-разрядых регистра R и S, а также ячейку L для хранения переноса при сложении операндов и флага переполнения при сложении содержимого старших разрядов чисел.

Работу сдвигового регистра с четырьмя ячейками S4, S3, S2 и S1 для хранения в исходном состоянии разрядов слова соответственно можно описать последовательностью присвоений

S1:=S2; S2:=S3; S3:=S4; S4:=X,

где X — двоичный сигнал, подаваемый на входную ячейку S4. Если такой регистр охвачен кольцом, то при передаче содержимого выходной ячейки S1 на вход ячейки S4 его работа описывается последовательностью присвоений

X := S1; S1=S2; S2:=S3; S3:=S4; S4:= X.

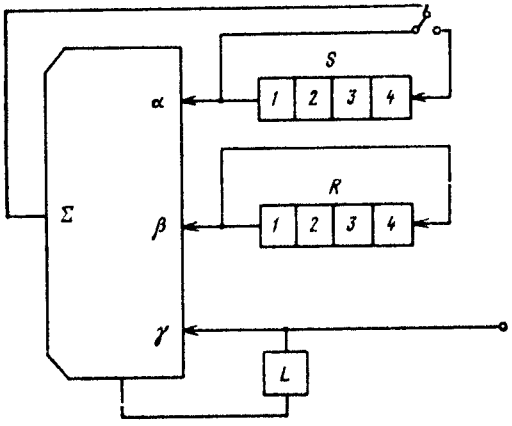


Рис. 3.7

Если в таком регистре хранится двоичное слово 1001 (S1 = 1, S2 = 0, S3 = 0, S4 = 1), то на каждом такте его содержимое будет перемещаться на одну двоичную ячейку (разряд):

Такт	S1	S2	S3	S4
0	1	0	0	1
1	0	0	1	1
2	0	1	1	0
3	1	1	0	0
4	1	0	0	1
5	0	0	1	1

Так как исходное содержимое регистра восстанавливается через каждые четыре такта, то в моменты времени, соответствующие каждому 4-му такту, содержимое регистра будет постоянным.

Состояние сдвиговых регистров R и S в кольцевом режиме описывается последовательностью присвоений

X:=S1; S1:=S2; S2:=S3; S3:=S4; S4:=X;

Y:=R1; R1:=R2; R2:=R3; R3:=R4; R4:=Y.

Тогда пересылка информации в обоих регистрах, например, при исходном содержимом S = 1001 и R = 0011 отображается таблицей

Такт	S1	S2	S3	S4	R1	R2	R3	R4
0	1	0	0	1	1	1	0	0
1	0	0	1	1	1	0	0	1
2	0	1	1	0	0	0	1	1
3	1	1	0	0	0	1	1	0
4	1	0	0	1	1	1	0	0
5	0	0	1	1	1	0	0	1

Заметим, что выбранные начальные состояния регистров через несколько тактов повторяются в разных регистрах, но при смене начальных состояний (например, 1001 и 1010) это не наблюдается.

При моделировании работы сумматора основное внимание будем уделять синхронизации состояний отдельных узлов процессора, а не операциям суммирования, выполняемым логическими элементами сумматора.

Операцию сложения можно описать присвоением C:=X+Y+П, где слагаемые — содержимое выходных ячеек S1 и R1 и ячейки переноса L на каждом такте. Так как эти слагаемые принимают только значения 0 или 1, то сумма C может принимать значения, соответствующие десятичному числу C от 0 до 3, которое можно представить двоичным числом c_2c_1, где содержимое старшего разряда c_2 равно переносу в старший разряд П на следующем такте, Тогда возможны значения:

C = 0 ($c_1 = 0$, $c_2 = 0$) или X = 0, П = 0,
C = 1 ($c_1 = 1$, $c_2 = 0$) или X = 1, П = 0,
C = 2 ($c_1 = 0$, $c_2 = 1$) или X = 0, П = 1,
C = 3 ($c_1 = 1$, $c_2 = 1$) или X = 1, П = 1,

Так как частное от деления c_2, c_1 на 2 равно c_2, c_1, то подобную таблицу соответствий можно задать функциями X(C) и П(C) дискретного аргумента, реализуемыми, например, соотношениями X = 2 * FRC(C/2), П = INT(C/2). В действительности суммирование осуществляется логическими элементами в соответствии, например, с логическими операциями, рассмотренными в гл. 1, но для моделирования принципа синхронизации удобно использовать подобную условную запись.

Если с помощью ключа (рис. 3.7) разорвать кольцевую связь и на вход регистра S подавать сигнал X с выхода сумматора, то при поступлении команды суммирования (например, A0) на данном такте X: = 2*FRC ((X+Y+П)/2); П:=INT((X+Y+П)/2), в противном случае X: = X. Тогда для выполнения операции сложения содержимого двух 4-разрядных регистров S и R потребуется следующая последовательность команд и выполняемых операций:

команда A0 (на 1-м такте) П: = 0; X:=S1; Y:=R1: X:=2*FRC((X + Y+П)/2); П:=INT((X+Y+П)/2); S1:=S2; S2:=S3; S3:=S4; S4:=X; R1:=R2; R2:=R3; R3:=R4; R4:=Y;

команда A1 (на 2-м, 3-м, 4-м тактах) П:=П; X:=S1; Y:=R1; X:=2*FRC ((X+Y+П)/2); П:=INT((X+Y+П)/2); S1:=S2; S2:=S3; S3:=S4; S4:=X; R1:=R2; R2:=R3; R3:=R4; R4:=Y;

команда B (на 5-м и следующих тактах) П:=П; X:=S1; Y:=R1; X:= X; S1:=S2; S2:=S3; S3:=S4; S4:=X; R1:=R2; R2:=R3; R3:=R4; R4:=Y.

Команды A0 и A1 отличаются лишь начальным значением переноса, командой B названа операция перемещения по кольцу содержимого регистров S и R без его изменения. Если, например, исходное содержимое S = 1001 и R = 0011, соответствующее десятичному представлению $1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 9$ и $0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 3$, изменяется в процессе выполнения команд A0, A1 и B, то эти изменения при сложении отображаются таблицей

Такт	П	S1	S2	S3	S4	R1	R2	R3	4R	Команда
0	0	1	0	0	1	1	1	0	0	A0
1	1	0	0	1	0	1	0	0	1	A1
2	1	0	1	0	0	0	0	1	1	A1
3	0	1	0	0	1	0	1	1	0	A1
4	0	0	0	1	1	1	1	0	0	B
5	0	0	1	1	0	1	0	0	1	

В этой таблице символ команды записан после содержимого ячеек на данном такте, так как команда определяет выполнение требуемых микроопераций на следующем такте. Как следует из таблицы, на 4-м такте содержимое регистра S = 1100, что соответствует десятичному представлению $1 \cdot 2^3 + 1 \cdot 2^2 + 0 \times \times 2^1 + 0 \cdot 2^0 = 12 = 9 + 3$. Таким образом, операция сложения 4-битовых слов выполняется за 4 такта, ее начало должно быть точно синхронизировано с исходным расположением содержимого регистров, которое смещается с одинаковой скоростью как при выполнении операции сложений, так и при ожидании следующей операции.

Изменяя вид фуикции X = f(X, Y, П), несложно получить описания и некоторых других операций. Так, если X = Y, то получим операцию передачи содержимого регистра R в регистр S, выполняемую за 4 такта. Обозначив команду выполнения подобной операции C, можно отобразить ее выполнение для ранее выбранного исходного содержимого регистров следующей таблицей:

Такт	S1	S2	S3	S4	R1	R2	R3	R4	Команда
0	1	0	0	1	1	1	0	0	C
1	0	0	1	1	1	0	0	1	C
2	0	1	1	1	0	0	1	1	C
3	1	1	1	0	0	1	1	0	C
4	1	1	0	0	1	1	0	0	B
5	1	0	0	1	1	0	0	1	

Таким образом, содержимое регистра R переписывается в регистр S за 4 такта, причем за время выполнения команды C информация перемещается в ячейках обоих регистров так, как если бы они образовали один 8-разрядный регистр.

Рассматривая только операции, выполняемые за 4 такта, можно упростить описание результатов моделирования процессора, оставляя в таблицах лишь начальное и сформировавшееся через интервалы времени, кратные 4 тактам, содержимое регистров. Тогда, например, описание сложения и перезаписи содержимого регистров можно представить таблицей

Такт	S1	S2	S3	S4	R1	R2	R3	R4	Команда
0	1	0	0	1	1	1	0	0	A
4	0	0	1	1	1	1	0	0	B
8	0	0	1	1	1	1	0	0	C
12	1	1	0	0	1	1	0	0	B
16	1	1	0	0	1	1	0	0	

Эту таблицу можно еще больше упростить, обозначив содержимое регистров R и S шестнадцатеричными цифрами:

Такт	S	R	Команда	Содержание команды
0	9	3	A	$S:=S+R;\ R:=R$
4	C	3	B	$S:=S;\ R:=R$
8	C	3	C	$S:=R;\ R:=R$
12	3	3	B	$S:=S;\ R:=R$
16	3	3		

Используя подобную запись, удобно упрощать и описание команд. Обозначение команд буквами A, B, C, ... можно рассматривать как шестнадцатеричные коды слов, представляющих команду последовательностью управляющих сигналов. Поясним такое представление на примере описания работы сумматора логическими операторами.

Прежде всего заметим, что для суммы $C=X+Y+П$, когда $C=0$ или $C=1$, выходной сигнал $П'=0$, а значение функции X' совпадает с C. Эта функция является результатом логнческих операций $X'=(X\vee Y\vee П)\wedge\overline{П'}$, а если $П'=1$, то $X'=1$ только в случае, когда $X=Y=П=1$. Так как $X'=X\wedge Y\wedge П\wedge П'$,

то для любых значений П′ справедливо соотношение $X' = ((X \vee Y \vee П) \wedge \overline{П'}) \vee \vee (X \wedge Y \wedge П)$. Второе логическое слагаемое записано в этой формуле без множителя П′, так как оно равно 1, только когда $П' = 1$. Это следует из логической функции $П' = (X \wedge Y) \vee (П \wedge X) \vee (П \wedge Y)$ и соответствует словесному описанию: перенос появляется, когда хотя бы два из трех значений X, Y и П равны 1, но если все три значения равны 1, то любые два из них также равны 1.

Для получения X′ необходимо заранее получить значение $\overline{П'}$, что реализуется логическими элементами, инвертирующими двоичные переменные с минимальной задержкой, меньшей длительности такта. Для получения неинвертированного (прямого) или дважды инвертированного значения П требуется еще один инвертор, но так как прямой сигнал используется только на следующем такте, то можно обойтись без дополнительного элемента памяти.

Одиако сумматор содержит не только ячейку L для запоминания значения переиоса, но и источник специального 3-бнтового сигнала $\langle P\rangle = \langle P1\rangle\langle P2\rangle\langle P3\rangle$, указывающего моменты времени, когда нужио запомнить $П := П$ и когда иужио прнсвоить П значения 0 нли 1. Эти сигналы позволяют модифицировать реализуемую сумматором функцню

$$П' = ((X \wedge Y) \wedge P1) \vee (((P2 \wedge П) \vee P3) \wedge (X \vee Y))$$

Различные сигналы P, вводимые для модификации преобразований, соответствуют н различиым значенням П на выходе сумматора:

P3	P2	P1	П
0	1	1	$(X \wedge Y) \vee (П \wedge (X \vee Y))$
1	0	0	$X \vee Y$
0	0	1	$X \wedge Y$

Следовательно, если рассматривать последовательность сигналов P1, P2, P3 как обратиую запись двоичиого слова P3P2P1, то трем описаииым вариантам этого слова соответствуют их шестнадцатеричные коды P=3, P=4 и P=1.

Рассмотренная модель сумматора с двумя 4-разрядными сдвиговыми регистрами соответствует архитектурному уровню микрокоманд, выполняемых на каждом такте и обрабатывающих двоичные разряды слов. Пересылка или обработка каждой тетрады слова выполняется последовательностями одинаковых микрокоманд за 4 такта. Следовательно, можно рассматривать работу процессора в течение интервалов времени, равных 4 тактам, которые можно назвать *макротактами*. Результаты операций над тетрадами слов в течение каждого макротакта удобно представлять десятичными или шестнадцатеричными цифрами, отображающими содержимое каждой тетрады, что также существенно упрощает моделирование работы микроЭВМ. При этом каждому макротакту соответствует временной адрес Д1Е1, Д1Е2, Д1Е3, Д2Е1, Д2Е2, ..., определяющий как относительные интервалы времени обработки определенных тетрад слов в соответствии с их форматом, так и распределение этих тетрад в регистрах памяти. Последовательность микрокоманд, выполняемых в течение одного или нескольких макротактов, удобно рассматривать как *макрокоманду*, соответствующую более высокому архитектурному уровню работы микроЭВМ, отображая ее результаты в общем случае шестнадцатеричными цифрами.

Обозначая 4-разрядный регистр R (I) с индексными скобками, можно рассматривать такую ячейку памяти как элемент массива таких же ячеек с порядковым номерам (индексом) I. Тогда моделью динамической памяти микроЭВМ может служить последовательность присвоений

$$X:=R(1);\ R\ (1):=R(2);\ R(2)=:=R\ (3);\ \ldots;\ R\ (N):=R(N+1);\ \ldots$$

В замкнутом в кольцо регистре (кольцевом) с подобными ячейками будет последовательность присвоений с выходными сигналами X, подаваемыми на вход цепочки ячеек. Для цепочки ячеек от 0 до $K-1$ (с входной ячейкой R ($K-1$) и выходной R (0)) эту запись можно упростить, воспользовавшись оператором цикла:

X:= R(0);
для $I=0$ шаг 1 до $K-2$ выполнить $R(I):=R(I+1)$; $R(K-1):=X$.

Следовательно, после выполнения первого присвоения, определяющего значение переменной X на выходной ячейке, должны быть выполнены все пересылки в смежных регистрах с меньшими номерами, обусловленные соответствующими макрокомандами, в результате будет получено значение X во входной ячейке.

Если предписано X : = X, то в замкнутом в кольцо регистре исходное содержимое ячеек восстанавливается через каждые K макротактов. Так как доступ к регистру R реализуется в рассматриваемом случае только через сигнальную шину X, соединяющую выходы ячеек R (0) и $R(K-1)$, то для преобразования информации необходимо выбрать соответствующий интервал времени.

Если организовать счетчик в ячейке R(0) (записывая в нее число кольцевых сдвигов) при нулевом содержимом регистра R, то введенный на первом макротакте в ячейку $R(K-1)$ сигнал $X=1$ появится в выходной ячейке $R(0)$ через K тактов. Для организации счетчика обозначим первую команду присвоением $X:=X+1$, а последующие—присвоениями $X:=X$. Тогда после выполнения первой команды значение X увеличится на единицу, будет присвоено ячейке $R(K-1)$ и попадет в ячейку R(0) лишь через $K-1$ макротактов, «холостых», соответствующих командам $X:=X$. Подобную последовательность команд на K макротактах (R (0) :=: R (0)+1 назовем макрокомандой. Следовательно, эта макрокоманда отображается на 1-м макротакте последовательностью микрокоманд, добавляющих единицу к содержимому тетрады, и на $(K-1)$-м макротакте — микрокомандами, обеспечивающими сдвиг содержимого ячеек кольца на одну ячейку. Для макрокоманды, изменяющей содержимое любой другой ячейки $R(I)$ регистра R, необходимо в цепочке микрокоманд на I-м микротакте ввести требуемое преобразование. В общем случае макрокоманда в течение каждого макротакта обрабатывает содержимое нескольких ячеек и время на холостые пересылки информации не затрачивается.

Таким образом, при анализе и моделировании процессов в однокристальных микроЭВМ можно не указывать циклы присвоений, реализующих сдвиги на каждом цикле работы микроЭВМ, ограничиваясь записью макрокоманды.

Для удобства анализа можно представить содержимое регистра неподвижным, а обрабатывающий информацию процессор (в данном случае преобразующий значение X в X) — перемещаемым на каждом макротакте на одну ячейку вдоль кольцевого регистра в соответствии с изменением индекса X. Подобный подход аналогичен переходу к подвижной системе координат для движущихся объектов, с положением которых связано начало отсчета координат. При этом номер *I* ячейки R (*I*), равный адресу *I* содержимого этой ячейки, однозначно определяет и интервал времени, соответствующий номеру макротакта.

Так как в ПМК последовательного действия информация одновременно перемещается в нескольких кольцевых сдвиговых регистрах разной длины, то необходимо четко контролировать изменение номера (индекса) ячеек каждого массива в кольцевых регистрах. Поэтому целесообразно выбрать такую нумерацию макротактов, которая обеспечит наибольшую простоту определения номера для доступа к нужному содержимому ячеек.

В однокристальных микроЭВМ К145ИК18 (К745ИК18) кроме регистра M содержатся кольцевые регистры R из 36 ячеек R (*I*) и RIN (*J*) из 3 ячеек. Поэтому номер макротакта удобно представить в системе счисления РДЕ, где Е принимает значения от 0 до 2 (в интервалы времени E1, E2 и E3 соответственно), Д — от 0 до 11, а Р — от 0 до нужного значения. Тогда для указания индекса *J* целесообразно использовать поле (часть слова) Е, а для индекса *I* — формулу $I = Д \times 3 + Е$. Массив R, номер ячейки которого определятся по этой формуле, можно рассматривать как двумерный массив R (*K*, *J*), разбитый на три массива RA (*K*), RB (*K*) и RC (*K*). Тогда замкнутая в кольцо последовательность ячеек массива R отображается последовательностью символов RA (0), RB (0), RC (0), RA (1), RB (1), RC (1), ..., RA (*N*), RB (*N*), RC (*N*) и на каждом макротакте окажется доступной ячейка с определенным адресом R (*K*, *J*). Подобное разбиение элементов массива, как показано далее, имеет существенное преимущество при моделировании процессов в однокристальных микроЭВМ серии К145 (К745) и упрощает проверку и создание их программного обеспечения.

Рассмотренные способы описания работы сдвиговых регистров позволяют построить модель архитектуры однокристальных микроЭВМ, перечислив все элементы памяти и допустимые для устройства управления операции пересылки и преобразования информации. Так как внутренняя организация обмена информацией различна для микроЭВМ К145ИК13 (К745ИК13) и К145ИК18 (К745ИК18), используемых в ПМК расширяющегося ряда, то рассмотрим отдельно работу каждой из них.

3.3. ОПИСАНИЕ РАБОТЫ МИКРОЭВМ К145ИК13 (К745ИК13)

Оперативная память этой микроЭВМ состоит из регистра M, а быстрая или сверхоперативная — из кольцевых регистров R и ST, каждый из которых содержит 42 ячейки. Доступ к определенной ячейке кольцевого регистра возможен лишь через одинаковые промежутки времени, необходимые для полного кольцевого сдвига содержимого регистра и подключения требуемой ячейки к шине, соединяющей выходную ячейку регистра. Однако в микроЭВМ имеются также два регистра общего назначения S и S1, доступ к которым возможен на каждом очередном макротакте.

Адрес ячеек регистров M, R и ST будем обозначать индексными скобками, например R (*I*), где *I* принимает значения от 0 до 41. Для нумерации макротактов используем формулу $I = 3 \times Д + Е$, где Д циклически изменяется от 0 до 13, а Е — от 0 до 2. Уменьшение или увеличение индекса *I* целесообразно описывать по дополнению до 42 (по модулю 42), так, $I + 1 = 0$ для $I = 41$, а $I - 1 = 41$ для $I = 0$.

Стробовые сигналы Д и Е, определяющие соответствующие интервалы времени, вырабатываются регистром синхронизации микроЭВМ, Поэтому в каждый интервал времени, соответствующий макротакту с временным адресом ДЕ, известно значение индекса *I*, указывающего адрес содержимого ячейки кольцевого регистра. При описании работы микроЭВМ приходится использовать команду, уменьшающую индекс *I* на единицу по модулю 42, для чего можно воспользоваться традиционной записью итерационного цикла.

Устройство управления микроЭВМ обеспечивает формирование последовательности микрокоманд, определяющих операции преобразования, выполняемые АЛУ, и операции пересылок, выполняемые аппаратурными средствами. Многоуровневая организация управления, уменьшающая общую длину управляющей информации и потребность в аппаратурном обеспечении, облегчает также анализ работы микроЭВМ, отвечает принципам структурированного программирования.

Рассмотрим операции преобразования и пересылки информации. Основным узлом, выполняющим арифметические операции, является сумматор, три входа α, β и γ которого осуществляют операцию логического сложения (дизъюнкцию) сигналов, поступающих от различных источников. Специальные сигналы управления, вырабатываемые на нижнем уровне управления, определяют, сигналы каких именно источников подаются на входы сумматора. Для программирования варианта управления входами сумматора используется бинарная последовательность, биты которой указывают на подключаемые источники. Наборы таких последовательностей записаны в памяти микрокоманд, что позволяет по адресу микрокоманды длиной только 6 бит выбрать один из заранее подготовленных вариантов управления. Таким образом, микрокоманда отображается адресом длиной 6 бит и реализуется словом микроприказов длиной 32 бит, что позволяет сократить объем информации а значит, упростить программирование.

Память микрокоманд фактически содержит описание системы микрокоманд, используемых программистом при составлении программы работы микроЭВМ для решения конкретной задачи. Возможность изменения содержимого памяти микрокоманд обеспечивает перестраиваемость этой системы по усмотрению программиста. Так как архитектура микроЭВМ задается системой микрокоманд, то подобная возможность позволяет программисту влиять на архитектуру нижнего уровня.

Выбор системы микрокоманд при моделировании работы микроЭВМ на рассматриваемом уровне можно осуществить по крайней мере двумя путями (выбор такого пути может оказаться особенно существенным при моделировании работы микроЭВМ на универсальной ЭВМ другого класса). В первом варианте используется массив внутренней памяти микрокоманд (ВПМК), каждый элемент которого позволяет принимать решение о выполнении операции присвоения, допустимой в моделируемом устройстве. Для выбора из массива ВПМК нужного элемента используют адрес микрокоманды, определяемый содержимым ячейки РАМК.

В микроЭВМ К145ИК13 на вход α сумматора могут быть поданы прямой и инверсный сигналы с выхода регистров R, M и S. Математическое описание должно обеспечивать поочередную проверку разрядов управляющего слова, определяющего прохождение сигналов, и при необходимости моделировать логическое сложение на входе α.

Определить содержимое k-го разряда слова $a_n \ldots a_1 a_0$, отображаемого десятичным числом $A = a_n \cdot 2^n + a_{n-1} 2^{n-1} + \ldots + a_1 \cdot 2^1 + a_0 \cdot 2^0$, можно следующим образом. Частное от деления A на 2^k представляет действительное число $a_n a_{n-1} \ldots a_k, a_{k-1} \ldots a_0$ с запятой после содержимого k-го разряда. Если теперь выделить целую часть частного и разделить ее на 2, то дробной частью результата и будет a_k. Умножив ее на 2, получим число a_k, равное 0 или 1. Это можно отобразить формулой

$$a_k = 2 \times \mathrm{FRC}\,(\mathrm{INT}\,(A/2^k)/2).$$

Аналогично можно получить тот же результат по формуле

$$a_k = \mathrm{INT}\,(2 \times \mathrm{FRC}\,(A/2^k/2)).$$

Так как для выполнения логического сложения необходимо лишь определить, равно ли содержимое k-го разряда нулю (в противном случае оно равно единице), то алгоритм логического сложения (дизъюнкции) при определении равенства нулю содержимого разрядов управляющего слова можно представить следующим образом:

1. Управляющее слово из массива ВПМК передать для анализа в ячейку R или R := ВПМК (РАМК).
2. Принять $\Sigma_\alpha = 0$.
3. Если $\mathrm{FRC}\,(P/2) \neq 0$, то $\Sigma_\alpha := \Sigma_\alpha \vee R\,[1]$.
4. R := INT $(P/2)$.
5. Если $\mathrm{FRC}\,(P/2) \neq 0$, то $\Sigma_\alpha = \Sigma_\alpha \vee \overline{R}\,[1]$.

6. R := INT (P/2).
7. Если FRC (P/2) $\neq$ 0, то $\Sigma_\alpha := \Sigma_\alpha \vee M[I]$.
8. R := INT (P/2).
9. Если FRC (P/2) $\neq$ 0, то $\Sigma_\alpha = \Sigma_\alpha \vee S$.

Сама процедура логического сложения представляет собой аналогичную последовательность операторов, но ее можно отобразить с помощью описания цикла. Если учесть, что для одинарных переменных справедливы соотношения $a \vee b = (a + b) - (a \times b)$; $a \wedge b = a \times b$; $a \overline{\vee} b = (a + b) - 2 \times a \times b$, и $\overline{a} = 1 - a$ (символы логических операций рассмотрены в гл. 2), то операцию дизъюнкции можно отобразить описанием

X := A + B; для K = 1 шаг 1 до 4 выполнить (X := X — 2 ↑ (K — 1) × INT (2 × FRC (A/2 ↑ K)) × INT (2 × FRC (B/2 ↑ K))); A := X

В этом описании использован стандартный оператор цикла и символ ↑ возведения в степень алгоритмического языка Аналитик (см. гл.1).

Следовательно, моделирование работы АЛУ по первому варианту предполагает использование одной достаточно длинной программы (которая может быть заменена многократными операциями в цикле), содержащей такое число условных операторов, которое равно числу разрядов управляющего слова. Должен также быть задан массив ВПМК, так как его содержимое определяет выбор системы микрокоманд.

Второй вариант моделирования предусматривает описание каждой микрокоманды отдельной процедурой (подпрограммой). Программист, осуществляя выбор системы микрокоманд, составляет набор таких микропрограмм. Вместо адреса массива ВПМК определяется адрес подпрограммы (адрес микрокоманды), реализующей микрокоманду. Так как текст подпрограмм относительно легко читается, то этот вариант обладает большей наглядностью.

Пусть, например, в приведенном выше примере по выбранному в РАМК адресу из памяти МК прочитан код R := ПМК (РАМК) = 3. Предположим, что для R (I) = 5 номер ячейки I = 2. Тогда после присвоения Σ_α := 0 на первой проверке FRC (3/2) $\neq$ 0 будет выполнена операция $\Sigma_\alpha = \Sigma_\alpha \vee$ R (I) = 5. Тогда R := INT (3/2) = 1, и на следующей проверке FRC (1/2) $\neq$ 0, что соответствует $\Sigma_\alpha := \Sigma_\alpha \vee \overline{\text{R}}$ (I).

Для ячейки максимальное содержимое 1111 соответствует шестнадцатеричному коду F = 15 и, следовательно, дополнение (инверсия) содержимого ячейки R (I), равное в шестнадцатеричном коде R (2) = 5, будет равно $\overline{\text{R}}$ (2) = 15 — 5 = 10. Представив в шестнадцатеричном коде дизъюнкцию $\Sigma_\alpha = 5 \vee 10$ в соответствии с приведенной выше формулой, получим Σ_α = 15.

Выбор допустимых источников сигналов для входов сумматора позволяет осуществлять операции сложения, вычитания, десятичной коррекции (переноса при шестнадцатеричном представлении содержимого регистра R (I) > 9), сложения с константой и маскирования отдельных разрядов для проверки. Все операнды, взятые из регистров R, M и ST, определяются по текущему значению индекса I.

Результат операции, выполненной сумматором, может быть переслан в ячейки R [I], R[I — 1], M [I], ST [I], S и S1. Кроме того, перенос (дополнительный 5-й разряд содержимого сумматора) может быть записан в специальную ячейку L. Поэтому при выполнении цепочки микрокоманд могут использоваться значения содержимого ячеек L, S и S1, полученные на предыдущих тактах.

За один сдвиг могут быть выполнены не все операции. Например, микрокоманда S := S + 1 может быть повторена, если необходимо прибавить к регистру S произвольную константу от 2 до 15. Чтобы ограничить число микрокоманд, на входы сумматора могут быть поданы аппаратурно подготовленные константы 8,2 и 6. Особую роль играет команда $10\times\overline{L}$, так как при умножении на $\overline{L}$ значение константы изменяется в зависимости от значения L. Если L = 0, то $\overline{L}$ = 1 и константа равна 10, в противном случае она равна нулю. Примером использования такой константы является операция десятичного сложения.

В соответствии с табл. 1.1 двоичный код цифр от 0 до 9 легко получить, последовательно разбивая представленное цифрой натуральное число на составляющие 8,4, 2 и 1. Предположим, что сумму содержимого ячеек R (7) и R (8) нужно записать в ячейку R (8), а исходное содержимое ячеек R (7) и R (8) не превышает 9. В этом случае возможны три исхода:

1) сумма не превышает 9 — заносится в ячейку R (8);

2) сумма больше 9, но не больше F = 15 (не возник «шестнадцатеричный» перенос в следующую ячейку) — вычесть из результата число A = 10 и установить признак десятичного переноса L := 1;

3) сумма больше F = 15 (появился перенос в следующую тетраду), а результат равен разности суммы и (F + 1) — перенос заносится в ячейку L, а к сумме (для обеспечения десятичного результата) добавляется 16—10 = 6.

В десятичной системе счисления модель сложения двух десятичных однозначных чисел описывается последовательностью операторов

X:=X+Y; L :=0; если X > 9, то X := X — 10; L := 1.

Для упрощения анализа ситуаций, возникающих при шестнадцатеричном сложении десятичных чисел, изменим несколько порядок вычислений. Так как на сумматор могут быть поданы только выходные сигналы ячеек регистров R, M и ST с индексом I текущего сдвига, то нельзя записывать сумму R (7) + R (8), так как доступ к ячейкам с нужным содержимым реализуется на различных макротактах I.

Рассмотрим следующую последовательность операций:

I=7: S: =R(7)+6;

I=8: S: =R(8)+S; установить L: =1 при переносе;

I=9: R(8): =A×$\overline{L}$+S;

Добавление 6 к числу, не превышающему 9, дает число, не превышающее F = 15, и переноса не возникает. Поэтому в S будет записано

число, однозначно соответствующее исходному содержимому ячейки R (7). Номер макротакта соответствует номеру (индексу) ячейки регистра, выходной сигнал которой подается на сумматор, а константа 6 может быть подана на другой вход сумматора.

На 8-м такте ($I = 8$) происходит шестнадцатеричное сложение содержимого ячеек R (8) и S со следующими исходами:

1. R (8) + R (7) $\leqslant 9$, тогда R (8) + S = R (8) + R (7) + 6 $\leqslant$ $\leqslant$ F = 15, перенос не возникает, но для получения десятичной суммы следует вычесть 6.

2. R (8) + R (7) + 6 $>$ F, возникает перенос, но R (8) + $R(7)$ является десятичной суммой, для которой перенос можно рассматривать как десятичный. А так как сложение шестнадцатеричное, то результатом будет десятичная сумма. Это подтверждается и описанием шестнадцатеричного сложения:

X := X + Y; L : = 0; если X $>$ 15, то X := X — 16; L :=1;

Пусть Y = Z + 6, L:= 0, тогда X := X + Z + 6, L := 0; если X $>$ 15, то X := X—16; L := 1. Но проверка X $>$ 15 соответствует проверке X — 6 $>$ 9, а оператор X := X — 16 — оператору X := (X — 6) — 10. Поэтому можно записать: X — 6: = X + +Z; L :=0; если X — 6 $>$ 9, то X := (X — 6) — 10; L := 1. Следовательно, при появлении шестнадцатеричного переноса содержимое результата соответствует десятичной сумме, а при отсутствии переноса для получения десятичной суммы следует вычесть 6 из суммы.

На 9-м макротакте ($I = 9$) выполняется условное вычитание 6, причем индекс ячейки регистра R равен 8, а не 9. В этом случае возможно использование ячейки R [I — 1] в качестве приемника сигнала сумматора, а вычитание единицы выполняется по модулю 42 и разность 0—1 равна 41. По микрокоманде, соответствующей оператору $A \times \overline{L}$, содержимое L не изменяется, хотя возникает шестнадцатеричный перенос, так как S $\geqslant$ 6, если L = 0, т. е. именно в том случае, когда $A \times \overline{L}$ = 10. Этот перенос не используется потому, что прибавление A = 10 соответствует вычитанию 6. Проверим это утверждение. Пусть X $\geqslant$ 6, тогда X := X + 10; L :=0 (так как обязательно новое значение X $\geqslant$ 16), и если X $>$ 15, то X:= X — 16; L := 1. Следовательно, последовательность операций X:= X + 10; X :=X — 16; L := 1 эквивалентна последовательности X:= X — 6; L :=1.

Так как сложение двоичного кода A с кодом 6 дает нули в первых четырех разрядах с переносом 1, то его можно рассматривать как обозначение отрицательного числа—6 в дополнительном коде. Сумма отрицательного числа и его модуля равна нулю, как и при сложении кодов A и 6 в результате, в связи с чем перенос 1 можно игнорировать.

В общем случае при выполнении арифметических операций над числами в дополнительном коде необходимо различать положительные и отрицательные числа. Если число отрицательное, то предполагается заем переноса заранее, поэтому при появлении переноса он сокращается с заемом и не учитывается.

Назначение регистра L не ограничивается десятичным переносом и при выполнении операций над многоразрядными десятичными числами приходится учитывать переносы из разряда в разряд. Например, при сложении двухразрядных десятичных чисел возможна такая последовательность микрокоманд:

```
I=7:   S:=R(7)+6;
I=8:   S:=R(8)+S; установить L по переносу;
I=9:   R(8):=A×L̄+S;
I=10:  S:=R(10)+6;
I=11:  S:=R(11)+S+L; установить L по переносу;
I=12:  R(11):=A×L̄+S;
```

В этом случае проявляются особенности архитектуры рассматриваемой микроЭВМ. Так как десятичное сложение с коррекцией заняло 3 макротакта, следующие десятичные цифры, отображающие однозначные натуральные числа, приходится выбирать из ячеек R(10) и R(11), пропустив R(9). Заметим, что содержимое очередных разрядов десятичного числа хранится в ячейках регистра R с приращением индекса I на 3, например R(7) и R(10), R(8) и R(11). Если использовать рассмотренную ранее нумерацию Д и Е, то всем разрядам десятичного числа будет соответствовать одинаковый индекс. Операция на макротакте 12 в рассмотренном примере переносится при сложении чисел с большей разрядностью на завершающий такт. Вместе с этим значением переноса L может оказаться признаком ветвления программы, появляющимся при моделировании верхних уровней управления.

Для выполнения операции вычитания необходимо перевести вычитаемое в дополнительный код и выполнить сложение. Дополнительным кодом десятичной цифры X будет 10—X, что можно записать как 10—X=— X mod 10. В общем случае операцию вычитания 9—3=6 можно заменить операцией сложения 9+7=6 mod 10, где 7—дополнительный код — 3. Смысл такой замены можно пояснить последовательностью операций: 9—3 = 9+(10—3)—10=9+7—10=9+7 mod 10. Запись числа—3 в виде 7—10 и соответствует заему из старшего разряда при замене вычитания сложением.

Для перевода отрицательного числа в дополнительный десятичный код по формуле 10 — X следует учесть, что операция инверсии над четырехбитовым двоичным представлением шестнадцатеричной цифры моделируется вычитанием $\overline{X} = 15-X$. Если построить последовательность микрокоманд, выполняющих десятичное вычитание R(8):= = R(8) — R(7), то на 7-м макротакте следует ввести операцию перевода R(7) в десятичный дополнительный код, добавив необходимые для последующих операций константы 6: S:= $\overline{R}(7) + 1$, что эквивалентно S := 15 — R(7) + 1 = (10 — R(7)) + 6. Тогда

```
I=7:   S=R̄(7)+1;
I=8:   S:=R(8)+S; установить L по переносу;
I=9:   R(8):=A×L̄+S;
```

Подобная последовательность операций выполнима микроЭВМ К145ИК13, но анализ этого алгоритма показывает, что вычитание, выполненное подобным образом, имеет особое состояние, приводящее к ошибочному результату. Рассмотрим это состояние, так как подобный анализ чрезвычайно важен для составления алгоритмов работы микроЭВМ.

Проверяя выполнение рассмотренного алгоритма вычитания для различных чисел, можно убедиться, что при R (7) = 0 получим следующий результат:

при $I = 7$ S := 0, так как $\overline{R}$ (7) = F, а F + 1 даст нули в четырех разрядах результата с единицей переноса;

при $I = 8$ S := R(8); L := 0, так как при прибавлении 0 перенос не возникает;

при $I = 9$ R(8) := R (8) + A, так как $\overline{L} = 1$.

Результат ошибочный: при вычитании нуля он не должен изменяться. Ошибку в описании легко исправить, приняв следующую последовательность микроопераций:

$I = 7$: S := $\overline{R}$(7);
$I = 8$: S := R(8) + S + 1; установить L по переносу;
$I = 9$: R(8) := S + A × $\overline{L}$;

Перенос добавления единицы из 7-го макротакта на 8-й приводит к тому, что на 7-м макротакте S = F и на 8-м макротакте при сложении обязательно возникнет перенос, который скомпенсирует действие коррекции кодом A на 9-м макротакте.

При вычитании многоразрядных чисел дополнительный код образуется как 10 — X только для младшего разряда, а для остальных 9 — X, что соответствует предполагаемому заему единицы из старших разрядов, например 347—125=347—1·100—2·10—5=347—1·100—10+5. Таким образом, операцию вычитания двухразрядных десятичных чисел можно представить описанием

$I = 7$: S := $\overline{R}$(7);
$I = 8$: S := R(8) + S + 1; установить L по переносу;
$I = 9$: R(8) := A × $\overline{L}$ + S;
$I = 10$: S := $\overline{R}$(10);
$I = 11$: S := R(11) + S + L; установить L по переносу;
$I = 12$: R(11) := A × $\overline{L}$ + S;

Если сравнить первые три микрооперации с тремя последующими, то единственное отличие — добавление 1 на 8-м макротакте вместо L на 11-м макротакте (при сравнении с операцией сложения можно заметить, что там вместо сложения с 1 сложение с 0). Если записать последовательность микроопераций десятичного сложения чисел большей разрядности, то окажется, что все тройки микроопераций над старшими разрядами, как и следует ожидать, идентичны, так как различия появляются только в номере макротакта, на котором выполняется

операция. Например, вычитание двух многоразрядных чисел можно представить описанием

$I=7$: $S:=\overline{R}[I]$;
$I=8$: $S:=R[I]+S+1$; установить L по переносу;
$I=9$: $R[I-1]:=A\times\overline{L}+S$;
$I=10$: $S:=R[I]$;
$I=11$: $S:=R[I]+S+L$; установить L по переносу;
$I=12$: $R[I-1]:=A\times\overline{L}+S$;
$I=13$: $S:=\overline{R}[I]$;
$I=14$: $S:=R[I]+S+L$; установить L по переносу;
$I=15$: $R[I-1]:=A\times L+S$;
. . . . . .

Соответствующая последовательность микрокоманд выполняется при вычитании двух чисел, из которых десятичные цифры вычитаемого представлены ячейками R (1), R (4), R (7), R (11), R (14), R (17), R (20), R (23) и уменьшаемого — ячейками R (2), R (5), R (8), R (12), R (15), R (18), R (21), R (24) регистра R с перечислением номеров ячеек от младших разрядов чисел к старшим. Например, для моделирования вычитания десятичных чисел

$$\begin{array}{r} 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8 \\ -\quad\quad\quad\quad\quad 9\ 8\ 7 \\ \hline 1\ 2\ 3\ 4\ 4\ 6\ 9\ 1 \end{array}$$

вначале следует занести коды цифр в ячейки регистра R : R (2)=8; R (5) = 7; R (8) = 6; R (11) = 5; R (14) = 4; R (17) = 3; R (20) = = 2; R (23) = 1; R (1) = 7; R (4) = 8; R (7) = 9; R (10) = R (13) = = R (16) = R (19) = R (22) = 0.

Для сокращения записи алгоритма решения рассматриваемой задачи используем оператор цикла:

$I=1$; $S:=\overline{R}[I]$;
$I=I+1$; $S:=R[I]+S+1$; установить L по переносу;
$I=I+1$; $R[I-1]:=A\times\overline{L}+S$;

для $K = 0$ шаг 1 до 6 выполнить

$I = I + 1$; $S:=\overline{R}[I]$;
$I = I + 1$; $S:=R[I] + S + L$; установить L по переносу;
$I = I + 1$; $R[I - 1]:=A \times \overline{L} + S$;

После выполнения операций, описываемых этим алгоритмом, разность исходных чисел будет записана в ячейках регистра R, где перед началом операций находилось уменьшаемое число, а вычитаемое сохраняется без изменений. Вычитание выполняеся за 24 такта.

Подобным образом описывается и операция сложения десятичных чисел, например добавление числа 13 к вычитаемому. Для записи числа 13 в ячейки регистра R используем описание R (0) := 3; R (3) :=

: =1; для $I = 6$ шаг 3 до 21 выполнить (R [I] : = 0). Операцию сложения следует начать с нулевого такта:

I:=0; S:=R[I] + 6;
I:=I + 1; S:=R[I] + S; установить L по переносу;
I:=I + 1; R[I — 1]:=A × $\overline{L}$ + S;

для $K = 0$ шаг 1 до 6 выполнить

I:=I + 1; S:=R[I] + 6;
I:=I + 1; S:=R[I] + S — L; установить L по переносу;
I:=I + 1; R[I — 1]:=A × $\overline{L}$ + S;

После выполнения этого алгоритма $I = 23$, а сумма 1000 будет храниться в ячейках с номерами соответственно 22, 19, 16, 13, 10, 7,4, 1.

Для сокращения объема хранимой управляющей информации, необходимой для выполнения последовательности микроопераций с замкнутыми циклами, предусмотрена возможность циклического повторения группы микрокоманд. Устройство синхронизации позволяет считывать микрокоманды из ВПМК в такой последовательности:

I (J) = 0 (0), 1 (1), 2 (2), 3 (3), 4 (4), 5 (5), 6 (3), 7 (4), 8 (5), 9 (3), 10 (4), 11 (5), 12 (3), 13 (13), 14 (5), 15 (3), 16 (4), 17 (5), 18 (3), 19 (4), 20 (5), 21 (3), 22 (4), 23 (5), 24 (6), 25 (7), 26 (8), 27 (0), 28 (1), 29 (2), 30 (3), 31 (4), 32(5), 33 (6), 34 (7), 35 (8), 36 (0), 37 (1), 38 (2), 39 (3), 40 (4), 41(5),

где I — номер такта; J — номер микрокоманды. Из этой последовательности видно, что каждому номеру такта I однозначно соответствует номер от 0 до 8, но обратной однозначности нет, так как некоторым J соответствуют несколько значений I. Называя J номером микрокоманды, мы подчеркиваем возможность уменьшения числа микрокоманд, которые необходимо указать для всей последовательности, но в этом случае возникает обязательная периодичность повторения одинаковых номеров микрокоманд, что накладывает ограничения на способы размещения представлений чисел в ячейках регистра.

3.4. ФОРМИРОВАНИЕ МИКРОКОМАНД

Для более четкого представления о способе использования номеров микрокоманд при работе микроЭВМ необходимо рассмотреть аппаратурные средства формирования микропрограмм, отображающих микрокоманды. В микроЭВМ представляющая микрокоманду микропрограмма представляет собой бинарное слово, хранящееся в блоке памяти микрокоманд, причем для вызова микропрограммы и ее выполнения на вход этого блока подается адрес микропрограммы, который можно рассматривать как код микрокоманды. Микропрограмма указывает, какие именно связи в АЛУ и между регистрами включаются на такте, соответствующем данной микрокоманде. Полный список допустимых связей описывается следующим образом.

На вход α сумматора могут быть поданы сигналы, отображающие содержимое ячеек R [I], M [I], ST [I], $\overline{\text{R}}$ [I], A $\times \overline{\text{L}}$, S и константу 4, причем каждый из этих источников информации может быть подключен как раздельно, так и совместно. Логическое сложение многобитовых кодов, одновременно подаваемых на вход α, выполняется по битам одного порядка. На вход β сумматора могут быть поданы выходные сигналы S, $\overline{\text{S}}$, S1, константы 6 и 1. На вход γ, предназначенный для приема переноса, могут быть поданы однобитовые сигналы L, $\overline{\text{L}}$ и $\overline{\text{T}}$. Результат сложения в сумматоре $\Sigma := \alpha + \beta + \gamma$ также может служить источником информации, причем полученный при суммировании перенос может быть передан в ячейку L.

На вход ячейки R [I] могут быть поданы выходные сигналы Σ, S, R [$I + 3$] и R [I]. Сигналы Σ и S могут подаваться одновременно с их логическим сложением. Если при этом на тот же вход подать выходиой сигнал R [I], то возможны логические суммы R [I] $\vee$ S, R[I] $\vee$ Σ, R [I] $\vee$ S $\vee$ Σ. Сигнал от R [$I + 3$] должен подаваться без других сигналов.

Варианты связей с приемником R [I] выбираются кодом управления длиной 3 бит, составляющим поле микропрограммы (слова микрокоманды). Среди 8 вариантов выбора, обеспечивающих 8 вариантов, код 000 оставляет содержимое ячейки R [I] неизменным. Таблица соответствия остальных кодов управления пересылкой информации в ячейку R [I] содержит список микроопераций

001 R[I] := R[$I+3$];
010 R[I] := Σ;
011 R[I] := S;
100 R[I] := R[I] $\vee$ S $\vee$ Σ;
101 R[I] := S $\vee$ Σ;
110 R[I] := R[I] $\vee$ S;
111 R[I] := R[I] $\vee$ Σ;

На входы R [$I - 1$], R [$I - 2$], M [I] и L могут подаваться сигналы только от одного источника: R [$I - 1$]:= Σ; L := Π (перенос при сложении); R [$I - 2$]:= Σ, M [I] := S.

Включение любого из указанных способов присвоения в алгоритмы, моделирующие работу процессора микроЭВМ при выполнении конкретной микрокоманды, определяется наличием единиц в соответствующих полях строки кода микрокоманды. Последовательность указания этих присвоений в алгоритме, описывающем работу процессора, имеет существеиное значение. В реальной микроЭВМ все операции, указанные кодом текущей микрокоманды, выполняются параллельно в течение одного элементарного такта. Поэтому при описании работы процессора необходимо, чтобы в правых частях операторов присвоений были записаны правильные значения сигналов источников информации. Все значения сигналов от R, M, ST, S, S1 и L используются прежние, а Σ и Π — новые. Это правило соблюдается, если последовательность присвоений записана в том порядке, в каком описываются возможные связи в процессоре микроЭВМ.

На вход S регистра могут подаваться сигналы сумматора Σ и S1 регистра. Управление этими связями осуществляется 2-битовым полем так, что допускается одновременное использование сигналов от обоих источников. В этом случае три возможных варианта управления моделируются таблицей соответствий

01 S: =S1;
10 S: =Σ;
11 S: =S1 ∨ Σ;

где 01 ∨ 10— код 11. На вход S1 регистра могут быть поданы сигналы сумматора Σ и внешнего входа H. Кодирование 2-битового поля также допускает дизъюнкцию:

01 S1: =Σ;
10 S1: =S1 ∨ H;
11 S1: =(S1 ∨ H) ∨ Σ;

Сигнал внешнего входа H может установить 1 либо в младшем (0001), либо в старшем (1000) разряде, что моделируется логическим сложением (или дизъюнкцией) S1 ∨ H. Сигнал внешнего входа 2-битовый, и наличие единицы в его разрядах не только определяет значение S1, но и устанавливает 1 в ячейке T, инверсное значение содержимого которой $\overline{T}$ используется в качестве источника информации для входа сумматора.

Последнее 2-битовое поле кода микрокоманды управляет работой регистра ST. Внешним по отношению к этому регистру источником информации является сумматор Σ. Первые две команды, изменяющие содержимое ячеек регистра ST, описываются последовательностями операторов присвоения:

1(01). ST[I + 2]:=ST[I + 1]; ST[I + 1]:=ST[I]; ST[I]:=Σ;
2(10). X:=ST[I]; ST[I]:=ST[I + 1]; ST[I + 1]:=ST[I + 2]; ST[I + 2]:=X;

Фактически эти команды отображают сдвиги информации в трех последовательных ячейках регистра ST. Команда 01 «вталкивает» сигнал сумматора в цепочку ячеек, последовательно подвигая Σ → ST [I] → → ST [I + 1] → ST [I + 2], причем прежнее значение ST [I + 2] теряется. Следовательно, эти команды управляют операциями, аналогичными операциям знакомого пользователям ПМК операционного стека. По команде 10 выполняется кольцевой сдвиг содержимого трех ячеек, для моделирования которого приходится использовать вспомогательную величину X как сигнал замыкания кольца.

Логическое сложение на входах ячеек регистра ST обеспечивает команда 3 (11) — трем вспомогательным величинам X, Y и Z следует присвоить значения содержимого ячеек регистра ST, а затем определить их новые значения:

3(11). X:=ST[I]; Y:=ST[I + 1]; Z:=ST[I + 2]; ST[I]:=Σ ∨ Y; ST[I + + 1]:=X ∨ Z; ST[I + 2]:=Y ∨ X.

Полный перечень допустимых связей в процессоре микроЭВМ не очень велик, тем не менее труден для восприятия. Поэтому при описа-

нии микрокоманды целесообразно перейти на более высокий уровень абстракции, заменив полное описание алгоритмов, моделирующих выполнение микрокоманды, символическим описанием ее содержания:

$$<МК> = <\Sigma> \quad <R[I]> <R\ [I-1]><R\ [I-2]>, \\ <M[I]><S><S1><ST>$$

Эта запись означает, что текст микропрограммы состоит из последовательности текстов, описывающих операции пересылки информации в соответствующие ее приемники. Если состояние какого-то приемника остается прежним, то соответствующий участок записи пропускается (можно полагать, что он содержит только пустую, не содержащую символов, строку). Подобная запись удобна еще и тем, что для управляющего кода, реализующего желаемую комбинацию связей, можно пользоваться аналогичной записью, полагая, что символом МК обозначен адрес микрокоманды в РАМК, а символы в треугольных скобках справа от знака равенства соответствуют бинарным полям микропрограммы. В этом случае сохранение прежнего состояния приемника кодируется полем, заполненным нулями.

Следовательно, код $<МК>$ всегда имеет постоянную длину, причем наибольшее число бит содержит поле $<\Sigma> = <\alpha><\beta><\gamma>$, где $<\alpha>$ — поле, указывающее сигналы источников, подключаемых к входу α сумматора, соответствует выбору сигналов R [I], M [I], ST [I], $\overline{R}$ [I], $A \times \overline{L}$, S и поэтому его длина 6 бит; поле $<\beta>$ соответствует выбору сигналов S; $\overline{S}$, S1, 6 и 1, содержит 5 бит; поле $<\gamma>$ соответствует выбору сигналов L, $\overline{L}$, $\overline{T}$ и содержит 3 бит. Следовательно, все поле $<\Sigma>$ имеет длину 14 бит.

При описании работы сумматора использовался символ логического сложения $\vee$, отсутствующий во многих алгоритмических входных языках (например, Бейсик или Фортран), но являющийся общепринятым математическим символом. Так как операция сложения позволяет достаточно просто (с помощью проверки на превышение кода 15) определить результат шестнадцатеричного сложения и переноса, целесообразно при моделировании операции логического сложения составить таблицу с массивом $DIS(X, Y) = X \vee Y$, содержащим готовые результаты этой операции. Значения элементов массива для справки приведены ниже. Заметим, что ПМК «Электроника МК-61» и «Электроника МК-52» выполняют поразрядное логическое сложение как встроенную операцию, реализуемую аппаратурными, а не программными средствами.

Подсчитав длину всех полей микропрограммы, можно записать

$<МК>$, 27 = $<\Sigma>$, 14+$<R[I]>$, 3+$R[I-1]>$, 1 + $<R[I-2]>$, 1+$<L>$, 1+ $<M[I]$, 1+ $<S>$, 2+ $<S1>$, 2+$<ST>$, 2

где после запятой указано число битов соответствующего поля.

Для выбора нужной микропрограммы из ВПМК используется адрес микрокоманды (АМК) длиной 6 бит, обеспечивающий адресацию

64 микропрограмм. Если номер микрокоманды не менее 60, то в зависимости от значения L из ВПМК будут выбираться различные слова и всего ВПМК содержат 68 слов по 27 бит. Если $60 \leqslant$ АМК $\leqslant 63$, то АМК := АМК$+4\times$L. Обозначив индекс массива ВПМК N, сформулируем правило выбора:

если $N \geqslant 60$, то $N := N + 4\times$L; <МК> := ВПМК [N]

Таким образом, для указания последовательности микроприказов, управляющих выбором связей в микроЭВМ, необходимо заполнить массив ВПМК и указать последовательность номеров N.

Возвращаясь к ранее описанным алгоритмам, используем представление микропрограмм микрокомандами. Пусть ВПМК [0] описывает микрокоманду (микрооператор) S := $\overline{\text{R}}$ [I]. Эта микрокоманда отображается микропрограммой, записываемой в виде последовательности полей микрооператоров:

<МК> := <Σ: = $\overline{\text{R}}$ [I]> < >< >< >< >< >< ><S : = = Σ>< >< >,

где < > обозначает пустые строки, кодируемые нулями. Соответствующую микропрограмму в бинарных кодах, учитывающую, что сигнал $\overline{\text{R}}$ [I] требуется подать на вход α сумматора и код управления $\overline{\text{R}}$ [I] записан в последнем поле, можно представить бинарным словом

<МК> : = 000100 00000 000 000 0 0 0 0 10 00 00

Пробелами отделены коды, управляющие передачей сигнала на входы сумматора и остальными связями. Код 10 в поле <S> обеспечивает присвоение <S := Σ>.

Подобным образом для R [I — 1] := S + A$\times\overline{\text{L}}$ составим микропрограмму

<МК> := <Σ := A$\times\overline{\text{L}}$+S>< > <R [$I$ — 1] := Σ>< >...
...< >,

ее код

<МК> := 000010 10000 000 000 1 0 0 0 00 00 00;

микрооперации S := R [I] + S + 1 соответствует микропрограмма

<МК> := <Σ := R [I] + S + (L $\vee$ $\overline{\text{L}}$> < >< >< >< > < >×<S := Σ>< >< >,

ее код

<МК> := 100000 10000 110 000 0 0 0 0 10 00, 00. В последнем из кодов для подачи 1 на вход γ сумматора использована дизъюнкция прямого и инверсного сигналов от ячейки переноса. Для микрокоманды S := R[I] + S + L, отображаемой микропрограммой

<МК> := <Σ := R [I] + S + L> < > < > < > < > < > < >
<S := Σ> < > < >,

код

<МК> = 10000 10000 1000 000 0 0 0 0 10 00 00.

Рассмотренные микропрограммы записаны в ВПМК под номерами 0, 1, 2 и 3, и при указании адреса ВПМК, например [АМК] : ВПМК [0], будет вызвана первая из рассмотренных микрокоманд.

Если записать микропрограммы в виде таблицы, строки которой соответствуют адресам микропрограммы, для первых четырех из них получим

000100	00000	000	000	0	0	0	0	10	00	00
000010	10000	000	000	1	0	0	0	00	00	00
100000	10000	110	000	0	0	0	0	10	00	00
100000	10000	100	000	0	0	0	0	10	00	00
000000	00000	000	000	0	0	0	0	00	00	00

Теперь последовательность микрокоманд может быть описана алгоритмом, состоящим из операторов присвоения, частично охваченных циклом:

I:=1; АМК:=0;
I:=I + 1; АМК:=2;
I:=I + 1; АМК:=1;
для К = 1 шаг 1 до 6 выполнить
(I:=I + 1; АМК:=0; I:=I + 1; АМК:=3; I:=I + 1; АМК:=1)

Если выписать номера тактов, сопоставив их с номерами микрокоманд и их адресами АМК, то получим таблицу

I	0	1	2	3	4	5	6	7	8	9	10	11	12
J	0	1	2	3	4	5	3	4	5	3	4	5	3
АМК	4	0	2	1	0	3	1	0	3	1	0	3	1

I	13	14	15	16	17	18	19	20	21	22	23	24
J	4	5	3	4	5	3	4	5	3	4	5	6
АМК	0	3	1	0	3	1	0	3	1	0	3	1

Это сопоставление показывает, что для выполнения последовательности микрокоманд необходимо указать таблицу их адресов в зависимости от индекса J:

J	0	1	2	3	4	5	6	7	8
АМК	4	0	2	1	0	3	1	4	4

Как видно из таблицы, микропрограмма, считываемая из ВПМК по адресу АМК = 4, не выполняет никаких пересылок; все ее бинарные поля содержат нули.

Последовательность АМК в зависимости от индекса *J* выполняет функции макрокоманды, называемой синхропрограммой, чтобы подчеркнуть значение синхронизирующего генератора, формирующего последовательность номеров *J*.

В рассматриваемых микроЭВМ имеется специальная память синхропрограмм (ПСП), из которой при указании адреса синхропрограммы (номера) (АСП) и номера *J* можно получить АМК. Закономерность работы синхронизатора выбрана так, что за время изменения *J* от 0 до 41 счет трижды начинается с 0. Это позволяет составить последовательность из трех синхропрограмм, исполняемых за время одного периода синхронизатора.

В памяти синхронизатора записаны 128 различных синхропрограмм, и, следовательно, АСП должен быть 7-битовым. Для указания последовательности синхропрограмм необходимо указать три 7-битовых кода на каждый полный период работы синхронизатора. В зависимости от того, на каком из трех мест будет указан адрес синхропрограммы, ее АМК будут появляться на различных тактах *J* в соответствии с правилами определения.

Всю последовательность из трех синхропрограмм также можно назвать синхропрограммой с векторным адресом из трех 7-битовых кодов АСП, составляющих синхропрограммы.

Рассматривая отдельную микрокоманду, составленную из набора простейших микрооператоров (микроприказов), выполняемых за один такт работы микроЭВМ, можно заметить, что последовательность микрокоманд, выполняемых за несколько тактов, позволяет выполнять достаточно сложные операции над числами. Если последовательность синхропрограмм, состоящих из последовательности микрокоманд, реализуемых последовательностями микроприказов (микропрограммой), реализует еще более сложные процедуры, то возникает беспокойство о возможности надежного управления более сложными информационными связями. Именно поэтому оправдан структурированный подход к управлению сложными системами, характеризующийся иерархической архитектурой межуровневых связей. Использование символических обозначений при описании синхропрограмм позволяет программисту, разрабатывающему программное обеспечение вычислительных систем (подобных ПМК) с использованием микроЭВМ рассматриваемого типа, легко понимать смысл микрооперации, выполняемой синхропрограммой. При этом номер ее отображается адресом, который можно рассматривать как кодовое слово, обозначающее эту макрооперацию на языке более высокого уровня. Иерархический подход позволяет программисту отрабатывать элементы программ решения сложных задач на каждом уровне, уточняя слова соответствующими последовательностями слов низшего семантического уровня.

Для составления программы работы микроЭВМ требуются команды не только для обработки числовых данных, но и для изменения, если необходимо, последовательности выполнения команд путем ветвления и

обращения к подпрограммам. В противном случае в памяти для хранения такой программы будет затрачено много места для записи повторяющихся фрагментов программы.

С целью организации управления ветвлениями, обращениями к подпрограммам и нормального выполнения команд высшего уровня в архитектуре рассматриваемых микроЭВМ принят следующий метод. Часть ячеек регистров R и ST отведена для хранения адресов главной программы и вспомогательной информации для организации ветвления.

Память главной программы содержит 256 ячеек для хранения команд ПК, и исполняемая очередная команда считывается из ПК по адресу, хранящемуся в ячейках R (36) и R (39). Шестнаддцатеричный код адреса АК := R (39)×16 + R (36). Это означает, что 8-битовый адрес разбит на две равные части, из которых первая хранится в ячейке R (36), а вторая — в R (39). По этому адресу команда выбирается из массива команд как <К> := ПК [АК]. Каждая команда (3,3) содержит несколько полей, в которых записаны адреса синхропрограмм (АСП), подлежащих выполнению в заданной последовательности тактов:

$$<К> = <АСП><АСП><АСП><МОД>$$

Первый адрес синхрокоманды указывает, какая синхрокоманда выполняется для индексов I от 0 до 26, следующие два — от 27 до 35 и от 36 до 41. Поле <МОД> предназначено для модификации работы синхропрограмм, запрещая изменение содержимого регистра R для индексов I от 0 до 35. В соответствии с принятым размещением в ячейках R (36) и R (39) адресов команды только третье поле <АСП> в команде указывает способ обработки адреса.

Каждая синхропрограмма занимает в памяти синхропрограмм 9 адресов микрокоманд, которые должны выполняться в соответствии с работой счетчика J. В то же время для адреса команд и вспомогательной информации отведено только 6 ячеек с индексами I = 36, 37, 38, 39, 40, 41, что соответствует J = 0,1, 2, 3, 4, 5. Кроме того, специфика операций, выполняемых над адресами, существенно отличается от операций над кодами. В частности, для чисел не применяются операции изменения содержимого только одного разряда, а операции десятичных сложения и вычитания не используются для обработки адресов. Для указания адреса синхропрограмм в первых двух полях команды требуется по 7 бит, а для указания адреса синхропрограммы в последнем поле отводится 8 бит. Если шестнадцатеричное число, записанное в этом поле, больше десятичного 31, то автоматически вырабатывается адрес 95 синхропрограммы, при выполнении которой весь код <АСП> указанного поля передается в ячейки R (38) и R (40) регистра. Таким образом, в памяти команд можно указать 8-битовый параметр, передаваемый в регистр R. В зависимости от дальнейшего использования этот параметр может быть адресом перехода или числовой константой для

вычислений. Всего в памяти синхропрограмм хранится набор из 128 различных синхропрограмм.

МикроЭВМ К145ИК13 содержит все компоненты ЭВМ — оперативную память, АЛУ, синхронизатор и управляющее устройство. Однако на вопрос о системе команд, определяющей архитектуру микроЭВМ рассматриваемого типа, нельзя дать однозначный ответ. С учетом формул (3.1) — (3.5) и способов их реализации можно выделить несколько основных семантических уровней:

<Программа> =<Последовательность команд>; <Команда> = = <Последовательность макрокоманд>; <Макрокоманда> = = <Последовательность составных синхропрограмм>; <Составная синхропрограмма> = <Последовательность синхропрограмм>; <Синхропрограмма> = <Последовательность микрокоманд>; <Микрокоманда> = <Последовательность микроприказов>; <Микроприказ> = <Последовательность состояний электронных логических элементов>.

При этом должно быть обеспечено не только согласование требуемых последовательностей управляющих слов каждого уровня, но и их выполнение во временных циклах. Описываемая микроЭВМ характеризуется гибкой, перестраиваемой архитектурой, перестройка которой осуществляется выбором наборов микрокоманд и синхропрограмм в матричных накопителях. Эта особенность архитектуры позволяет на одном и том же базовом кристалле организовать две существенно различные архитектуры; процессор обработки команд программы пользователя и процессор для вычисления трансцендентных функций и выполнения арифметических операций с плавающей запятой. Процессор логических операций также отличается по архитектуре в соответствии с системой используемых микрокоманд и синхропрограмм.

В заключение описания микроЭВМ К145ИК13 укажем, что на внешние выводы БИС поданы сигналы синхронизатора, имеющие длительность 3 такта, каждый из которых равен 4 элементарным (для обработки одного бита информации) тактам. Эти сигналы обычно используются для опроса клавиатуры и управления мультиплексной индикацией, обеспечивая подсветку для индикации очередных разрядов. С выхода дешифратора шестнадцатеричного кода от одной из ячеек регистра R подаются сигналы управления на матрицу, 8 выходов которой управляют сегментами знакомест индикатора. Так как информация в регистре R непрерывно сдвигается, специальные сигналы управления фиксируют поочередно на трех макротактах в соответствии с длительностью сигналов синхронизатора, подаваемых на знакоместа индикатора, значения кодов из ячеек R (0), R (3), R (6), R (9), R (12), R (15), R (18), R (21), R (24), R (27), R (30), R (33).

Хотя БИС К145ИК13 не имеет внутреннего генератора, определяющего время обработки бита информации, использование сигналов внешнего генератора фазовых сигналов обеспечивает синхронную работу нескольких микроЭВМ.

3.5. ОСОБЕННОСТИ РАБОТЫ МИКРОЭВМ К145ИК18

МикроЭВМ К145ИК18 обладает рядом особенностей по сравнению с К145ИК13, что определяет ее использование в качестве контроллера внешних устройств.

В этой микроЭВМ регистры R и M имеют 36 ячеек, отдельные ячейки RA и RB выделены для счетчика адреса и регистра параметров ветвления. Регистр S содержит одну ячейку, но вместо S1 организован регистр RIN из 3 ячеек, соединенных в кольцо. Добавлены также 6 ячеек ROUT, предназначенных для вывода сигналов на внешние устройства с 24 выводов БИС. Содержимое этих ячеек при приеме информации от сумматора смещается последовательно, если присутствует специальный сигнал, длительность и назначение которого определяется программистом. Регистра ST в рассматриваемой микроЭВМ нет. Таким образом, общая емкость оперативной памяти микроЭВМ К145ИК18 меньше, чем микроЭВМ К145ИК13, но использование кольцевого регистра из 3 ячеек существенно увеличило гибкость адаптации микроЭВМ к временным параметрам внешних устройств.

В микроЭВМ К145ИК18 предусмотрены средства для работы с параллельным кодом. Для приема параллельного кода используются внешние входы регистра RIN (IN от INPUT — вход). Всего таких входов 12, по 4 у каждой ячейки регистра. Так как при работе микроЭВМ на каждом такте содержимое ячеек этого регистра сдвигается по кольцу, возникает возможность заполнять все три ячейки, передавая информацию только через четыре входа, связанные с одной из ячеек. Это обстоятельство напоминает о принятом нами соглашении не указывать передачу информации из ячейки в ячейку на каждом такте, а менять лишь индекс ячейки, к которой подсоединены внешние устройства. Поясним сказанное следующим примером.

Пусть три ячейки RIN(0), RIN(1), RIN(2) обрузуют кольцо и на вход INP0 первой из них на каждом такте поступают последовательно коды шестнадцатеричных чисел A, B и C. Запишем процедуру приема информации полностью, выписывая сдвиги:

```
I=1:  X:=RIN(0); RIN(0):=RIN(1); RIN(1):=
      :=RIN(2); RIN(0):=A; RIN(2):=X;
I=2:  X:=RIN(0); RIN(0):=RIN(1); RIN(1):=
      :=RIN(2); RIN(0):=B; RIN(2):=X;
I=3:  X:=RIN(0); RIN(0):=RIN(1); RIN(1):=
      :=RIN(2); RIN(0):=C; RIN(2):=X.
```

Перемещение информации в ячейках регистра при ненулевом начальном значении отображается следующей таблицей значений:

	RIN [0]	RIN [1]	RIN [2]
Начальное значение	5	6	7
Макротакт 1	A	7	5
Макротакт 2	B	5	A
Макротакт 3	C	A	B

Как видно из приведенного примера, начальные значения 5, 6 и 7 через 3 макротакта после полного сдвига по кольцу заменяются значениями входных кодов. Этот процесс можно представить и таблицей присвоения ячейкам значений входных сигналов:

Макротакт 1 ($I = 0$)	RIN [I] := INP0 [I]
Макротакт 2 ($I = 1$)	RIN [I] := INP0 [I]
Макротакт 3 ($I = 2$)	RIN [I] := INP0 [I]

В зависимости от момента времени, когда внешний код записывается в ячейки регистра RIN, может быть изменено взаимное расположение отдельных бит принимаемой информации. Чтобы содержимое в каждой ячейке регистра не искажалось, сигнал записи стробируется в конце каждого такта.

Регистры ROUT предназначены для вывода информации параллельным кодом. Сигналы на выходах изменяются непосредственно в процессе засылки информации с выхода сумматора четыре раза на каждом такте. Однако после присвоения ROUT [K] := Σ содержимое K-й ячейки регистра изменяться не будет. Если специальными программными средствами разрешить запись в ROUT [K] только в течение одного элементарного такта (при котором обрабатывается один бит информации), то в сдвиговый регистр попадет только бит информации из сумматора. В этом случае регистры ROUT [K] для K = 0,1, 2,3,4,5 оказываются хранителями предыдущей информации, последовательно сдвигающими ее на один внешний вывод при каждом обращении.

Подобная возможность обусловлена специальным устройством управления конфигурацией регистров RIN и ROUT. Это устройство состоит из матрицы накопителя памяти конфигураций и регистра адреса этой матрицы. Регистр адреса конфигурации (РК) может принять 4-битовый код из регистра R в такте I : РК := R [I — 2].

Матрица конфигураций (МК) содержит 16 заранее выбранных вариантов включения связей вида

ROUT [K] := Σ, RIN [I] := INP0;; RIN [I + 1]:= INP1;
RIN [I + 2] := INP2.

Конкретная конфигурация определяется как <Конфигурация выходов> = МК [РК]. При последовательном сдвиге РК := R [I — 2] конфигурация выводов изменяется по мере того, как изменяется содержимое РК. Например, если начальное значение РК равно 0 (0000 по двоичной системе), а значение сигнала на выходе R (I — 2) равно 1 (т. е. 0001), то при последовательном сдвиге РК принимает значения: 0000 1000, 0001 000, 0010 00, 11000, 1000 (при сдвиге влево начиная с младших разрядов). Из этого примера видно, что содержимое РК, первоначально равное 0, последовательно станет равным 1, 2, 4 и 8. Если в матрице конфигураций соответствующие этим адресам варианты содержат команды включения ROUT [K] := Σ только для РА = 8,

то в ROUT [K] попадет лишь младший бит сигнала сумматора. Для однозначности в микрокоманде должен быть указан специальный микроприказ запись-считывание. Если этот микроприказ подан после окончательной установки значений РК, то в соответствующий ROUT [K] перейдет полное значение сигнала сумматора и на выходах зафиксируется параллельный код.

При взаимодействии с внешними устройствами роль оперативной памяти может выполняться регистром RIN, состоящим из трех ячеек. Для обработки информации и возможности получения реакции микроЭВМ на изменение входных сигналов за минимальный промежуток времени необходимо иметь доступ к счетчику адреса постоянно, а не на последних тактах периода, как в микроЭВМ К145ИК13. Именно поэтому регистр адреса не включен в общее кольцо ячеек регистра R. Специальный микроприказ SHORT (короткий) в микрокоманде позволяет выполнить переход на новый адрес главной программы. Однако синхронизатор по-прежнему будет отсчитывать номера микрокоманд, и при переходе по адресу в новой команде главной программы, указывая новый адрес синхропрограммы, будет использоваться другая часть синхропрограммы для формирования последовательности микрокоманд.

В простейших случаях в памяти синхропрограмм записываются «короткие» синхропрограммы, в которых трижды повторяется одинаковая последовательность из трех микрокоманд. Тогда независимо от момента исполнения микроприказа SHORT однозначно определяется вид последующей обработки данных.

МикроЭВМ К145ИК18 отличается и особенностями кодирования команд верхнего уровня, имеющих более разнородные поля: адрес синхропрограммы, поле модификации исполнения последовательности микрокоманд, адрес перехода, код условия перехода, или сокращенно $<К> = <АСП> <МОД> <А><КУС>$.

В памяти синхропрограмм управляющего устройства микроЭВМ по одному АСП записываются одновременно две синхропрограммы СП1 и СП2. Поле <АСП> (или команды <К(3)> вместе с полем <МОД>) позволяет осуществить выбор семи различных вариантов: <АСП1> <АСП2>, <АСП1>, <> <> <АСП2>, <АСП2> <АСП1>, <АСП2> <>, <> <АСП1>, <> <>. Как и в микроЭВМ К145ИК13 (К745ИК13), адреса АСП1 и АСП2 записываются по одному АСП, и, следовательно, количество вариантов синхропрограмм соответственно уменьшается. Однако память команд рассматриваемой микроЭВМ содержит лишь 128 команд, а не 256, как в микроЭВМ К745ИК13 (К145ИК13). Поэтому допустимое число синхропрограмм также уменьшено до 32, хотя каждая синхропрограмма содержит удвоенную последовательность из 18, а не 9 микрокоманд, как в микроЭВМ К145ИК13 (К745ИК13).

В заключение рассмотрим содержимое микрокоманд микроЭВМ К145ИК18 (К745ИК18). Команда содержит 5 многобитовых и 7 однобитовых полей. Поле входа сумматора $<\alpha> = <R[I]> <\overline{R}[I]>$,

$<S><M[I]><A\overline{L}>$ обеспечивает соединение микроприказами с уровнем логической 1 входа α сумматора с прямым R [I] и инверсным $\overline{R}$ [I], а также выходами S и M [I] соответствующих регистров и подачу на вход α значения $A \times \overline{L}$. Поле $<\beta> = <S> <\overline{S}> <RIN$ $[I_{mod}\ 3]><6><BK>$ обеспечивает подачу на вход β сумматора сигналов с прямого S и инверсного $\overline{S}$ выходов регистра S, выхода RIN $[I_{mod}\ 3]$ регистра RIN, а также кода натурального числа 6 и сигнала с внешнего входа BK. Поле $<\gamma> = <1> <L> <\overline{T}>$ разрешает подачу на вход γ сумматора кода натурального числа 1, запись переноса в ячейку L результата сложения в сумматоре и инверсного значения ячейки клавиши $\overline{T}$.

При подаче на любой вход сумматора нескольких сигналов одновременно выполняется операция их логического сложения (дизъюнкция):

Микроприказы уровня логической 1 в поле

$$<R\ [I]> = <R\ [I] := R[I]><R\ [I]:=S> <R\ [I] := R\ [I + 3_{mod}\ 36]>$$

обеспечивают соответствующие присвоения значениям R [I], причем при поступлении одновременно нескольких сигналов они логически складываются на входе ячейки R [I] регистра.

Поле

$$<RIN> = <RIN\ [I_{mod}\ 3] := \Sigma> <RIN\ [I_{mod}\ 3] := S>, <RIN[I_{mod}\ 3]:= 0>$$

обеспечивает пересылку в соответствующую ячейку регистра RIN сигнала с выхода сумматора или регистра S или кода числа 0. Если все микроприказы этого поля имеют уровень логического 0, то содержимое регистра RIN не изменяется.

Однобитовые поля предназначены для микроприказов: $<M[I] :=$ $:= S>$, подключающих выходной сигнал регистра S к ячейке M [I]; $<S := \Sigma>$ и $<BK := \Sigma>$, подключающих выход сумматора соответственно ко входу регистра S или внешнему входу BK; $<R[I - I_{mod} \times$ $\times 36] := \Sigma >$, подключающих выход сумматора к ячейке регистра R, предыдущей по отношению к ячейке с индексом I; <SHORT>, <W/R>, обеспечивающего ввод-вывод информации в соответствии с конфигурацией выходов, определяемой как <Конфигурация> := $:= M_{конф}$ [PK], а также выполнение необходимых присвоений, $<ROUT\ [I] := \Sigma >$ сигналов с выхода сумматора при $I = 0, 1, 2$ 3, 4, 5 и $K = 0,1, 2$, а также присвоение $RIN\ [I + K_{mod}\ 3] :=$ $:= INPK$. Последний микроприказ $<PK> := R\ [I - 2_{mod}\ 36]$ обеспечивает при уровне логической 1 пересылку сигнала с выхода ячейки $R\ [I - 2_{mod}\ 36]$ на вход ячейки PK.

Глава 4

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРОГРАММИРУЕМЫХ МИКРОКАЛЬКУЛЯТОРОВ РАСШИРЯЮЩЕГОСЯ РЯДА

4.1. ВХОДНЫЕ ЯЗЫКИ

Программное (математическое) обеспечение ПМК состоит из сменных прикладных программ (программ пользователя), составленных в соответствии с входным языком, и программ операционной системы, постоянно хранимых в ПЗУ и обеспечивающих выполнение операций, предусмотренных директивами и операторами прикладных программ.

Теоретически возможна одноступенчатая интерпретация каждой директивы или оператора прикладной программы отдельной микропрограммой, образованной последовательностью микрокоманд, выполняемых в соответствующие интервалы времени. Однако при таком способе требуется хранение больших последовательностей микрокоманд, многократно повторяющихся в одной и той же и различных микропрограммах, а значит, и ПЗУ большей емкости. Для уменьшения аппаратурных затрат в ПМК последовательного действия использована многоступенчатая интерпретация директив и операторов последовательностями микрокоманд в соответствии с формулами (3.1) — (3.6). Различают несколько архитектурных или семантических уровней — входного языка, команд, синхропрограмм и микрокоманд, соответствующих различным уровням сложности операций. При этом операнды отображаются на всех уровнях одними и теми же двоичными или двоично-десятичными кодами. Однако уровень входного языка соответствует отображению без разбиения на поля и операциям над всем операндом (например, $x := y + x$), уровень синхропрограмм и команд — операциям над основными полями формата операндов в течение соответствующих интервалов времени, а уровень микрокоманд — выполнению простейших операций над содержимым одного или нескольких разрядов определенного поля формата операнда.

Операции над операндами на уровне входного языка описываются символами операндов и операторов входного языка. Символы алфавитов входных языков ПМК вместе с символами директив обозначены на клавиатуре, причем в зависимости от режима работы некоторые используются для ввода как операторов, так и директив. Часть символов алфавита является общей для входных языков ПМК расширяющегося ряда с тем отличием, что символы П, ИП, $\uparrow$, $\overrightarrow{\text{ХУ}}$, arcsin, arccos и arctg на клавиатуре ПМК типа «Электроника Б3-34» заменены в остальных ПМК расширяющегося ряда стандартизованными символами $x \rightarrow$ П, П$\rightarrow x$, В$\uparrow$, $\leftrightarrow$, $\sin^{-1}$, $\cos^{-1}$ и tg^{-1} соответственно.

Словарный запас операторов входных языков ПМК расширяющегося ряда приведен в табл. 4.1. В кодах операторов, приведенных в

Таблица 4.1

Операторы входных языков ПМК расширяющегося ряда

Код	Нажимаемые клавиши	Описание операции
00	0	Набор цифры 0
01	1	Набор цифры 1
...	...	...
09	9	Набор цифры 9
0A	.	Набор запятой
0B	/—/	Изменение знака
0C	ВП	Ввод порядка
0D	Cx	Очистка регистра X
0E	B ↑	Смещение «вверх» операционного стека
0F	F Bx	Вызов предыдущего содержимого регистра X
10	+	Сложение содержимого регистров Y и X
11	—	Вычитание содержимого регистров X из Y
12	×	Умножение содержимого регистров X и Y
13	÷	Деление содержимого регистра Y на X
14	↔	Обмен содержимым регистров *X* и *Y*
15	F 10^x	Вычисление $x=10^x$
16	F e^x	Вычисление $x=e^x$
17	F lg	Вычисление $x=\lg x$
18	F ln	Вычисление $x=\ln x$
19	F $\sin^{-1}$	Вычисление $x=\arcsin x$
1A	F $\cos^{-1}$	Вычисление $x=\arccos x$
1B	F tg^{-1}	Вычисление $x=\operatorname{arctg} x$
1C	F sin	Вычисление $x=\sin x$
1D	F cos	Вычисление $x=\cos x$
1E	F tg	Вычисление $x=\operatorname{tg} x$
20	F π	Вызов числа $\pi=3{,}1415296$
21	F √	Вычисление $x=\sqrt{x}$
22	F x^2	Вычисление $x=x^2$
23	F 1/x	Вычисление $x=1/x$
24	F →	Поворот операционного стека
25	F x^y	Вычисление $x=x^y$
26*	K $\overrightarrow{0'}$	Перевод градусов (часов), минут в дробные градусы (часы)

Продолжение табл. 4.1

Код	Нажимаемые клавиши	Описание операции
2A*	К $0^{\rightarrow}{}'''$	Перевод градусов (часов), минут и секунд в дробные градусы (часы)
30*	К $0^{\leftarrow}{}'''$	Перевод дробных градусов (часов) в градусы (часы), минуты и секунды
31*	К |x|	Определение модуля x
32*	К ЗН	Определение знака x
33*	К $0^{\leftarrow}{}'$	Перевод дробных градусов (часов) в градусы и минуты
34*	К [x]	Определение целой части x
35*	К {x}	Определение дробной части x
36*	К МАХ	Определение большего из x и y
37*	К $\wedge$	Логическое умножение кодов x и y
38*	К $\vee$	Логическое сложение кодов x и y
39*	К $\oplus$	Логическое сложение кодов x и y иад полем модуля 2 (исключающее ИЛИ)
3A*	К ИНВ	Инверсия кода x (логическое НЕ)
3B*	К СЧ	Вызов квазислучайного числа с равномерным распределением в интервале (0, 1)
3m**	К Р	Ввод сегмента из накопителя эмулятора внешнего ЗУ
4n	x $\rightarrow$ П N	Засылка копии x в регистр данных с адресом N
50	С/П	Стоп
51	БП *hd*	Безусловный переход по адресу *hd*
52	В/О	Возврат из подпрограммы
53	ПП *hd*	Переход к подпрограмме по адресу *hd*
54	К НОП	Нет операции (пропуск шага программы)
55**	К ПВ	Обмен содержимым регистров данных
56**	К ОД	Обмен страницами программной памяти
57	F x$\neq$0 *hd*	Переход по адресу *hd*, если условие $x \neq 0$ не соблюдено
58	F L2 *hd*	Переход по адресу *hd*, если P2$\neq$1
59	F x$\geqslant$0 *hd*	Переход по адресу *hd*, если условие $x \geqslant 0$ не соблюдено
5A	F L3 *hd*	Переход по адресу *hd*, если P3$\neq$1
5B	F L1 *hd*	Переход по адресу *hd*, если P1$\neq$1
5C	F x$<$0 *hd*	Переход по адресу *hd*, если условие $x < 0$ не соблюдено
5D	F L0 *hd*	Переход по адресу *hd*, если P0$\neq$1
5E	F x = 0 *hd*	Переход по адресу *hd*, если условие $x = 0$ не соблюдено
6n	П $\rightarrow$ x N	Вызов из памяти копии содержимого регистра N

Окончание табл. 4.1

Код	Нажимаемые клавиши	Описание операции
7n	К x ≠ 0 *N*	Переход по адресу, хранящемуся в регистре *N*, при несоблюдении условия $x \neq 0$
8n	К БП *N*	Безусловиый переход по адресу, хранящемуся в регистре *N*
9n	К x ⩾ 0 *N*	Переход по адресу, хранящемуся в регистре *N*, при несоблюдении условия $x \geqslant 0$
An	К ПП *N*	Переход к подпрограмме по адресу, хранящемуся в регистре *N*
Bn	К x → П *N*	Засылка копии *x* в регистр, адрес которого хранится в регистре *N*
Cn	К x < 0 *N*	Переход по адресу, хранящемуся в регистре *N*, если не соблюдено условие $x < 0$
Dn	К П → x *N*	Вызов копии содержимого регистра данных, адрес которого хранится в регистре *N*
En	К x = 0 *N*	Переход по адресу, хранящемуся в регистре *N*, при несоблюдении условия $x = 0$

Примечание. Звездочкой отмечены коды операторов, отсутствующих во входном языке ПМК семейства «Электроника МК-54», двумя звездочками — коды операторов, имеющихся во входных языках малосерийных аналогов ПМК семейства «Электроника МК-52».

табл. 4.1, буквой *N* обозначен код адреса регистра данных в операторах прямого или косвенного обращения к памяти, *hd* — двузначный код адресов перехода с набором в общем случае шестнадцатеричной цифры *h* (в ПМК семейства «Электроника МК-52» и их малосерийных аналогах с памятью емкостью 105 ячеек) и десятичной цифры *d*. В ПМК с емкостью программной памяти более 98 слов для прямого перехода к операторам с адресами А0, А1, А2, А3 и А4 достаточно при наборе адреса нажать клавишу, под которой обозначен символ *a* адреса А, и клавишу входа следующей цифры (например, для безусловного перехода по адресу 101 следует нажать клавиши БП А1).

В режиме программирования в разрядах порядка высвечивается содержимое регистра счетчика шагов или программных слов (РС), равное адресу программного слова, вводимого следующим, в разрядах мантиссы — три кода шагов программы, введенных последними. При этом шестнадцатеричные цифры А, В и D высвечиваются соответственно как -, L и Г, а цифра F не высвечивается и используется для отображения пробела. Поэтому, например, код DD оператора К П → x D высвечивается как ГГ, а код 0F оператора F Bx высвечивается как 0.

В обозначениях нажимаемых клавиш после символов префиксных клавиш К и F указаны символы, обозначаемые соответственно или слева или справа или над клавишами (в ПМК семейства «Электроника МК-54» символ НОП обозначен под клавишей), а при вводе адресов ре-

гистров памяти с шестнадцатеричными цифрами нажимают клавиши, под которыми (или для цифры Е, справа от которой) обозначены соответствующие буквы.

Операторы входного языка по назначению подразделяются на несколько групп.

К *операторам набора чисел* относятся операторы набора десятичных цифр от 0 до 9 (обычно называемых литералами), десятичного разделительного знака (точки или запятой), изменения знака /—/, ввода порядка ВП, а также Fπ для вызова числа $\pi = 3{,}1415296$ и СЧ для вызова квазислучайного числа с равномерным распределением в интервале (0,1). В связи с выбранным алгоритмом формирования квазислучайных чисел при некоторых сочетаниях операторов искомая последовательность таких чисел не образуется и приходится переставлять операторы или вводить дополнительные операторы после оператора СЧ.

Синтаксическая группа включает операторы В↑, ↔, F Вх для изменения последовательности операндов, а также оператор К НОП для пропуска шага программы и Сх для стирания содержимого регистра Х. Необходимость в операторе Сх связана с тем, что при очистке регистра Х вводом цифры 0 содержимое операционного стека смещается «вверх», тогда как при вводе оператора Сх в регистр Х заносится цифра 0 без изменения содержимого остальных регистров, что необходимо, например, при ошибочном наборе числа для его исправления без изменения содержимого остальных регистров операционного стека.

Группа операторов обращения к памяти данных включает операторы х→П *N* для засылки копии содержимого регистра Х в регистр памяти с номером (адресом) *N* и П→х *N* для вызова в регистр Х копии содержимого регистра данных с адресом *N*. Кроме того, к этой группе относятся операторы косвенного обращения к памяти К х→П *N* для засылки копии содержимого регистра Х в регистр памяти данных, адрес которого равен модифицированному содержимому адресного регистра *N*, и К П→х *N* для вызоза в регистр Х копии содержимого регистра памяти, модифицированный адрес которого хранится в регистре *N*. При каждом выполнении оператора косвенной адресации содержимое адресного регистра модифицируется — отбрасывается его дробная часть при целой ненулевой части, а целая часть уменьшается на единицу, если адрес $N \leqslant 3$, увеличивается на единицу, если адрес $3 < N \leqslant 6$, и ие изменяется, если $N > 6$.

Группа функциональных операторов, предназначенных для выполнения вычислений, содержит одноместные операторы, выполняемые над содержимым регистра Х, и двухместные операторы, выполняемые над содержимым регистров Х и Y. К двухместиым относятся операторы арифметические +, —, × и ÷, оператор вычисления степенной функции общего вида F x^y, а в ПМК семейства «Электроника МК-52» и их малосерийных аналогах к ним также относятся операторы К МАХ для засылки в регистр

X большего из операндов, хранящихся в регистре X и Y, а также логические операторы К $\wedge$, К $\vee$, К $\oplus$.

Логические операции выполняются над шестнадцатеричными кодами двоичных операндов, содержащих до 28 разрядов. Эти коды формируются набором любой значащей (отличающейся от нуля) цифры и следующими за ней семью или менее шестнадцатеричными цифрами, каждая из которых отображает содержимое тетрады разрядов, отсчитываемых от младшего разряда операнда. Для формирования кодов, содержащих цифры A, B, C, D, E и F, соответствующие двоичные операнды представляют логическими слагаемыми, каждая из которых кодируется только десятичными цифрами, а затем выполняют над этими кодами операцию логического сложения вводом оператора К $\vee$. Код результата логической операции содержит цифру 8, отделенную от следующих в общем случае шестнадцатеричных цифр, отображающих содержимое тетрад двоичного результата операции.

Пусть например, требуется найти логическую функцию $\overline{b}_1 \wedge b_2$ двух двоичных операндов $\overline{b}_1 = 10001010110011110001001 = 1000\ 0101\ 0110\ 0111\ 1000\ 1001$ и $b_2 = 10101011110011011111101111 = 1010\ 1011\ 1100\ 1101\ 1110\ 1111$. Отобразив в регистре X первый операнд кодом 1456789 и введя оператор КИНВ, получим $\overline{B}_1 = 8{,}BA9876$ (высвечивается 8,L —9876), что соответствует $\overline{b}_1 = 1011\ 1010\ 1001\ 1000\ 0111\ 0110$. Представим b_2 слагаемыми $b_2' = 1000\ 1000\ 1000\ 1000\ 1000\ 1000$ и $b_2'' = 0010\ 0111\ 0100\ 0101\ 0110\ 0111$ и отобразим их кодами $B_2' = 2888888$ и $B_2'' = 2234567$. Введем эти коды в регистры X и Y, выполнив операцию логического сложения вводом оператора $K\vee$, получим требуемое значение кода второго операнда $B_2 = 8{,}ABCDEF$ (высвечивается 9, — LCFE). Выполнив над кодами $\overline{B}_1$ и B_2 операцию логического умножения вводом оператора $K\wedge$, получим результат $\overline{B}_1 \wedge B_2 = 8{,}AA8866$ (высвечивается 8, — —8866) или $\overline{b}_1 \wedge b_2 = 1010\ 1010\ 1000\ 1000\ 0110\ 0110$.

Все рассмотренные операторы исполняются как при их вводе нажатием клавиш в рабочем режиме, так и при считывании из программной памяти в режиме автоматических вычислений. Кроме них в программах автоматических вычислений, предварительно вводимых в область программ оперативной памяти, используется *группа операторов управления* выполнением программы: С/П, записываемый в местах останова программы, В/О, записываемый в конце подпрограмм, и операторы косвеиного безусловного перехода общего вида КБП N и обращения к подпрограмме КПП N, обеспечивающие безусловный переход по адресу, равному модифицированному содержимому регистра N памяти. Модификация выполняется так же, как и в операторах косвенного обращения к памяти данных. К этой группе относятся и операторы косвенного условного перехода Кx$\geqslant 0\ N$, Кx $= 0\ N$, Кx$< 0\ N$, Кx$\neq 0N$ с переходом при несоблюдении проверяемого условия по адресу, равному модифицированному содержимому регистра N памяти.

Все эти операторы, как и операторы других рассмотренных групп, занимают по одной ячейке программной памяти независимо от числа нажимаемых клавиш.

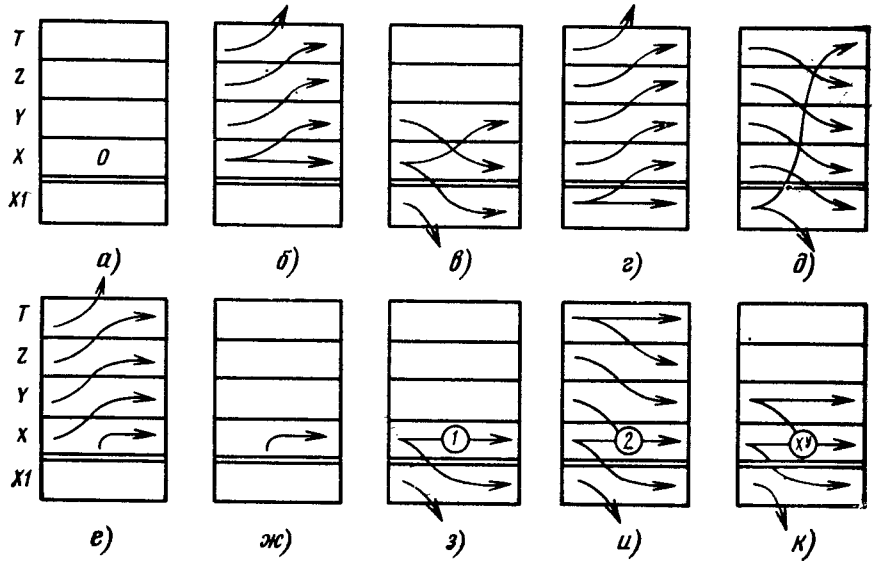


Рис. 4.1

К группе операторов управления выполнением программы относятся операторы, содержащие два программных слова: БП *hd* — безусловного перехода по адресу *hd*, ПП *hd* — обращения к подпрограмме по адресу *hd*, условного перехода $x < 0$ *hd*, $x = 0$ *hd*, $x > 0$ *hd* и $x \neq 0$, *hd* с переходом по адресу *hd* при невыполнении проверяемого условия, а также операторы цикла вида L*N hd* с переходом по адресу *hd*, если модифицированное содержимое регистра *N* больше 1.

Синтаксические правила входных языков ПМК расширяющегося ряда в основном связаны со следующими правилами работы операционного стека:

1. При вводе оператора Cx содержимое регистра X стирается (заменяется цифрой 0) без изменения содержимого остальных регистров операционного стека (рис. 4.1, *а*).

2. При вводе оператора B ↑ операционный стек смещается «вверх» (рис. 4.1, *б*), что может быть описано как PT: = PZ; PZ: = PY; PY: = PX.

3. При вводе оператора ↔ содержимое регистра Y засылается в регистр X, прежнее содержимое которого засылается в регистры Y и X1 (рис. 4.1, *в*) или PX1 := PX; PX: = PY; PY := PX1.

4. При вводе оператора F Bx содержимое операционного стека смещается «вверх» с засылкой содержимого регистра X1 в регистр X (рис. 4.1, *г*): PT := PZ; PZ := PY; PY := PX; PX = PX1.

5. При вводе оператора F → содержимое операционного стека «поворачивается» по часовой стрелке со смещением «вниз» и засылкой предыдущего содержимого регистра X в регистры X1 и T (рис. 4.1, *д*)

согласно цепочке присвоения: PX1:= PX; PX :=PY; PY := PZ; PZ :=PT; PT := PX1.

6. При наборе числа после любого оператора, кроме B↑ и CX, а также при вызове в регистр X копии содержимого регистра памяти содержимое стека смещается «вверх», а вводимый операнд засылается в регистр X (рис. 4.1, *е*).

7. При наборе числа после операторов B↑ или CX оно заносится в регистр X без изменения содержимого остальных регистров операционного стека (рис. 4.1, *ж*).

8. При вводе одноместного функционального оператора результат операции над содержимым регистра X заносится в этот же регистр, а его прежнее содержимое засылается в регистр X1 без изменения содержимого остальных регистров стека (рис. 4.1, *з*).

9. При вводе двухместного функционального оператора (кроме x^y) содержимое стека смещается «вниз», результат операции над содержимым регистров Y и X засылается в регистр X, прежнее содержимое которого пересылается в регистр X1 (рис. 4.1, *и*).

10. При вводе двухместного функционального оператора x^y результат операции над содержимым регистров Y и X засылается в регистр X, прежнее содержимое которого пересылается в регистр X1 без изменения содержимого остальных регистров стека (рис. 4.1, *к*).

Лексические правила рассматриваемых входных языков определяются указанным в табл. 4.1 порядком нажатия клавиш при вводе операторов. При вводе операндов запятая (точка) фиксируется только после набора цифры, отрицательные знаки мантиссы и порядка вводятся оператором /—/ после набора соответствующих цифр. При этом оператор ВП для ввода числа в показательной форме должен вводиться после набора мантиссы, в противном случае может нарушиться согласование во времени выполняемых ПМК операций. При вводе оператора ВП для нулевого значения мантиссы устанавливается ее значение, равное единице.

Следует добавить, что коды некоторых операторов можно вводить нажатием различных клавиш. Это обстоятельство можно использовать при составлении прикладных программ. В частности, при вводе операторов обращения к памяти и операторов косвенной адресации вместо набора адреса регистра памяти 0,1, 2, 3 или 4 клавиши набора цифр можно нажать соответственно клавишу +, —, ×, ÷, или ↔. В ПМК семейства «Электроника МК-54» при вводе этих операторов вместо клавиши 0 можно нажать клавишу B↑, в этом случае в коде операторов вместо цифры 0 будет находиться цифра E, а содержимое регистра 0 при выполнении операторов косвенной адресации не будет уменьшаться на единицу.

Входные языки ПМК предназначены для представления алгоритмов решения задач, которые описываются на естественных языках, но сказуемым и подлежащим в предложениях на естественных языках в программах на входных языках соответствуют операнды и операторы. В рассматриваемых входных языках операнды отображаются опе-

раторами набора чисел или вызова их из памяти. Следовательно, предложение на входном языке образовано по допустимым синтаксическим правилам последовательностью операторов, приводящих к определенному результату. Одно или несколько предложений, приводящих к искомому результату решения задачи, составляет программу на входном языке.

Основной текст программы оканчивается оператором С/П, но этот оператор может использоваться и несколько раз для вывода промежуточных результатов. Подпрограмма записывается после операторов С/П, БП *hd* или В/О и оканчивается оператором В/О. В ПМК расширяющегося ряда обеспечивается до 5 обращений из подпрограммы к подпрограмме, называемых вложениями подпрограмм, так как адресный счетчик содержит 5 регистров (кроме счетчика программных слов).

4.2. ОПЕРАЦИОННАЯ СИСТЕМА

Операционная система ПМК расширяющегося ряда отличается простотой, несмотря на небольшой объем она обеспечивает ввод и вывод информации, работу в различных режимах, интерпретацию и редактирование программы, реакцию на ошибочные ситуации, а также взаимодействие отдельных микроЭВМ. Выбор операционной системы определялся рациональным использованием ограниченных ресурсов программного обеспечения ПМК последовательного действия.

В состав операционной системы входят мониторная программа, драйверы пульта управления и индикаторного устройства (с программой формата выводов) и обрабатывающие программы связи между микроЭВМ вычислительной системы, анализа ошибочных ситуаций, лексического разбора кодов входных символов, редактирования и интерпретации кодов операторов.

Мониторная программа обеспечивает пользователю возможность управления работой ПМК. При включении напряжения питания программа вводит в определенную область оперативной памяти код «метка», являющийся началом координат в системе временно́й адресации, и очищает оперативную и сверхоперативную память, включая флаги исполнения ФИ := 0 и режима ФР := 0, что соответствует режиму исполнения (автоматическая работа). В этом режиме ПМК может работать как непрограммируемый с непосредственным исполнением операторов входного языка, вводимых нажатием клавиш, а также выполнять прикладную программу, хранящуюся в программной памяти, по директиве С/П (Пуск) согласно описанию ОП := Мр (РС), ФИ := F, где РС — текущее состояние счетчика шагов. Директива исполняется при отпускании клавиши. Кроме того, в режиме исполнения пользователь может воспользоваться директивой ПП (Пошаговое прохождение) для исполнения программы по шагам при ее отладке согласно описанию ОП := Мр (РС), РС := РС + 1.

При вводе директивы F ПРГ (Программирование) устанавливается флаг ФР := F, и коды операторов, набираемые нажатием клавиш,

засылаются в программную память, а программа-редактор и драйвер индикаторного устройства обеспечивают высвечивание содержимого регистра счетчика шагов (в разрядах порядка) и шестнадцатеричных кодов трех операторов, введенных последними.

При вводе операторов нажатием клавиш в режиме исполнения или программирования драйвер пульта управления формирует коды символов в соответствии с поступающими с клавиатуры при нажатии клавиш разрядными сигналами Д2,..., Д11, подаваемыми на вертикальные шины клавиатуры (см. рис. 2.18) с выхода микроЭВМ К745ИК1302, и коммутируемыми на входы К1 и К2 этой микроЭВМ с горизонтальных шин клавиатуры К1, К2 и К3 (К1, К2). Соответствие символов на нажимаемых клавишах и формируемых драйвером кодов отображено в табл. 4.2. Организация матричного поля клавиатуры, от которой зависит компактность и эффективность фрагментов программного обеспечения операционной системы, связана с разработкой драйвера пульта управления, программы лексического разбора и интерпретатора входного языка.

Таблица 4.2

Соответствие символов нажимаемых клавиш и кодов

Вход / Выход	К1		К2		К1, К2	
	Символ	Код	Символ	Код	Символ	Код
Д2	0	80	$\frac{1}{2}$	10	С/П	90
Д3	1	81	—	11	БП	91
Д4	2	82	×	12	В/О	92
Д5	3	83	÷	13	ПП	93
Д6	4	84	↔	14	x → П	94
Д7	5	85	·	15	$\overrightarrow{\text{ШГ}}$	95
Д8	6	86	/—/	16	П → x	96
Д9	7	87	ВП	17	$\overleftarrow{\text{ШГ}}$	97
Д10	8	88	Сх	18	К	98
Д11	9	89	В↑	19	F	99

Программа лексического разбора формирует коды операторов из последовательности кодов символов, формируемых драйвером пульта управления, причем преобразования выполняются в шестнадцатеричной системе счисления. Программой используются ячейки текущего ТК (1 : 0) и предыдущего ПК (1:0) кодов клавиши, причем драйвер пульта управления формирует коды в ячейке ТК и перед вводом оператора ПК := 00. Если код получен по входу К1, то ТК := ТК ∧ OF. Коды символов по входу К2 обрабатываются согласно описанию: если ТК (0) ⩾ 5, то ТК := ТК (0) + 05, иначе ТК := ТК (0) + 10.

Так, при нажатии клавиши 1 на вход К1 поступает двоичный код 10000001 символа 1 и результат обработки 10000001 ∧ 00001111 = 0000 0001, или ТК:=01. При нажатии клавиши ÷ на вход К2 поступает код 00010011, и результат обработки 00000011 + 00010000 = 00010011, или ТК:=13, тогда как при нажатии клавиши Cx результат обработки 00001000 + 00000101 = 00001101, или ТК:=0D (в режиме программирования высвечивается как 0Г). При поступлении кодов символов на входы К1, К2 ТК:=ТК(0) + 50, причем для кода символа К выполняется ПК:=ПК + F0 и происходит обращение к драйверу пульта управления для обработки следующего символа. Если введен символ F, то для последующих символов обработки выполняется ТК:=ТК(0) + 15 по входу К1 и ТК:=ТК(0)+40 по входам К1, К2, а по входу К2 выполняется следующая процедура:

если ТК(0) < 6, то ТК:=ТК(0) + 20, если ТК(0) = 9, то ТК:=ТК(0) + 06, иначе выполняется переход на декодирование директив F CF, F ABT или F ПРГ.

Для содержимого тетрады ТК(1) ⩾ 40 выполняется следующая процедура:

если ПК(1) ⩾ 5, то (ТК:=ТК + 17, ПК(1):=1, ПК(0):=ТК(0)), иначе (если ТК(1) < 5, то (ТК:=ТК + 17, ПК:=20), иначе (если ТК(0) ∨ A = B, то ПК:=20, если ТК(0) ∨ A = E, то (ПК(1):=1, ПК(0):=ТК(0)), если ТК(0) ∨ A = = F, то декодируются директивы $\overrightarrow{\text{ШГ}}$ или $\overleftarrow{\text{ШГ}}$)).

После выполнения этой процедуры происходит обращение к драйверу пульта управления. Если ТК(1) < 4, то выполняется процедура:

если ПК(1) > 2, то, если ТК(1) ≠ 0, то ТК:=ТК + 16, иначе, если ТК(0) ⩾ ⩾ 3, то ТК:=ТК + 3D, иначе ТК:=ТК + 54, если ПК(1) = 2, то ПК:=10 ∨ ∨ ТК(0), если ПК(1) = 1, то ТК(1):=ПК(0), ОП:=ТК.

Основная часть программы лексического разбора хранится в микроЭВМ К745ИК1302, размерность аргументов тригонометрических и значений обратных тригонометрических функций определяется в микроЭВМ К745ИК1303, а в микроЭВМ К745ИК1801 реализуются директивы A↑ и ⇵ для засылки адреса обращения к ВЗУ и начала обмена в соответствии с положениями переключателей Д-П и С—З—СЧ.

Драйвер индикаторного устройства (ДИУ) управляет отображением информации на индикаторе, соответствие знакомест которого тетрадам регистров данных и разрядным сигналам Д2, ..., Д13 показано на рис. 2.8, *а*, а соответствие сегментных сигналов I1, ..., I8 сегментам знакомест — на рис. 2.8, *б*. При обращении к ДИУ на индикаторе выводится содержимое регистра индикации (РИ) в соответствии с сегментными сигналами I1, ..., I7, формируемыми таблицей символов (ТС), хранящейся в ПЗУ микроЭВМ К745ИК1302. Сегментный сигнал I8 для высвечивания запятой определяется содержимым ячейки L регистра состояний этой же микроЭВМ. Соответствие содержимого i-го знакоместа ДИУ (i) и тетрады регистра индикации (РИ) определяется как ДИУ (i + 2) := ТС (РИ (i)).

В состоянии инженерного калькулятора вводимое число отображается на индикаторе последовательностью выбираемых знаков, но в регистре X это число представляется в нормализованной показательной форме. При выполнении любых других операторов содержимое РИ совпадает с содержимым регистра X. На время выполнения оператора индикатор отключается, а в процессе заполнения РИ используются ячейки: положения запятой (ПЗ), счетчика символов (СС), про-

межуточного хранения (ПХ) и порядкового номера вводимого десятичного знака (П).

При выводе содержимого регистра X после выполнения операции содержимое ячейки П := 0 и выполняется следующая процедура:

РИ:=РХ; СС:=7; РИ(11):=F; ПЗ:=0;
если РХ(11) = 9, то РИ(11):=А, РИ(10:9):= — РХ(10:9), иначе, если 0⩽⩽РХ(11:9)<8, то ПЗ:=РХ(9), РИ(11:9):=FFF;
М: если СС + П + 1 ≠ 0, то ПХ:=РХ(0); для $i = 0$ до 7 выполнить (РХ(i):=РХ(i + 1), РИ(i):=РИ(i + 1)), РХ(8):=ПХ; СС:=СС + 1;
если СС + ПЗ + 1 ≠ 0, то если РХ(8) + F ⩽ F, то РИ(8):=РХ(8) + F; идти на метку М; иначе для $i = 0$ до 7 выполнить (РХ(i):=РХ(i + 1), РИ(i):=:=РИ(i + 1), РХ(8):=ОП∧0F; СС:=СС + 1;
если СС ≠ 0, то ПХ:=РХ(0); для i=0 до 7 выполнять (РХ(i):=РХ(i+1), РИ(i):=РИ(i + 1); РХ(8):=ПХ, СС:=СС + 1); ПЗ:=ПЗ + 8;
если РИ(8) + F > F, то РИ(8):=РИ(8 + 1), иначе РИ(8):=F.

Таким образом, числа с порядком от 0 до 8 отображаются в естественной форме с запятой, определяемой содержимым ПЗ, изменяющимся от 8 до F. При последовательном выводе содержимого РИ устанавливается ПХ:=ПЗ, а затем для каждого вводимого знака ПХ:=ПХ+1 в соответствии с содержимым ячейки L. Переполнение ячейки ПХ в одном из разрядов фиксируется в регистре состояний, и по сигналу I8 на индикаторе отображается сегмент запятой. Числа, порядок которых находится вне указанного интервала, отображаются в показательной форме при ПЗ: = 8, что соответствует отображению запятой после первой цифры мантиссы. Незначащие нули в соответствии с процедурой вывода замещаются в РИ кодом F. Вызов в регистр X индекса или адреса первого ранга ($0 \leqslant x < 10^8$) отображается на индикаторе как целое с запятой, фиксированной в младшем разряде ДИУ(2) и нулями в незанятых старших разрядах.

В режиме программирования код оператора засылается в программную память Мр (РС) := ОП по текущему состоянию счетчика РС:= РС+1. Это значение и коды трех введенных последними операторов высвечиваются на индикаторе при заполнении РИ по следующей процедуре:

для $i = 0$ до 4 выполнить SPC(5 — i):=SPC(4 — i);
РИ(8):=F; РС:=РС—3;
пока SPC(1) ≠ SPC(0), выполнить для $i = 0$ до 10; РИ(i):=РИ(i + 1);
если РС > С0, то РИ(11:9):=FFF, иначе (для $i = 0$ до 10 выполнить (РИ(i):=РИ(i + 1)); РИ(9):=F; РИ(11:10):=Мр(РС)), РС:=РС + 1;
для $i = 0$ до 10 выполнить РИ(i):=РИ(i + 1));
для $i = 0$ до 5 выполнить SPC(i):=SPC(i + 1); РИ(11:10):=РС;
для $i = 0$ до 11 выполнить РИ(i):=((i + 1)mod12).

При реализации этой процедуры первый сдвиг содержимого РИ совпадает с анализом РС > С0, а третий — с проверкой SPC(1) ≠ ≠ SPC (0). В последнем сдвиге $i = 11$, РИ (11):= РИ (0) или РИ (11) := F (код пробела, который был занесен в РИ (8)). После выполнения процедуры

РИ(11):=F, РИ(10:9):=РС, РИ(8) = F, РИ(7:6):=Мр(РС—1), РИ(5):=F, РИ(4:3):=Мр(РС—2), РИ(2):=F, РИ(1:0):=Мр(РС—3).

Условие РС > СО выполняется и в нижней области допустимых значений РС — при переходе в режим программирования РИ (11) := := F, РИ (10:9) := 00, РИ (8:0) = FFFFFFFFF.

Ввод директив $\overrightarrow{\text{ШГ}}$ или $\overleftarrow{\text{ШГ}}$ изменяет на единицу содержимое РС и соответственно РИ. Директива FCF обеспечивает возможность исправления ошибочного ввода символов F, К, П → х, х → П и сочетаний К х < 0, К х = 0, К х ≠ 0, К х ⩾ 0, К П→ х, К х→ П, К БП и К ПП. Если повторно вводимый оператор начинается с символа F или К, то исправление выполняется без ввода F CF. Аналогично вводятся исправленные операторы, начинающиеся символами F, К, П → х, х →П, БП, ПП, В/О, С/П, после ошибочно набранных префиксов, начинающихся с символов К, П → х или х → П. Например, вместо ошибочно набранного префикса К достаточно дважды ввести В/О.

Ввод директивы F CF в режиме исполнения обеспечивает устранение последствий ошибочно набранных символов F, К, П → х, х → П. БП.

При автоматическом исполнении программы (ФИ :=F) выполняются присвоения ОП := Мр (РС), РС := РС + 1. В этом случае останов (ФИ: = 0) возможен при считывании оператора С/П, прерывании программой анализа ошибочных ситуаций и при нажатии любой клавиши на пульте управления.

Программа анализа ошибочных ситуаций информирует пользователя об ошибках, возникающих в процессе выполнения прикладной программы. В случае «фатальной» ошибки, как правило, на дисплей выводится сообщение ЕГГОГ и происходит прерывание выполнения прикладной программы. В некоторых случаях выполнение оператора, приводящего к переполнению, блокируется или он выполняется без сообщения об ошибке и прерывании. «Фатальная» ошибка диагностируется при переполнении регистра X, попытке вычислить функции вне пределов ее определения (÷ и 1/х при $x = 0$, $\sqrt{}$ при $x < 0$, 10^x при $|x| > 100$, e^x при $|x| > 100$ ln 10, ln, lg и x^y при $x < 0$, sin, cos и tg при $|x| > 10^{10}$, $\sin^{-1}$, $\cos^{-1}$ при $|x| > 1$), а также при попытке выполнить операторы, коды которых зарезервированы для ПМК расширяющегося ряда (например, К $\sqrt{}$ или К 1/х), за исключением операторов К ПВ и К 0Д (коды которых могут быть набраны нажатием клавиш К 1 и К 2), при использовании которых в ПМК, не имеющем памяти программ-данных, пропускается один шаг программы, как и при выполнении оператора К НОП. В этих случаях микроЭВМ К745ИК1303 не посылает в канал связи (КС) сообщение о выполнении оператора, а программа анализа ошибочных ситуаций проверяет содержимое внутреннего таймера и при его переполнении очищает канал связи (КС := := =FF), заполняя РИ строкой символов FFFFEDD0DFFF, выводимой на индикатор, и прекращая выполнение программы.

Использование внутреннего таймера для контроля за сеансом связи позволяет идентифицировать отсутствие адресата при передаче резервного кода оператора или ошибку в случае неверного задания ар-

гумента функций, а также исключить изменение программного обеспечения, хранимого микроЭВМ, при изменении конфигурации вычислительной системы.

При занесении операнда в регистр X операторами-литералами мантисса ограничивается восемью разрядами, после чего выполнение следующих операторов набора знаков блокируется без высвечивания сообщения ЕГГОГ. Результат арифметических операций и операторов x^2 и $1/x$, попадающий в область машинного нуля, отображается присвоением РХ := 0. В режиме автоматических вычислений при выполнении этих же операторов программы промежуточные значения могут превышать допустимые, достигая 10^{900}. При вызове таких чисел из регистров данных или выводе на индикатор, если их порядок от 100 до 800, диагностируется «фатальная» ошибка и высвечивается сообщение ЕГГОГ, а числа с порядком от 800 до 900 замещаются нулем. У большой серии ПМК «фатальная» ошибка диагностируется, если порядок чисел от 100 до 200, а вывод числа с порядком от 200 до 800 может привести к нарушению нормальной работы ПМК (для восстановления которой может оказаться необходимым выключение питания). Это следует учитывать при составлении прикладных программ, помещая в сомнительных местах операторы В↑ или /—/, обеспечивающие диагностирование «фатальной» ошибки, или проверяя сомнительные места программы в режиме пошагового выполнения. Например, при автоматическом выполнении программы x^2 x → П9√ С/П с начальным значением $1 \cdot 10^{90}$ = РХ будет получен ошибочный результат РХ = $1 \cdot 10^{10}$ при хранении сообщения в регистре 9. Если же программу ввести как x^2В↑ С/П, то после выполнения оператора x^2 на индикаторе высвечивается сообщение ЕГГОГ. При автоматическом выполнении программы В↑ × 1/x С/П с исходным значением $1 \cdot 10^{90}$ = РХ результат вычислений РХ = 0, хотя при пошаговой проверке после выполнения оператора × будет диагностирована «фатальная» ошибка.

Для определения оператора, приведшего к фатальной ошибке, следует учесть, что код приведшего к переполнению оператора-литерала находится в тетрадах РИ (7:6), в остальных случаях код оператора, вызывающий останов, — в РИ (5:4).

Отдельная программа операционной системы обеспечивает протокол связи (порядок обмена информацией) между микроЭВМ ПМК. В системе временной адресации системной магистрали выделена область, образующая 8-разрядный канал связи для обмена информацией в течение 8 тактов на каждом цикле. Через этот канал связи каждая микроЭВМ имеет доступ ко всей оперативной памяти. При включении питания и формирования временной метки, определяющей начало цикла, в канал связи засылается код КС := FF, отображающий его готовность. Условием доступности канала связи для передачи информации является состояние КС := XF, где X — произвольная шестнадцатеричная цифра.

Рассмотрим работу программы связи между микроЭВМ К745ИК1302, К745ИК1303 и эмулятором накопителя ВЗУ. В первой

микроЭВМ при выполнении операторов БП ЕА, ПП ЕА, $x=0$ ЕА, $x \neq 0$ ЕА, $x \geqslant 0$ ЕА, $x < 0$ ЕА, LN ЕА программа анализа лексики анализирует первый шаг. Для операторов безусловного и условного переходов устанавливается флаг адреса перехода ФАП := F, и при считывании адреса он пересылается в счетчик шагов РС := АП. Если ФАП = 0, то проверяется состояние канала связи: если КС $\wedge$ 0F = = 0F, то канал свободен и управляющая информация, если она передавалась, принята адресатом. Выполнение этого условия разрешает исполнение любого следующего оператора. Если ОП $\leqslant$ 0F или ОП $\geqslant$ $\geqslant$ 40, то управление передается фрагменту программы-интерпретатора этой микроЭВМ, в противном случае выполняется пересылка РХ1:=РХ и код оператора (или составляющее его слово для двухшаговых операторов) передается в канал связи КС:= ОП. После этого проверяется флаг исполнения. Если ФИ= F, то происходит обращение к программной памяти по текущему содержимому РС для считывания очередного слова, в противном случае после выполнения текущего оператора управление передается драйверу индикаторного устройства для вывода содержимого РХ на индикатор.

Если при проверке канала связи КС $\wedge$ F0 $\neq$ 0, то включается внутренний таймер, при его переполнении прерывается связь с микроЭВМ, которой адресовано сообщение. Время работы таймера ограничено значением КС = 0F, после чего операционная система переводится в режим ожидания. Это позволяет выполнять сложные процедуры обработки информации и связи между микроЭВМ без непосредственного контроля мониторной программы. Подобным образом обеспечивается возможность расширения входного языка и операционной системы.

При включении питания микроЭВМ переходит в состояние анализа содержимого канала связи. Если КС $\wedge$ F0 = 10, то выделяется одна группа операторов, если $20 \leqslant$ КС $\leqslant 26$, то другая и управление передается соответствующему фрагменту интерпретатора входного языка этой микроЭВМ. Сообщение о выполнении адресованного ей оператора передается в канал связи сигналом КС := КС $\vee$ 0F до вывода результата в оперативную память. Это обеспечивает завершение сеанса связи и переход к следующему или запуск драйвера устройства вывода для уменьшения затрат времени. К моменту обращения к операционному стеку, драйверу индикаторного устройства или интерпретатору для выполнения очередного оператора содержимое стека будет готово к использованию.

При включении напряжения питания эмулятор накопителя ВЗУ производит загрузку в программную память ПМК сегмента, обращение к которому выполняется оператором СЧ (код 3В). При нескольких эмуляторах приоритет на загрузку отдается программе наибольшей длины, а при равной длине программ — эмулятору накопителя ВЗУ, включенного в системную магистраль последним относительно выхода микроЭВМ К745ИК1302. После загрузки запоминается адрес блока (АМе) эмулятора, устанавливаемый переключателем или пере-

мычкой на входе этой микроЭВМ. Если эмулятор переведен в состояние безадресного обращения, то он реагирует на оператор КР (Ввод сегмента) без адреса. В противном случае ожидается выполнение условия КС:= АМЕ для данного эмулятора. Совпадение адреса запроса с адресом эмулятора подтверждается ответом КС := КС $\vee$ 0F, посылаемым в канал связи. После этого эмулятор переходит к ожиданию кода КС := 3m (m = 0 ... B), появление которого приводит к началу загрузки сегмента, а ее окончание подтверждается сообщением КС := := КС $\vee$ 0F.

Работа ПМК связана с организацией оперативной памяти, образованной оперативными регистрами памяти микроЭВМ К745ИК13 и регистрового ОЗУ К745ИР2 (их емкость 1024 бит, но для согласования во времени используется лишь 1008 бит), включенных в системную магистраль. Оперативная память условно разбита на три блока M1, M2 и M3 (см. рис. 2.12), обращение к которым в каждой микроЭВМ производится соответственно по временным адресам (в интервалы времени E1, E2 и E3. Каждый блок можно рассматривать как массив n регистров емкостью 56 бит каждый или n страниц из 144-битовых слов, распределенных по блокам M1 (n — 1:0), M2 (n — 1:0), M3 (NPG — 1:0), где n = 14 для семейства ПМК «Электроника МК-54» и n = 15 для семейства ПМК «Электроника МК-52».

4.3. ИНТЕРПРЕТАЦИЯ ОПЕРАТОРОВ

Математическое обеспечение интерпретации операторов прикладных программ на язык микрокоманд распределено в ПЗУ различных микроЭВМ. В частности, операторы, коды которых в табл. 4.1 отмечены звездочкой, интерпретируются в микроЭВМ К745ИК1306, большинство функциональных операторов интерпретируются в микроЭВМ К745ИК1303, а операторы набора чисел — в К745ИК1302. Примеры интерпретации некоторых операторов рассмотрены в следующих разделах, здесь ограничимся лишь особенностями реализации операторов.

При вводе операторов набора цифр — литералов (ЛИТ) очищается регистр X и флаг запятой (ФЗ := 0, при вводе запятой ФЗ: =F), после чего реализуется процедура

для i = 1 до 8 выполнить PX(8—i): = ЛИТ; если i > 1, то, если ФЗ:=0, то PX(10 : 9) : = PX (10 : 9) + 1.

Ввод оператора ВП реализуется установкой флага ФП :=F и выполнением процедуры

если PX(7:0):=0, то PX(7):=1, иначе PX(10):=PX(9), PX(9):=ЛИТ.

Оператор /—/ в зависимости от текущей области ввода, определяемой флагом ФП, изменяет знак мантиссы PX (8) := $\overline{(PX\ (8)}$ + + 10) mod 16 или порядка PX (11) := $\overline{(PX\ (11)}$ + 10) mod 16. Положительные мантисса и порядок имеют код знака 0, отрицательные—

код 9. Так как 0 рассматривается как положительное число, то при вводе оператора /—/ для нулевой мантиссы на индикаторе высвечивается — 0, но в регистре X код знака мантиссы не изменяется. Изменение знака порядка управляется оператором ВП и сохраняется до ввода операторов В↑, F Вх, П→ xN или начала ввода нового числа.

Оператор x→ ПN реализуется присвоением исполнительному адресу ЕА значения N, а выполнение оператора К x→ ПN — согласно описанию PN := Мд (N), ЕА := f_D(PN); Мд (ЕА) := РХ, где процедура передачи управления по адресу ЕА (обозначенная f_D) следующая:

если $N \leqslant 3$, то PN:=PN — 1, иначе, если $N \leqslant 6$, то PN:=PN + 1; пока PN(9) < 7, выполнить (для i = 1 до 8 выполнить PN(i — 1):=PN(i); PN(9): =PN(9) + 1; ЕА:=PN(1:0)).

При выполнении операторов косвенной адресации шестнадцатеричный адрес второго ранга N отображается одной или двумя десятичными цифрами. Если порядок равен нулю, то адрес хранится в тетраде PN (7), иначе исполнительный адрес ЕА определяется согласно процедуре

пока PN(9) < 7, выполнить (для i = 1 до 8 (PN(i — 1):=PN(i), PN(9): =PN(9) + 1); ЕА:=PN(1:0)).

Для области программной памяти исполнительный адрес, определяемый этими процедурами, задается в диапазоне от 00 до 99. Для памяти данных исполнительный адрес (адрес первого ранга) ЕА определяется по указанному в операторе адресу второго ранга N согласно описанию

ЕА:=f_D(PN); если PN(1):=0, то ЕА:=PN(0), иначе ЕА:=(А + PN(0)) mod$10_н$

(индекс «н» соответствует шестнадцатеричной записи кодов цифр одной тетрадой двоичных разрядов).

Регистр памяти данных можно использовать для организации стека адресов обращения к программной памяти или памяти данных, содержащего до 4 двухразрядных исполнительных адресов (последний может быть одноразрядным):

PN(9) = 7; EA_1 = PN(1:0); EA_2 = PN(3:2); EA_3 = PN(5:4); EA_4 = = PN(7:6).

После каждого извлечения адреса из стека должен уменьшаться порядок PN (9) := PN (9) — 2, но при $N < 7$ следует учитывать модификацию адреса, которая может затронуть содержимое всего регистра PN.

Оператор К ПВ обмена регистрами данных областей памяти данных Мд и программ-данных Ms обеспечивает обмен начиная с адреса N =0. Число m регистров, участвующих в обмене, зависит от числа страниц оперативной памяти. Для аналогов ПМК семейства «Электроника МК-52» n = 15 и m = 8, для аналогов ПМК семейства «Электроника МК-54» n = 14 и m = 7. В описании этого оператора учитывается регистр ТМ временного хранения:

для N = 0 до n — 1 выполнить ТМ:=Ms(N); Ms(N):=Мд(N); Мд(N):=ТМ.

Оператор К ПД управляет обменом страницами памяти Ms и Mp:

для РС = 0 до $n-1$ выполнить ТМ:=Ms(РС); Ms(РС):=Mp(РС); Mp(РС):= :=ТМ.

Функциональные операторы управляют определенными преобразованиями исходных чисел (аргументов функций) в результаты преобразования, называемые значениями функций. В простейших случаях функциональные преобразования сводятся к изменению по определенным правилам содержимого полей кода числа в регистре X. Так, оператор К|x| во входном языке ПМК семейства «Электроника МК-52» управляет выделением абсолютного значения содержимого регистра X, заканчивающегося засылкой кода цифры F в тетраду знака мантиссы или согласно формату (2.3) присвоению РХ (8) := F. Выполнение оператора выделения К⌊x⌋ целой INT (x) или К{x} дробной FRC (x) части содержимого регистра X сводится к присвоению нулевого значения дробной или целой части числа x, представленного в естественной форме INT (x) := 0 с последующим приведением к показательной форме.

Оператор вывода знака К ЗН содержимого регистра X, аналогичный оператору алгоритмических языков SGN (x), управляет проверкой соответствующих флагов в слове состояния процесса по составному условию

если РХ $<$ 0, то РХ:=$-$1, иначе, если РХ $>$ 0, то РХ:=1.

Выполнение логических операторов КИНВ, К$\wedge$, К$\vee$ и К$\oplus$ сводится к выполнению в АЛУ однокристальной микроЭВМ логических операций над разрядами операндов, соответствующих их кодам, вводимым в регистры Y и X. Использование подобных кодов обеспечивает возможность выполнения логических операций над словами, содержащими до 28 разрядов, тогда как при непосредственном вводе слов, отображаемых десятичными цифрами 0 и 1 (каждая из которых занимает четыре двоичных разряда), логические операции можно выполнять лишь при числе разрядов мантиссы не более 8.

Арифметические операции сложения содержимого регистров Y и X реализуются в сумматоре однокристальной микроЭВМ последовательным выполнением простейших логических операций над разрядами мантисс операндов, уравниваемых по порядку. Выполнение других арифметических операций сводится к сложению с использованием дополнительных логических операций, а любые вычисления сводятся к последовательности выполнения арифметических операций. (По этой причине профессиональные программисты ЭВМ любые расчетные формулы называют арифметическими выражениями.)

Дополнительные логические операции используются и при выполнении ряда операторов преобразования формы представления физических величин. Так, выполнение оператора $\overleftarrow{\text{К0}}'$ для представления дробных градусов или часов в целые градусы (часы) и минуты (в дробной части результата) сводится к выделению целой INT (x) или дробной FRC (x) части содержимого регистра X и выполнению арифмети-

ческих операций с результатом $x := \mathrm{INT}(x) + (\mathrm{FRC}(x) \times 60)/100$. Например, при исходном значении $x = 5{,}525$ реализация этой формулы приводит к значению $x := 5 + (0{,}525 \times 60)/100 = 5{,}315$ или $5°31{,}5'$.

Оператор $\vec{K0'}$ (см. табл. 4.1) выполняет обратное преобразование $x := \mathrm{INT}(x) + (\mathrm{FRC}(x)/60) \times 100$. Например, для $x = 5{,}315$ результатом выполнения этого оператора будет $x := 5 + (0{,}315/60) \times 100 = 5{,}525$.

Оператор $\vec{K0'''}$ управляет преобразованием числа градусов или часов с дробной частью в целое число часов (градусов), минут (два десятичных разряда после запятой) и дробных секунд (остальные числа после запятой). Результат этого преобразования описывается присвоением $x := \mathrm{INT}(x) + \mathrm{INT}(\mathrm{FRC}(x) \times 60)/100 + ((\mathrm{FRC}(\mathrm{FRC}(x) \times 60)/1000$. Например, при $x = 5{,}525$ ввод этого оператора приводит к результату $x := 5 + 0{,}31 + 0{,}003 = 5{,}313$ или, например, $5°31'30''$.

Оператор $\vec{K0'''}$ управляет выполнением обратного преобразования с результатом $x := \mathrm{INT}(x) + (\mathrm{INT}(\mathrm{FRC}(x) \times 100) + \mathrm{FRC}(\mathrm{FRC}(x) \times 100) \times 10/6)/60$. Например, при $x = 5{,}313$ после выполнения этого оператора получим $x := 5 + (31 + 0{,}5)/60 = 5{,}525$.

Функциональные операторы могут реализоваться арифметическими выражениями, содержащими условные переходы. Например, оператор MAX (ПМК семейства «Электроника МК-52») является по существу условным оператором: если $(\mathrm{PY} - \mathrm{PX}) \geqslant 0$, то $\mathrm{PX} := \mathrm{PY}$.

Некоторые простейшие одноместные операторы могут быть точно (до погрешностей округления) вычислены по расчетным формулам. Например, операторы Fx^2 и $\mathrm{F1/x}$ вычисляются по формулам $x := x \times x$ и $x := 1/x$. Однако операторы, управляющие вычислением трансцендентных функций, не могут быть точно представлены формулами с конечным числом арифметических операций, и выполнение таких операторов реализуют численными методами. Простейшим примером может служить оператор $\mathrm{F}\sqrt{}$, при выполнении которого для вычисления функции $y = \sqrt{x}$ используется метод последовательных приближений по итерационному выражению, называемому формулой Герона

$$y_{i+1} = (y_i + x/y_i)/2, \quad i = 0, 1, 2, \ldots,$$

с предварительным выбором начального приближения y_0. Результаты вычислений по этой формуле достаточно быстро сходятся к искомому значению функции $\sqrt{x}$, и после каждой итерации число верных цифр результата y_{i+1} примерно в 2 раза больше числа верных цифр в предыдущем приближении y_i.

При вычислениях на ЭВМ начальное значение y_0 выбирают, представив заданное число произведением $x = 2^m k$, где m — целое число; $1/2 \leqslant k < 1$. В этом случае в качестве начального принимают значение $y_0 = \mathrm{INT}(m/2)$. Однако автоматическое определение y_0 подобным

методом связано с относительно большими затратами времени, и целесообразно выбирать $y_0 = 1$ или $y_0 = x$. Эти варианты равнозначны: в обоих случаях первое приближение $y_1 = (y_0 + x/y_0)/2 = (1 + x)/2$. При таком выборе y_0 число итераций может оказаться бо́льшим, чем при традиционном выборе y_0 по показателю степени множителя 2^m, но суммарные затраты времени в среднем оказываются меньшими. Так, при $x = 10$ и традиционном выборе y_0 принимают $10—2^4\,(10/16) = 2^4 \cdot 0{,}625$ и, следовательно, $y_0 = 2$. В этом случае по формуле Герона очередные приближения $y_1 = 3{,}5$, $y_2 = 3{,}1785714$, $y_3 = 3{,}1623194$, $y_4 = 3{,}1622776$, $y_5 = 3{,}1622776$. Следовательно, результат с 8 верными цифрами мантиссы получен после 4 итераций. При $y_0 = 1$ или $y_0 = x$ и $x = 10$ по формуле Герона получим $y_1 = 5{,}5$, $y_2 = 3{,}659909$, $y_3 = 3{,}196005$, $y_4 = 3{,}1624556$, $y_5 = 3{,}1622776$. Следовательно, в этом случае результат с 8 верными цифрами получен за 5 итераций, но без затрат времени на вычисление y_0.

Следует отметить, что все специальные (включая и элементарные) трансцендентные функции не могут быть точно описаны расчетными формулами с конечным числом операций и для определения их значений используют численные методы. В этом случае обычно используют аппроксимирующие формулы, полученные ограничением числа членов сходящихся степенных рядов, цепных дробей и других представлений специальных функций с бесконечным теоретически числом членов. Число членов аппроксимирующих выражений, обеспечивающее требуемую точность аппроксимации функции, и затраты времени уменьшаются при изменении аргумента в небольшом интервале, обычно вблизи начала его отсчета. Поэтому большое значение имеет приведение диапазона изменения аргумента в относительно небольшой интервал.

Приведение аргумента упрощается, если значения функции характеризуются каким-либо видом симметрии. В частности, диапазон аргумента четных и нечетных функций может быть заменен в 2 раза более узким интервалом в соответствии с соотношением $f(-x) = sf(x)$, где $s = 1$ для четных функций и $s = -1$ для нечетных. В этом случае с учетом множителя s функцию можно вычислять в области положительных значений при изменении аргумента на всей числовой оси: $F(x) = f(x)$ при $x \geqslant 0$ и $F(x) = sf(x)$ при $x < 0$. В общем случае зеркальной или центральной симметрии значений функции относительно значения x аргумент определяется соотношением $f(x - x_0) = sf(x - x_0)$ и область изменения аргумента сужается при вычислениях $F(x) = f(x)$ для $x \geqslant x_0$; $F(x) = sf(2x_0 - x)$ для $x < x_0$, где $s = 1$ при зеркальной симметрии и $s = -1$ при центральной.

Если функция периодическая и ее значения повторяются с периодом T аргумента, то для их вычислений можно представить аргумент формулой $x = kT + r$, $0 \leqslant r < T$, где $k = \mathrm{INT}(x/T)$ — целое число, а $r = x - kT$ — остаток. В этом случае значения функции вычисляют по формуле $f(x) = f(r)$ с приведением диапазона изменения аргумента в интервал, равный периоду изменения функции.

Необходимо учитывать, что при вычислениях функций возникают особенности, связанные как с видом функции (например, при стремлении ее знаменателя к нулю), так и с влиянием погрешностей вычислений при ограниченной разрядности операндов. Эти особенности приводят к попаданию промежуточных значений функции в область машинного нуля или бесконечности и, как следствие, к ошибочным результатам вычисления функции. В этих случаях приходится усложнять математическое обеспечение вычисления функций в ПМК для полного или по крайней мере частичного устранения влияния подобных особенностей.

В ПМК расширяющегося ряда большинство элементарных функций вычисляются через базовые функции по расчетным формулам, а базовые функции аппроксимированы бесконечными цепными дробями вида

$$f(x) = \alpha_0/(\beta_0 + \alpha_1/(\beta_1 + \alpha_2/(\beta_3 + \ldots)),$$

записываемыми условно в форме

$$f(x) = \frac{\alpha_0}{\beta_0} + \frac{\alpha_1}{\beta_1} + \frac{\alpha_2}{\beta_2} + \ldots .$$

В качестве базовых, через которые вычисляются остальные, выбраны четыре функции, аппроксимируемые цепными дробями:

$$\operatorname{tg} x = \frac{x}{1} - \frac{x^2}{3} - \frac{x^2}{5} - \frac{x^2}{7} - \frac{x^2}{9} - \ldots - \frac{x^2}{(2n+1)} - \ldots;$$

$$\operatorname{th} x = \frac{x}{1} + \frac{x^2}{3} + \frac{x^2}{5} + \frac{x^2}{7} + \frac{x^2}{9} + \ldots + \frac{x^2}{(2n+1)} + \ldots;$$

$$\operatorname{arctg} x = \frac{x}{1} + \frac{x^2}{3} + \frac{4x^2}{5} + \frac{9x^2}{7} + \frac{16x^2}{9} + \ldots + \frac{n^2x^2}{(2n-1)} + \ldots;$$

$$\operatorname{arth} x = \frac{x}{1} - \frac{x^2}{3} - \frac{4x^2}{5} - \frac{9x^2}{7} - \frac{16x^2}{9} - \ldots - \frac{n^2x^2}{(2n+1)} - \ldots$$

Пары этих функций отличаются только знаками членов с индексом $n > 0$, что упрощает их программную реализацию. Кроме того, знаменатели членов всех этих функций совпадают, а в парах функций числители соответствующих членов одинаковы, что также обеспечивает минимизацию аппаратурных затрат на программное обеспечение вычисления этих функций в микроЭВМ ПМК.

При вычислении тригонометрических функций значения аргумента x в градах (десятичных градусах) преобразуются в градусы и аргумент приводится в интервал [0; 90°] или при задании аргумента в радианах интервал [0; $\pi/2$]. Константа $\pi/2$ представлена в ПЗУ с 8 верными цифрами, и поэтому приведение аргумента в указанные интервалы производится раздельно в соответствии с размерностью аргумента, определяемой положением переключателя Р — ГРД — Г. Если аргумент задан в радианах, то с увеличением его абсолютного значения погрешность приведения к интервалу [0; $\pi/2$] будет возрастать при сохранении разрядности константы $\pi/2$. Поэтому для повышения точности вычисления тригонометрических функций этот интервал разбит на два частных интервала [0; 1) и [1; $\pi/2$]. Аналогично интервал аргумента, задаваемого в градусах, разбит на два частных интервала [0; 50°) и [50; 90°]. В первом интервале по разложению в цепную дробь получают значение $\operatorname{tg} x$, а во втором $\operatorname{tg} x = -\operatorname{ctg}(x - \pi/2)$.

Остальные тригонометрические функции вычисляют по формулам

$$\sin x = \operatorname{tg} x/\sqrt{1 + \operatorname{tg}^2 x}; \quad \cos x = 1/\sqrt{1 + \operatorname{tg}^2 x}$$

для первого интервала и

$$\sin x = 1/\sqrt{1 + \operatorname{ctg}^2 x}; \quad \cos x = \operatorname{ctg} x/\sqrt{1 + \operatorname{ctg}^2 x}$$

для второго интервала. Допустимый диапазон изменений аргумента тригонометрических функций независимо от его размерности определяется неравенством $|x| < 10^{10}$.

Функция arctg x вычисляется в интервале изменения аргумента $[0; 10^{100})$, который для повышения точности результата разбивается на частные интервалы $[0; 1)$ и $[1; 10^{100})$. На первом интервале функция arctg x вычисляется по соответствующей цепной дроби, а на втором интервале используется соотношение

$$\operatorname{arctg} x = \pi/2 - \operatorname{arctg} x = \pi/2 - \operatorname{arctg}(1/x).$$

Остальные обратные тригонометрические функции вычисляются по формулам

$$\arcsin x = \operatorname{arctg}(x/\sqrt{1-x^2}); \quad \arccos x = \operatorname{arctg}(\sqrt{1-x^2}/x).$$

Допустимый диапазон изменения аргумента этих функций ограничен значениями $|x| \leqslant 1$. Размерность значений вычисляемых обратных тригонометрических функций зависит от положения переключателя Р — ГРД — Г и может выбираться в радианах, градах или градусах.

Для вычисления показательных функций e^x и 10^x используется аппроксимация функции th x цепной дробью. Экспоненциальная функция вычисляется по формуле

$$e^x = (1 + \operatorname{th}(x/2))/(1 - \operatorname{th}(x/2))$$

при допустимых значениях аргумента $x < 100 \ln 10$. Порядок результата вычисления функции 10^x определяется по формуле INT (x), а функции e^x — по формуле INT $(x/\ln 10)$. Так как $10^x = e^{x \ln 10}$, то вычисление мантисс функций e^x и 10^x выполняется в интервале $[0; \ln 10]$ изменения аргумента $x' = \mathrm{FRC}(x) \times \ln 10$ для функции 10^x и $x' = \mathrm{FRC}(x/\ln 10) \ln 10$ для функции e^x.

Для повышения точности результата вычислений и уменьшения числа звеньев в разложении функции th $(x/2)$ в цепную дробь указанный интервал изменения аргумента x' разбивается на два частных интервала $[0; 1]$ и $[1; \ln 10]$. На первом интервале значение мантиссы функции определяется по формуле

$$M = (1 + \operatorname{th}(x'/2))/(1 - \operatorname{th}(x'/2)),$$

а на втором интервале — по формуле

$$M = 10/e^{\ln 10 - x'} = 10\,(1 - \operatorname{th}((\ln 10 - x')/2))/(1 + \operatorname{th}(1 + \operatorname{th}((\ln 10 - x')/2))).$$

Вычисление логарифмических функций $\ln x$ и $\lg x$ производится по формулам

$$\ln x = 2 \operatorname{arth}((x-1)/(x+1)); \quad \lg x = \ln x/\ln 10$$

с предварительным вычислением обратной гиперболической функции по разложению в цепную дробь. Результат вычисления функции $\ln x$ состоит из произведения порядка аргумента, взятого со своим знаком, на множитель $\ln 10$, к которому прибавляется значение логарифмиче-

ской функции, вычисленной от мантиссы аргумента. Поэтому диапазон изменения аргумента при вычислении логарифмических функций может быть сведен к интервалу [1; 10). Для повышения точности вычислений с уменьшением разрядности результатов промежуточных вычислений и количества членов разложения функции arth x в цепную дробь этот интервал разбивается на два частных интервала [1; 3,1975309] и [3,197531; 10). Граница между интервалами выбрана из условия лучшей сходимости цепной дроби. На первом интервале функция ln x вычисляется по формуле

$$\ln x = 2 \operatorname{arth}((x - 1)/(x + 1)),$$

а на втором интервале — по формуле

$$\ln x = \ln 10 - \ln(10/x) = \ln 10 - 2 \operatorname{arth}((10 - x)(10 + x)).$$

Степенная функция x^y при вводе оператора x^y вычисляется по формуле $x^y = e^{y \ln x}$ в диапазоне аргумента $x > 0$ и $y \ln x < 100 \ln 10$, определяемом допустимым диапазоном изменения аргумента логарифмической функции e^x и ln x. В связи с относительно большой программой, реализующей вычисления этой функции, результат содержит не менее 6 верных цифр, тогда как результаты вычисления e^x и ln x содержат не менее 7 верных цифр. Время выполнения оператора x^y также наибольшее по сравнению с вычислением других элементарных функций.

При вычислении элементарных функций в общем случае используется 10 членов разложения в цепную дробь базовых функций. Однако в зависимости от разрядности операндов при промежуточных вычислениях и значений приведенного аргумента в разложениях в цепную дробь используется от одного до 10 членов разложения в цепные дроби. Число цифр частных, получаемых при вычислении членов (звеньев) цепной дроби, также зависит от значения приведенного аргумента и изменяется от одной до восьми.

Формирование заданной последовательности шестнадцатеричных цифр на индикаторе ПМК также относится к функциональным преобразованиям. Эти последовательности могут формироваться для указания вида или характера получаемых результатов или с другой целью.

Строки символов, содержащие до 7 шестнадцатеричных цифр, в форме, отображаемой на индикаторе (включая пробел при отображении цифры F), удобно формировать с помощью логических операторов входного языка ПМК семейства «Электроника МК-52». Для этого достаточно так подобрать исходные десятичные коды операндов, чтобы результат логической операции содержал после запятой требуемую последовательность символов. Для устранения цифры 8 и запятой в этом результате можно использовать последовательность операторов К{x} В↑ВП k, где k — число формируемых символов.

Так, для формирования на дисплее слова SLICE-4 (слой-4) достаточно нажать клавиши 15314624 В↑1080880К∧К{x} В↑ВП 7 или другую последовательность клавиш для набора кодов логических опе-

рандов, приводящих к искомому результату. Аналогично, например, для высвечивания слова ЕС 1035 следует нажатием клавиш ввести последовательность операторов 1 8 8 8 1 0 3 5 В↑ 1 6 4 7 0 0 0 0 К⋀К {x} В↑ ВП 7. Подобную последовательность операторов можно ввести в программную память для автоматического формирования нужных символов, но в связи с ограниченным ресурсом оперативной памяти целесообразнее предварительно набирать нужные символы на индикаторе в рабочем режиме и засылать их для дальнейшего использования в регистры данных памяти.

Для формирования подобных строк символов на ПМК семейства «Электроника МК-54» , входной язык которых не содержит логических операторов, можно использовать различные методы. Один из них основан на особенностях исполнения оператора ВП, вводимого до операторов набора мантиссы.

Ввод последовательности операторов Сх÷ВП В↑х → П 0 в рабочем режиме приводит к выводу на индикатор и засылке в регистр 0 невысвечиваемой цифры F и запятой, а последующий ввод с циклическим повторением операторов КП → х 0 П → х 0 В↑ приведет к формированию на индикаторе и в регистре 0 символов Е,; Г,; С,; L, —,; 9, ... Для набора на индикаторе и засылки в регистр памяти символов вида ШДДДДДД, где Ш — шестнадцатеричная цифра от А до F, а Д — десятичная цифра, достаточно набрать на клавиатуре последовательность Д Д Д Д Д Д Д $\sin^{-1}$ ВП В↑ х → П2. Например, при ДДДДДДД= = 2346785 получим на индикаторе и в регистре 2 символ Е2346785.

Возможны и другие комбинации с промежуточным формированием сообщения ЕГГОГ, использованием операторов ВП и В↑ и, в частности, косвенной адресации для модификации содержимого регистров памяти. Следует, однако, учитывать, что при использовании таких комбинаций возможно нарушение синхронизации последовательности выполнения операций в управляющем устройстве ПМК (в связи с нестандартным использованием оператора ВП и некоторых других операторов), что приведет к нарушению процесса исполнения функциональных операторов или даже к засылке цифр F во все знакоместа дисплея. При этом выход из подобных ситуаций и восстановление нормальной последовательности выполнения операций при вводе операторов часто оказывается возможным лишь при выключении ПМК и, следовательно, при стирании составленных символов.

Используя аналоги массовых ПМК с памятью программ-данных, можно формировать строки символов, записывая их кодами операторов в область программ оперативной памяти с пересылкой через память программ-данных в регистры данных оперативной памяти с помощью операторов КОД и КПВ. Подобный способ обеспечивает формирование строк из 9 символов с произвольным положением запятой или из 12 символов с фиксированным положением запятой в разряде старшей цифры мантиссы на индикаторе. При необходимости ввести в программную память коды, которые нельзя набрать на клавиатуре (например,

BF), они формируются с помощью логических операторов, засылаются в соответствующие разряды регистра данных и с помощью операторов КПВ и КОД пересылаются в требуемую ячейку программной памяти.

4.4. УРОВЕНЬ СИНХРОПРОГРАММЫ

Коды операторов и директив, вводимых нажатием клавиш или вызываемых из памяти, интерпретируются последовательностями команд, каждая из которых отображается последовательностью адресов синхропрограмм (см. гл. 3), распределенных между полями команды:

$$<К> = <АСП><АСП><АСП><МОД> = <К3><К2><К1><К0>$$

Поля К (3) и К (2) содержат адреса синхропрограмм для обработки областей мантиссы и порядка слов данных, а однобитовое поле К(0) обеспечивает управление вводом информации в регистр R микроЭВМ. Содержимое К(1) < 20 (здесь и далее все адреса команд и двузначные числа представлены в шестнадцатеричной системе счисления) является адресом синхропрограммы, формирующей адрес следующей команды в соответствии с текущими условиями. Если К (1) $\geqslant 20$, то выполняется пересылка R2 (Д14:Д13) : = К (1) с последующим выполнением синхропрограммы с адресом 5F для вычисления адреса следующей команды.

Быстрая память микроЭВМ содержит регистр R и стек SR с такой же разрядностью и организацией, как и страница оперативной памяти:

R1 = R (E1), R2= R (E2), R3 = R (E3), SR1 = SR (E1), SR2= = SR (E2), SR3 = SR (E3).

Эти триггеры используются как рабочие (операционные), а также для запоминания процесса преобразования информации. Общим для микроЭВМ является расположение программного счетчика PC : = R1 (Д14 : Д13). Параметр адреса записывается в R2 (Д14 : Д13) непосредственно из поля К (1) $\geqslant 20$. Регистр SR обычно используется для хранения информации.

Карта памяти регистра R микроЭВМ К745ИК1302 показана на рис.4.2, *а* с указанием флагов и ячеек, а также адресного счетчика (PC), используемого при адресации на синхропрограммном уровне. Аналогичная карта стека программного счетчика SPC показана на рис. 4.2, *б*. Порядок числа, вводимого после оператора ВП, заносится в R2 (Д12:Д10). При каждом последующем вводе оператора-литерала изменяется содержимое R1 (Д12:Д10). Содержимое SR1 (Д12:Д10) вычисляется каждый раз согласно описанию

если R1(Д12) = F, то SR1(Д12:Д10):=SR2 (Д12:Д10) + R1(Д11:Д10), иначе SR1(Д12:Д10):=SR1(Д12:Д10) — R1(Д11:Д10).

В микросхеме К745ИК1303 регистр R используется как рабочий при выполнении функциональных операторов, в нем хранятся исход-

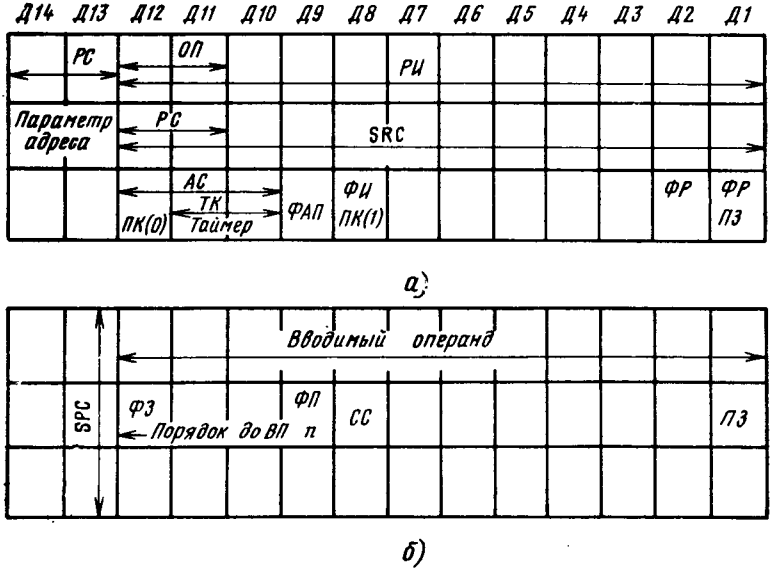


Рис. 4.2

ные данные и промежуточные результаты вычисления функций и содержимое программного счетчика. В эмуляторе накопителя ВЗУ регистр R используется для формирования текущей страницы загружаемого сегмента из текста синхропрограммы и параметров адреса, а также для хранения содержимого адресного счетчика и счетчика числа загружаемых страниц. Регистр SR используется для стека программного счетчика.

Система адресации микросхемы К745ИК13 обеспечивает использование всех возможностей АЛУ и преимущественное применение относительной и косвенной адресации, безусловного или условного увеличения на единицу (инкрементирования) содержимого программного счетчика или его полей. Возможности прямой адресации связаны с использованием содержимого поля К (1) при $20 \leqslant K(1) < FF$ и в меньшей степени с безусловным формированием части или всего содержимого РС синхропрограммой обработки адреса при $0 \leqslant K(1) < 20$.

Эффективная система адресации реализуется при разработке синхропрограмм формирования адреса очередной команды и процедур обращения к подпрограммам с использованием поля К(1) команды. Содержимое R2 (Д14:Д13) может быть использовано на текущей или последующих командах. В последнем случае, как правило, увеличиваются затраты времени, и содержимое поля К(1) целесообразно использовать непосредственно или с некоторой модификацией при выполнении соответствующей команды. Если поле К (1) используют для формирования адреса следующей команды, то необходимо выбрать вид програм-

много ветвления, наиболее удобный для конкретного назначения микроЭВМ. Так, для безусловного перехода следует предварительно присвоить соответствующему разряду слова состояния процесса значение, обеспечивающее требуемое направление условного ветвления в синхропрограмме.

Содержимое синхропрограммы (СП) с адресом 5F у микроЭВМ К745ИК1302, К745ИК1303 и эмулятора накопителя ВЗУ приведено в табл. 4.3, где принадлежность синхропрограммы указана в скобках соответственно символами 02, 03 и Э. В первых двух синхропрограммах поле К (1) используется для организации условного программного ветвления по значению разряда слова состояния, но может быть использовано и для обращения к подпрограмме.

Таблица 4.3

Такт СП	Адрес МК	Обработка данных	Такт СП	Адрес МК	Обработка данных
АСП = 5F (02)			АСП = 5F (03)		
A1	3B	R1 [A] := R1 [A] + 1; S := R1 [A]	A1	39	S := R1 [A] + 1
A2	3C	S := (R2 [A] + 1) × L ∨ ∨ S × $\overline{L}$ R1 [A] := S	A2	3C	R1 [A] := R2 [A] × L ∨ S × $\overline{L}$; т := S; S := R2 [A] × L; R2 [A] := т × L
A3	2F	R1 [A] := S + 1	A3	21	R1 [A] := S + 1
B1	37	S := R1 [B]; SR1 [B] := S	B1	01	S := R1 [B]
B2	3C	S := (R2 [B] + 1) × L ∨ ∨ S × $\overline{L}$; R1 [B] := S	B2	3C	R1 [B] := R2 [B] × L ∨ S × $\overline{L}$; т := S; S := R2 [B] × L; R2 [B] := т × L
B3	01	S := R3 [B]	B3	01	S := R3 [B]
C1	00		C1	06	т := S; S := S1; S1 := т
C2	00		C2	1F	S := F + 1 + A × $\overline{L}$
C3	00		C3	19	т := S1 + S; L := П; S := т; S1 := т
АСП = 5F (Э)			АСП = 1D (03)		
A1	1F	SR1 [A] := R1 [A] + 1; L := П; R1 [A] := S	A1	38	S := R1 [A] + F; L := П; R1 [A] := S
A2	1D	S := 1	A2	14	S := S + F
A3	00	R3 [A] := 0	A3	0C	R1 [A] := S
B1	21	SR1 [B] := R1 [B] + L; R1 [B] := S	B1	00	

Окончание табл. 4.3

Такт СП	Адрес МК	Обработка данных	Такт СП	Адрес МК	Обработка данных
B2	23	R2 [B] : = $\overline{\text{R2 [B]}}$; L : = 0	B2	08	S : =0; L : = 1
B3	0F	R3 [B] : = F; S1 : = F	B3	06	т : = S; S : = S1; S1 : = т
C1	03	R1 [C] : = 3; S1 : = 3	C1	20	R3 [B] : = 0
C2	25	R1 [C] : = R2 [C]; S : =S1	C2	1B	S+7; L : = П
C3	0F	R3 [C] : = F; S1 : = F	C3	34	R3 [C] : = S1
АСП = 16 (02)			АСП = 11 (03)		
A1	07	S : = 2	A1	0D	R3 [C] : = S+L; S : = S+L
A2	0B	S : = S+8; L : = П	A2	06	т : = S; S : = S1 : S1 : = т
A3	0D	R1 [A] : = S	A3	3B	R3 [A] : =F+L+A×$\overline{\text{L}}$
B1	03	R1 [B] : = S+1	B1	13	R1 [B] : = R1 [B]+S
B2	0C	S : = S+1	B2	0A	S : = 6
B3	0E	S : =F	B3	02	R2 [B] : =S
C1	1E	S : = $\overline{\text{S}}$	C1	00	
C2	3A	S : = S + $\overline{\text{T}}$; L : = П	C2	27	S : = $\overline{\text{R2 [C]}}$+L; L : = П
C3	2B	S1 : = R3 [C]	C3	00	
АСП = 0C (02)			АСП = 15 (03)		
A1	35	S : = R1 [A] + L	A1	37	SR1 [A] : = R1 [A]+1
A2	34	S : = S+S1; L : = П	A2	12	S : =3
A3	0D	R1 [A] : = S	A3	2A	R1 [A] : = $\overline{\text{S}}$+1
B1	24	S : = 4	B1	31	SR1 [B] : = R1 [B]; R1 [B] : = S
B2	1E	S : = $\overline{\text{S}}$	B2	02	R1 [B] : = S
B3	1A	S : = R3 [B]+S; L : = П	B3	00	
C1	09	S : = 6	C1	12	S : = 3
C2	0C	S : = S+1	C2	06	т : = S; S : = S1; S1 : = т
C3	0F	т : = S; S : = S1; S1 : =т	C3	11	S1 : = S1+$\overline{\text{L}}$

В микроЭВМ К745ИК1302 очередная команда выбирается по адресу в соответствии с синхропрограммой

$R1(Д13) := (R1(Д13) + 2) \times \bar{L} \vee (R2(Д13) + 2) \times L$; $R1(Д14) := (R1(Д14) \times \times \bar{L} \vee R2(Д14) + 1) \times L$.

Стек программного счетчика расположен в области SPC := SR (Д14: Д13), и при выполнении этой синхропрограммы в область стека безусловно передается адрес возврата из подпрограммы

SR3(Д14 : Д13) : = SR2(Д14 : Д13); SR2 (Д14:Д13) : = SR1(Д14 : Д13); SR1(Д13):=R1(Д13) + 1; SR1(Д14):=R1(Д14).

Эта синхропрограмма обеспечивает безусловное и условное обращение к подпрограмме.

В микроЭВМ К745ИК1303 адрес очередной команды вычисляется согласно описанию (табл. 4.3)

$R1(Д13) := (R1(Д13) + 2) \times \bar{L} \vee (R2(Д13) + 1) \times L$; $R1(Д14) := R1(Д14) \times \times \bar{L} \vee R2(Д14) \times L$.

Обращение к подпрограмме выполняется при L = 1, а адрес возврата из подпрограммы передается в R2 (Д14:Д13), где до этого находилось содержимое поля К(1) команды. В эмуляторе ВЗУ содержимое поля К (1) используется на последующих командах, а адрес перехода определяется содержимым аккумулятора S, передаваемым в младшую тетраду R1 (Д13) := S; R1 (Д14) : = 1.

Для формирования адреса очередной команды используются также синхропрограммы с адресами $0 \leqslant К(1) < 20$. Синхропрограмма с адресом $0 \leqslant К (1) < 4$ формирует разрешение на вывод информации из РИ на индикатор и используется в драйвере индикаторного устройства. В противном случае синхропрограммы с этими адресами могут быть использованы для обработки мантиссы и порядка. Синхропрограмма обработки адреса охватывает временной интервал Д13, Д14, а формат текста синхропрограммы допускает обработку по крайней мере трех таких временных интервалов, что позволяет шире использовать метод наложения. Он основан на записи двух или более синхропрограмм по одному адресу для использования в различных областях прикладной программы без искажения сохраняемых данных.

Наиболее просто накладываются синхропрограммы, обрабатывающие информацию и проверяющие условия. Так, в табл. 4.3 синхропрограмма с адресом АСП = 5F (03) выполняет две функции — формирование адреса и проверка условия $R3 [B] \geqslant 7$. Перед выполнением синхропрограммы проверки L := 0, и содержимое R3 [B] пересылается в S1, а в аккумуляторе S формируется константа 9. При суммировании содержимого S и S1 при $R3 [B] \geqslant 7$ значение L := 1.

Рассмотрим различные способы вычисления адреса следующей команды, реализованные в синхропрограммах с адресами $0 \leqslant АСП \leqslant 1F$.

1. Вычисление адреса следующей команды при изменении на единицу содержимого программного счетчика:

а) R1(Д13):=R1(Д13) + 1; АСП = 0(Э), АСП=06(02), АСП=09(03);
б) R1(Д13):=R1(Д13) + 2; АСП = 08(02), АСП=ОС(03);

в) R1(Д13):=R1(Д13) + 3; АСП = 09(02);
г) R1(Д13):=R1(Д13) — 1; АСП = 10(02); АСП = 0A(03);
д) R1(Д13):=R1(Д13) — 2; АСП = 0A(02); АСП = 1D(03).

Синхропрограмма с АСП = 1D (03) (табл. 4.3) кроме вычисления адреса R1 (Д13) := R1 (Д13) — 2 выполняет запись нуля в ячейки R3 [B], R3 [C] и проверку содержимого S1 := S1 + 7.

2. Безусловный переход или засылка константы в разряд РС:

а) R1(Д14:Д13):=00; АСП = 0A(Э);
б) R1(Д13):=A; R1(Д14):=B; АСП = 16(02);
в) R1(Д13):=8; R1(Д14):=6; АСП = 14(02);
г) R1(Д13):=9; R1(Д14):=2; АСП = 1B(03);
д) R1(Д13):=1;АСП = 07(Э);
е) R1(Д14):=2; АСП = 0B(Э);
ж) R1(Д13):=F; R1(Д14):=R1(Д14) + 7; АСП = 07(02).

Синхропрограмма с АСП = 16 (02) (табл. 4.3) при формировании адреса обеспечивает выполнение безусловного перехода по адресу BA, проверку состояния разряда T слова состояния процессора и передачу в S1 содержимого R3 [C].

3. Индексная адресация относительно РС в зависимости от содержимого S или S1:

а) R1(Д13):=R1(Д13) + S; АСП = 01(02);
б) R1(Д14):=R1(Д14) + S; АСП = 00(02);
в) R1(Д14):=R1(Д14) + S1; АСП = 11(03).

Синхропрограмма с АСП = 11(03) (табл. 4.3), вычисляющая адрес, выполняет процедуру R3 [C] := $S + 1$; R3 [A] := $0 \times L \vee 9 \times \overline{L}$; R2×[B] := 6 и проверку условия R2 [C] + L.

4. Условные программные ветвления по содержимому разряда слова состояния процессора

а) R1(Д13):=R1(Д13) + L; АСП = 06(Э), АСП = 11(02);
б) R1(Д13):=R1(Д13) + $\overline{L}$; R1(Д14):=$\overline{L}$; АСП = 0D(Э);
в) R1(Д13):=(R1(Д13)+ F) × L$\vee$(S1 + 1)L; АСП = 0E(Э);
г) R1(Д13):=R1(Д13) + L + 1; АСП = B(02), АСП = 17(03);
д) R1(Д13):=R1(Д13) + S1 + L; АСП = OC(02);
е) R1(Д13):=R1(Д13) + SL + 1; R1(Д14):=R1(Д14)+FL; АСП = 0E(02);
ж) R1(Д13):=R1(Д14) + L; АСП = 1A(03);
з) R1(Д14):=R1(Д14) + F + L; АСП = 13(03);
и) R1(Д14):=(R1(Д14)+ S) × $\overline{L}\vee$R2(Д14) × L; АСП = 12(02).

Синхропрограмма с АСП = OC (02) (табл. 4.3) обеспечивает вычисление адреса перехода в зависимости от состояния ячейки L. При обработке других областей регистра R выполняется проверка R3 [B]$\geqslant$ $\geqslant$ 5 и запись константы 7 в ячейку S1.

5. Обращение к подпрограмме обеспечивается описанными выше синхропрограммами с АСП=5F. Обозначив процедуру SR1 (Д13) := := R1 (Д13), SR1 (Д14) := R1 (Д14) символом СТЕК, приведем еще несколько вариантов такого обращения:

а) СТЕК; R1(Д13):=1; R1(Д14):=0; АСП=17(02);
б) СТЕК; R1(Д14:Д13):=R2(Д14:Д13); АСП=07(03);
в) СТЕК; R1(Д13):=S; R1(Д14):=S1; АСП=0A(03);

г) СТЕК; R1(Д13):=$\bar{S}$ + $\hat{2}$; R1(Д14):=3; АСП=1F(03);
д) СТЕК; R1(Д13):=D; R1(Д14):=2; АСП=16(03);
е) СТЕК; R1(Д13):=3; R1(Д14):=3; АСП=18(03);
ж) СТЕК; R1(Д13):=D; R1(Д14):=3; АСП=15(03);
з) СТЕК; R1(Д13):=0; R1(Д14):=6; АСП=19(03).

Последние пять процедур обеспечивают безусловное обращение к наиболее важным подпрограммам микроЭВМ К745ИК1303. Синхропрограмма с АСП-15 (03) кроме обработки адреса используется для присвоения S1 := 3 + $\overline{L}$.

6. Возврат из подпрограммы реализуется процедурами:

а) R1(Д14:Д13):=SR1 (Д14:Д13); АСП=1E(02), АСП=04(03);
б) R1(Д13):=SR1(Д13) + 2; R1(Д14):=SR1(Д14); АСП=14(03);
в) R1(Д14:Д13):=(SR1(Д14:Д13)) × L$\vee$(S + 1) × $\overline{L}$; АСП=08(Э).

При выполнении синхропрограммы с АСП = 08(Э), приведенной в табл. 4.3, в области обработки мантиссы выполняется присвоение R2 (Д9) := 8; S :=4; L := 0.

Рассмотренные способы вычисления адреса следующей команды позволяют эффективно использовать всю область памяти микроЭВМ К745ИК1302, К745ИК1303 и эмулятора накопителя ВЗУ.

Отображение адресного пространства областей памяти данных, операционного стека, программной памяти, эмулятора накопителя ВЗУ и ВЗУ на адресное пространство физических устройств памяти выполняется в контроллере ВЗУ и эмуляторе накопителя ВЗУ для областей программ, данных и программ-данных микроЭВМ К745ИК1302 и для области операционного стека в микроЭВМ К745ИК1303 и К745ИК1306.

При обращении к памяти данных адрес *N* регистра указывается в шестнадцатеричной системе счисления или преобразуется в нее при косвенной адресации. В оперативной памяти адрес этого регистра находится на странице с тем же адресом и адрес непосредственно передается в адресный счетчик АС (2 : 0) = R3 (Д12 : Д10) с выполнением процедуры доступа к оперативной памяти. Адресный счетчик содержит три тетрады (см. рис. 4.2, *а*).

Адрес страницы, к которой происходит обращение, записывается в шестнадцатеричной системе счисления в R3 (Д12), адрес слова на странице — в семеричной системе счисления в R3 (Д11), R3 (Д10) и используется для подсчета числа страниц или слов на ней от начального адреса. В процессе доступа к оперативной памяти происходит сравнение содержимого R3 (Д12) или R3 (Д11) с текущим значением R3 (Д10). При обращении к программной памяти АС (2) :=INT (PC/7) mod $10_{и}$; АС (1) := PC—7 INT (PC/7) mod $10_{и}$. Адресный счетчик позволяет обращаться к программной памяти, для которой NPG $\leqslant$ 16 (Мр $\leqslant$ 112 байт), по условию ОП := Мр (PC) mod 112.

Еще одна особенность обращения к оперативной памяти ПМК с NPG $<$ 16 связана с тем, что вследствие ограниченности ресурса операционная система не контролирует выход адреса обращения в адресном счетчике за пределы адресного пространства программной

памяти и памяти данных. Выполнение оператора с ошибочным адресом не блокируется, выполнение программы не прерывается, и диагностическое сообщение в этом случае не выдается. С учетом работы адресного счетчика соотношения между адресами, записанными в формате оператора или содержащимися в программном счетчике, и фактической областью памяти данных, а также программной памятью, к которой обеспечивается доступ, будут следующими:

Мр (N): = Мд((N) mod NPG), Мр(РС) = Мр(((РС) mod 112) mod 7 NPG), Мд ($ЕА_н$ mod NPG).

Например, для NPG = 15 при РС = 2, РС=А7, РС = В4 доступной окажется ячейка Мр(02). При выполнении оператора Кх→ → П9 и 15 = Р9 будет вычислен исполнительный адрес $ЕА_н$ = F, но для NPG = 15 содержимое РХ будет записано в Р0, а для NPG= = 14 — в Р1. Этот механизм адресации необходимо учитывать не столько во избежание ошибок при адресации, устраняемых при отладке, сколько для возможного использования при нехватке программной памяти. Так, если оператор безусловного перехода на начало не вмещается в конце программы, то допустим естественный переход в начало программной памяти за счет увеличения содержимого РС на единицу после выполнения оператора из последней ячейки программной памяти. В подобных случаях необходимо учитывать последующие обращения к программной памяти.

Пусть, например, в программной памяти ПМК «Электроника МК-61» по адресам от 00 до 50 размещен линейный участок прикладной программы, на который нужно перейти после выполнения оператора ОП:=Мр(А4) из последней ячейки. В этом случае после выполнения этого оператора РС = А4 + 1 = А5 и будет выбран код оператора ОП:=Мр(00), а также последующие операторы до ОП:=Мр(06) при РС = В1. Здесь адресный счетчик переполняется и при РС = = В2 считывается код ОП:=Мр(00). Далее будет сказываться конечный цикл программного счетчика с максимальным содержимым $РС_{max}$ = F9. Поэтому будут считываться коды до ОП:=Мр(47) при РС = F9, после чего РС переполняется и следующими начнут считываться коды операторов, начиная с ОП:=Мр(00) при РС = 00. Таким образом, соотношение между содержимым РС и фактическим адресом обращения к блоку Мр будет следующим: Мр(РС)= Мр(((РС)mod160) × × mod112)mod7NPG. Напомним, что в режиме программирования при РС > > С0 индицируется только содержимое РС, а содержимое ячеек Мр не отображается.

При обращении к памяти данных по адресу за пределами ее адресного пространства можно повысить гибкость системы адресации ПМК, используя косвенное обращение к блоку Мр или Мд через один из регистров памяти с модификацией и без модификации содержимого этого адресного регистра. Например, при выполнении оператора К П→ х 0 в ПМК «Электроника МК-54» реализуется адресация с предварительным уменьшением на единицу содержимого адресного регистра 0 (предавтодекрементная), а при выполнении оператора К х→ ПЕ (с вводом символа Е нажатием клавиши В↑) — косвенная адресация без уменьшения содержимого регистра 0. Однако эту возможность пользователь должен использовать осмотрительно: она нарушает програм-

мную совместимость ПМК с разными NPG. Поэтому при подобном использовании операторов косвенной адресации необходимо четко оговаривать, для какого типа ПМК она может быть использована.

В табл. 4.4 приведена подпрограмма (в системе команд микроЭВМ К745ИК1302), обеспечивающая адресацию к блокам Мр, Мд и Ms оперативной памяти ПМК. Синхропрограммы, к которым обращаются команды этой подпрограммы, хранятся по адресам АСП-5F (табл. 4.3), а остальные указаны в табл. 4.5.

На командах с адресами 63, 64 и 66 происходит преобразование содержимого РС в систему счисления, в которой работает адресный счетчик (АС), и запись преобразованного значения в АС. Начальный адрес в системе временной адресации определяется на команде с адресом 01, а вычисление требуемой страницы оперативной памяти выполняется по команде с адресом 02. Запись информации в блок Мд производится по команде с адресом D4, а вызов содержимого регистра памяти — по команде с адресом D6. По команде с адресом C9 происходит обмен содержимым области стека SPC и страницы программной памяти. Запись информации в блок Ms выполняется по команде с адресом BC. Вызов содержимого страницы памяти программ-данных при выполнении оператора обмена страниц программ производится по команде с адресом C8, а при выполнении оператора обмена регистров данных — по команде с адресом C7.

В эмуляторе накопителя ВЗУ параметры сегментов, которые загружаются в блок Мр при выполнении операторов КР, содержатся в формате команды РС (P — 1) блока Ме. Список параметров сегментов эмулятора располагается в памяти команд с адресами 02 ...0Д. Объем сегмента в страницах (SPGe) записывается как адрес синхропрограммы в поле К (3). Начальный адрес сегмента EAe, указывающий адрес в памяти команд, записывается как параметр адреса в поле К (1).

Таблица 4.4

Адрес команды	АСП К[3]	АСП К[2]	АСП К[1]	МОД К[0]	Обработка данных	Определение адреса перехода
63	40	4B	06	0	R3 (Д11 : Д10) : = : = R2 (Д12 : Д11); R3 (Д12) : = F	PC : = PC + 1 = 64
64	40	4A	52	0	R3 (Д11 : Д10) : = : = R3 (Д11 : Д10) − 7; L : = П; R3 (Д12) : = : = R3 (Д12) + 1	Если L = 1, то PC : = 64, иначе PC : = 66
66	40	49	1B	0	R3 (Д11) : = R3 (Д12); R3 (Д12) : = R3 (Д10) + D; S1 : = 9	R1 (Д13) : = : = R1 (Д13) + 1; SR1 (Д14) : = : = R1(Д14); PC : = 01

Окончание табл. 4.4

Адрес команды	АСП К[3]	АСП К[2]	АСП К[1]	МОД К[0]	Обработка данных	Определение адреса перехода
01	60	73	11	0	Поиск кода «метка»; R3 (Д10) : = 01	PC : = PC + 1
02	40	48	11	0	S : = R3 (Д11) + $\bar{S}$ + 1; S : = S + 2; S : = S + 1; L : = П	Если L = 1, то PC : = 03, иначе PC : =02
03	40	02	04	1	S : = 4	PC : = 04
04	40	40	16	0	Временная задержка	PC : = BA
BA	40	9	0D	1	S1 + 9; L : = П; S : = 1	R1 (Д13) : = : = R1 (Д13) + S × × L + 1; R1 (Д14) : = : = R1 (Д14) + 1
CB	40	7A	0D	1	S : = 1; L : = 1	То же
DD	5A	56	05	0	S : = R1 (Д1); для i = 1 до 11 (R1 (Дi) : = R1 (Дi + 1)	R1 (Д13) : = S1 (4 $\vee$ 6)
D6	36	36	1E	0	R1 (Д12 : Д1) : = : = M1 (Д12 : Д1)	PC : = SR1 (Д14 : : Д13)
D4	37	37	1E	0	M1 (Д12 : Д1) : = : = R1 (Д12 : Д1)	PC : = SR1 (Д14 : : Д13)
CC	40	17	0E	1	S1 + 6; L : = П; S : = E	R1 (Д13) : = : = R1 (Д13) + S × × L + 1; R1 (Д14) : = : = R1 (Д14) + F × L
BB	23	24	06	0	S : = R2 (Д1); для i = 1 до 13 (R2 (Дi) : = R2 (Дi + 1))	PC : = PC + 1 = BC
BC	3A	3A	1A	0	M2 : = R2	PC : = SR1 (Д14 : : Д13)
CD	40	73	05	0	R3 (Д10) : = 0	R1 (Д13) : = S1
C7	34	34	1E	0	т : = R1 (Д12 : Д1); R1 (Д12 : Д1) : = : = M2 (Д12 : Д1); M2 (Д12 : Д1) : = т	PC : = SR1 (Д14 : : Д13)
C8	38	38	19	0	R2 : = M2	PC : = SR1 (Д14 : : Д13)
C9	39	39	18	0	т : = R2; R2 : = M3; M3 : = т	PC : = SR1 (Д14 : : Д13)

Таблица 4.5

Такт СП	Адрес МК	Обработка данных	Такт СП	Адрес МК	Обработка данных
АСП = 02			АСП = 04		
A1	15	S := R1 [A] + S + 1; L := П	A1	00	
A2	18	R1 [A] := S + A × $\overline{L}$	A2	03	R1 [A] := S
A3	09	S := 6	A3	0E	S := F
B1	16	S := R1 [B] + S + L; L := П	B1	1E	S := $\overline{S}$
B2	18	R1 [B] := S + A × $\overline{L}$	B2	33	S := S + A $\overline{L}$; L := П
B3	09	S := 6	B3	00	
C1	16	S := R1 [C] + S + L; L := П	C1	00	
C2	18	R1 [C] := S + A × $\overline{L}$	C2	00	
C3	24	S := 4	C3	00	
АСП = 05			АСП = 06		
A1	23	R1 [A] := S1	A1	11	R1 [A] := R1 [A] + 1; L := П
A2	00		A2	32	R2[A] := R2[A + 1]
A3	00		A3	00	
B1	00		B1	00	
B2	2F	B3 [B—1] := S + 1	B2	00	
B3	00		B3	03	R2 [B] := S
C1	2C	S := R1 [C] + 1; L := П	C1	00	
C2	00		C2	0E	S := F
C3	01	S := R3 [C]	C3	1A	S := R3 [C] + S; L := П
АСП = 09			АСП = 0D		
A1	1C	S := R1 [A] + 1	A1	3D	R1 [A] := R1 [A] + S × L + 1
A2	0C	S := S + 1	A2	00	
A3	2F	R1 [A] := S + 1	A3	00	
B1	09	S := 6	B1	1C	S := R1 [B] + 1
B2	1E	S := $\overline{S}$	B2	03	R1 [B] := S
B3	34	S := S1 + S; L := П	B3	0E	S := F
C1	0E	S := F	C1	0A	S := S + F; L := П
C2	1E	S := $\overline{S}$	C2	0F	т := S; S := S1; S1 := т
C3	0C	S := S + 1	C3	06	S := R3 [C] + F; L := П

Продолжение табл. 4.5

Такт СП	Адрес МК	Обработка данных	Такт СП	Адрес МК	Обработка данных
АСП = 0E			АСП = 11		
A1	3D	R1 [A] : = R1 [A] + S × L + 1	A1	35	S : = R1 [A] + L
A2	00		A2	03	R1 [A] : = S
A3	0E	S : = F	A3	07	S : = 2
B1	3F	(S : = R1 [B] + S; L : = П) × × L ∨ (S : = R1 [B]) $\overline{L}$	B1	0C	S : = S + 1
B2	03	R1 [B] : = S	B2	1E	S : = $\overline{S}$
B3	01	S : = R3 [B]	B3	1A	S : = R3 [B] + S; L : = П
C1	00		C1	00	
C2	00		C2	00	
C3	0E	S : = F	C3	00	
АСП = 16			АСП = 17		
A1	07	S : = 2	A1	3C	S : = (R1 [A] + 1) × L ∨ ∨ S × $\overline{L}$; R3 [A—1] : = S
A2	0B	S : = S + 8; L : = П	A2	03	R1 [A] : = S
A3	0D	R1 [A] : = S	A3	00	
B1	0C	S : = S + 1	B1	09	S : = 6
B2	03	R1 [B] : = S	B2	34	S : = S1 + S; L : = П
B3	0E	S : = F	B3	0E	S : = F
C1	1E	S : = $\overline{S}$	C1	1E	S : = $\overline{S}$
C2	3A	S : = S + $\overline{T}$; L : = П	C2	0C	S : = S + 1
C3	2B	S1 : = R3 [C]	C3	1E	S : = $\overline{S}$
АСП = 18			АСП = 19		
A1	2E	R1 [A] : = SR1 [A]	A1	2E	R1 [A] : = SR1 [A]
A2	01	S : = R2 [A]	A2	30	S : = M2 [A]
A3	31	R2 [A] : = M3 [A]; M3 [A] : = S	A3	03	R2 [A] : = S
B1	2E	R1 [B] : = SR1 [B]	B1	2E	R1 [B] : = SR1 [B]
B2	01	S : = R2 [B]	B2	30	S : = M2 [B]
B3	31	R2 [B] : = M3 [B]; M3 [B] : = S	B3	03	R2 [B] : = S
C1	00		C1	00	
C2	00		C2	00	
C3	00		C3	00	

Такт СП	Адрес МК	Обработка данных	Такт СП	Адрес МК	Обработка данных
АСП = 1A			АСП = 1B		
A1	2E	R1 [A] : = SR1 [A]	A1	3B	S : = R1 [A] + 1; SR1 [A] : = S
A2	2D	M2 [A] : = S; S : = R2 [A]	A2	04	S : = 0; L : = 1
A3	00		A3	2F	R1 [A] : = S + 1
B1	2E	R1 [B] : = SR1 [B]	B1	37	SR1 [B] : = R1 [B]; S : = R1 [B]
B2	2D	M2 [B] : = S; S : = R2 [B]	B2	12	R1 [B] : = 0
B3	00		B3	00	
C1	00		C1	00	
C2	00		C2	00	
C3	00		C3	00	
АСП = 1E			АСП = 23		
A1	2E	R1 [A] : = SR1 [A]	A1	04	S : = 0; L : = 1
A2	00		A2	14	S : = R2 [A]; R2 [A] : = : = R2 [A + 1]
A3	00		A3	00	
B1	2E	R1 [B] : = SR1 [B]	B1	00	
B2	00		B2	32	R2 [B] : = R2 [B + 1]
B3	00		B3	00	
C1	2E	R1 [C] : = SR1 [C]	C1	00	
C2	00		C2	32	R2 [C] : = R2 [C + 1]
C3	00		C3	0C	S : = S + 1
АСП = 24			АСП = 34		
A1	0A	S : = S + F; L : = П	A1	01	S : = R1 [A]
A2	32	R2 [A] : = R2 [A + 1]	A2	31	R1 [A] : = M2 [A]; M2 [A] : = S
A3	00		A3	00	
B1	00		B1	01	S : = R1 [B]
B2	32	R2 [B] : = R2 [B + 1]	B2	31	R1 [B] : = M2 [B]; M2 [B] : = S
B3	00		B3	00	
C1	00		C1	01	S : = R1 [C]
C2	32	R2 [C] : = R2 [C + 1]	C2	31	R1 [C] : = M2 [C]; M2 [C] : = S
C3	00		C3	36	S1 : = S1 + 1; L : = П

Продолжение табл. 4.5

Такт СП	Адрес МК	Обработка данных	Такт СП	Адрес МК	Обработка данных
АСП = 36			АСП = 37		
A1	30	S := M1 [A]	A1	2D	M1 [A] := S; S := R1 [A]
A2	03	R1 [A] := S	A2	00	
A3	00		A3	00	
B1	30	S := M1 [B]	B1	2D	M1 [B] := S; S := R1 [B]
B2	03	R1 [B] := S	B2	00	
B3	00		B3	00	
C1	30	S := M1 [C]	C1	2D	M1 [C] := S; S := R1 [C]
C2	03	R1 [C] := S	C2	00	
C3	2B	S1 := R3 [C]	C3	00	
АСП = 38			АСП = 39		
A1	0A	S := S + F : L := П	A1	00	
A2	30	S := M2 [A]	A2	01	S := R2 [A]
A3	03	R2 [A] := S	A3	31	R2 [A] := M3 [A]; M3 [A] := S
B	00		B1	00	
B2	30	S := M2 [B]	B2	01	S := R2 [B]
B3	03	R2 [B] := S	B3	31	R2 [B] := M3 [B]; M3 [B] := S
C1	00		C1	00	
C2	30	S := M2 [C]	C2	01	S := R2 [C]
C3	03	R2 [C] := S	C3	31	R2 [C] := M3 [C]; M3 [C] := S
АСП = 3A			АСП = 48		
A1	00		A1	00	
A2	2D	M2 [A] := S; S := R2 [A]	A2	00	
A3	00		A3	1C	S := R3 [A] + 1
B1	00		B1	03	R3 [B—1] := S
B2	2D	M2 [B] := S; S := R2 [B]	B2	1E	S := $\overline{S}$
B3	00		B3	15	S := R3 [B] + S + 1; L := П
C1	00		C1	02	S := S + 1; L := П
C2	2D	M2 [C] := S; S := R2 [C]	C2	02	S := S + 1; L := П
C3	00		C3	0C	S := S + 1

Такт СП	Адрес МК	Обработка данных	Такт СП	Адрес МК	Обработка данных
АСП=49			АСП=4А		
A1	07	S:=2	A1	09	S:=6
A2	1E	S:=$\overline{S}$	A2	1E	S:=$\overline{S}$
A3	10	S:=R3[A]+S	A3	1A	S:=R3[A]+S; L:
B1	0F	т:=S; S:=S1; S1:=т	B1	30	R3[B—1]:=S+A×L
B2	09	S:=6	B2	1D	S1:=0
B3	32	R3[B]:=R3[B+1]	B3	17	S:=R3[B]+F+L: L:=П
C1	1E	S:=$\overline{S}$	C1	03	R3[C—1]:=S
C2	0F	т:=S; S:=S1; S1:=т	C2	0F	т:=S; S:=S1; S1:=т
C3	08	т:=S; S:=R3[C]; R3[C]:=т	C3	3D	R3[C]:=R3[C]+S×L+1
АСП=4В			АСП=56		
A1	07	S:=2	A1	32	R1[A]:=R1[A+1]
A2	0B	S:=S+8; L:=П	A2	00	
A3	1A	S:=R3[A]+S; L:=П	A3	00	
B1	1D	S1:=0	B1	32	R1[B]:=R1[B+1]
B2	28	R3[B—1]:=R2[B]	B2	00	
B3	00		B3	00	
C1	0E	S:=F	C1	00	
C2	28	R3[C—1]:=R2[C]	C2	03	R1[C]:=S
C3	08	т:=S; S:=R3[C]; R3[C]:=т	C3	00	
АСП=5А			АСП=60		
A1	14	S:=R1[A]; R1[A]:=:=R1[A+1]	A1	00	
A2	00		A2	00	
A3	00		A3	00	
B1	32	R1[B]:=R1[B+1]	B1	00	
B2	00		B2	30	S:=M2[B]
B3	00		B3	00	
C1	32	R1[C]:=R1[C+1]	C1	02	S:=S+1; L:=П
C2	00		C2	24	S:=4
C3	00		C3	1E	S:=$\overline{S}$

Окончание табл. 4.5

Такт СП	Адрес МК	Обработка данных	Такт СП	Адрес МК	Обработка данных
АСП = 73			АСП = 7A		
A1	0E	S : = F	A1	00	
A2	1E	S : = $\overline{S}$	A2	00	
A3	25	S : = (S1 $\vee$ $\overline{S}$) + 1	A3	12	R2 [A] : = 0
B1	00		B1	00	
B2	2F	R3 [B—1] : = S + 1	B2	00	
B3	00		B3	12	R2 [B] : = 0
C1	00		C1	04	S : = 0; L : = 1
C2	09	S : = 6	C2	0C	S : = S + 1
C3	0C	S : = S + 1	C3	12	R2 [C] : = 0

Содержимое команды <K> = <SPGe><5F><EAe> <0> и при ее исполнении R2 (Д9) = SPGe; R2 (Д14:Д13) := EAe.

Синхропрограмма с адресом АСП = 5F (табл. 4.3) выполняет присвоение S : = 3, используемое при вычислении адреса перехода R1 (Д13) : = S; R1 (Д14) : = 1. Фрагмент программы, где выполняется определение адреса блока Ме, идентификация этого адреса из содержимого канала связи и обращение к списку параметров сегмента, приведен в табл. 4.6. Синхропрограммы, к которым обращаются команды этого фрагмента, записаны в табл. 4.7. Микрокоманды, которые используются в синхропрограммах, описаны в системе микрокоманд эмулятора накопителя ВЗУ при описании уровня микрокоманд ПМК.

Поиск кода «метка» в оперативной памяти обеспечивает определение начала отсчета в системе временной адресации. Запись КС (0) : =F выполняется по команде с адресом 01 при включении питания для подтверждения приема адреса блока Ме (при выполнении оператора загрузки сегмента в блок Мр с адресом АМе $\neq$ 0) и для сообщения о завершении загрузки сегмента в память Мр. Содержимое КС (0) = 3 обеспечивает идентификацию кода операции в операторе КР, переданном в канал связи (КС) для исполнения. Обращение к списку параметров сегментов выполняется косвенной адресацией по содержимому КС (0) + + 2 на команде с адресом 0F. Проверка включения эмулятора накопителя ВЗУ производится по команде с адресом 1E.

В качестве ВЗУ в ПМК «Электроника МК-52» используется РПЗУ 1601РР1 с подключенными дополнительными съемными блоками расширения памяти (БРП) с 4-разрядными шинами данных. При размещении информации в РПЗУ и для ее записи в дополнительный БРП необходимо знать соответствие между адресами ячеек этого блока и

Таблица 4.6

Адрес команды	АСП К[3]	АСП К[2]	АСП К[1]	МОД К[0]	Обработка данных	Определение адреса перехода
00	0	03	06	1	Поиск кода «метка»	Если «метка», то PC:=:=01, иначе PC:=00
01	1	B	B	0	КС(0):=F	PC:=21
21	A	5	1	1	Сопровождение записи в КС сигналом I8:=0	Если включение, то PC:=0, иначе (если АМе принят, то PC:=:=0E, иначе PC:=1E)
0E	D	3	6	0	Поиск кода «метка»	Если «метка», то PC:=:=0F, иначе PC:=0E
0F	A	B	E	1	Идентификация кода операции КС(1)=3	Если КС(1)≠3, то PC:=0E, иначе R1(Д13):=КС(0)+2
1E	E	3	D	0	Определение адреса Ме по входу H	Если Ме присвоен адрес, то PC:=1F, иначе PC:=0E
1F	D	3	6	1	Поиск кода «метка»	Если «метка», то PC:=10, иначе PC:=1F
10	A	C	C	1	Сравнение адреса Ме и содержимого КС	Если КС=АМе, то PC:=00, иначе PC:=1F

ячеек программной памяти или памяти данных: $П = П0 + 14N + i$, где N = INT ((П — П0)/14) — адрес регистра, содержащего слово с адресом П в ВЗУ; i = П — П0 — 14INT ((П — П0)/14) — индекс слова в регистре PN, указывающий на слово с адресом П в ВЗУ; П0 — начальный адрес сегмента в ВЗУ; П — текущий адрес ВЗУ, К — адрес программной памяти (содержимое младшей тетрады слова по этому адресу соответствует слову из ВЗУ по адресу П).

Адресация слов на странице памяти Мр для ПМК с памятью Ms показана на рис. 2.12. Соответствие адресов ВЗУ и памяти Мр для этих ПМК описывается несложными выражениями

$$П = П0 + 2К;\ К = \text{INT}\ ((П — П0)/2).$$

Для ПМК без памяти программ-данных соответствие адресов ВЗУ и Мр описывается более сложными выражениями

$$П = П0 + 2Т;\quad Т = 7\text{INT}\ (К/7) + (К + 6)\ \text{mod}\ 7;$$
$$К = 7\text{IN}\ Т \times (P/7) + (P+1)\text{mod}\ 7,\ P = \text{INT}((П — П0)/2).$$

При выполнении директив обмена информацией контроллер ВЗУ оперирует единицами объема информации, равными странице. Поэтому десятичный размер сегмента в байтах LL (в формате 1AAAALL) содержимого регистра X при выполнении директивы ⇅ должен быть кратным 7. Если это условие не выполняется, то контроллер ВЗУ ав-

Таблица 4.7

Такт СП	Адрес МК	Обработка данных	Такт СП	Адрес МК	Обработка данных
АСП=01			АСП=03		
A1	2B	S:=C	A1	10	
A2	36	R1[A2]:=R2[A2]+S+1	A2	10	
A3	11	т:=S; S:=R3[A3]; R3[A3]:=т	A3	10	
B1	28	S1:=0	B1	1B	R3[B1]:=S1+1; L:=П
B2	37	R1[B2]:=R2[B2]+S	B2	0B	R2[B2]:=B
B3	30	S:=8	B3	10	
C1	10		C1	10	
C2	01	S:=S; R2[C2]:=1	C2	03	R2[C2]:=3
C3	2D	S:=4	C3	2D	S:=4
АСП=05			АПС=06		
A1	13	S1:=7	A1	27	R1[A1]:=R1[A1]+1
A2	25	R1[A2]:=R2[A2]; S:=S1	A2	10	
A3	1E	S:=4; L:=0	A3	10	
B1	14	S1:=S1+F; L:=П	B1	10	
B2	22	R1[B2]:=$\overline{R2[B2]}$; L:=0	B2	10	
B3	19	т:=S; S:=S1; S1:=т	B3	0F	R3[B3]:=F; S1=F
C1	1E	S:=4; L:=0	C1	10	
C2	05	R2[C2]:=5	C2	06	R2[C2]:=6
C3	0F	R3[C3]:=F; S1:=F	C3	1E	S:=4; L:=0
АСП=0A			АСП=0B		
A1	00	R1[A1]:=0	A1	10	
A2	38	R2[A2]:=R2[A2+1]	A2	28	S1:=0
A3	00	R3[A3]:=0	A3	19	т:=S; S:=S1; S1:=т
B1	00	R1[B1]:=0	B1	02	R1[B1]:=2; S1:=2
B2	38	R2[B2]:=R2[B2+1]	B2	39	S:=M2[B2]+S+1
B3	00	R3[B3]:=0	B3	0B	R3[B3]:=B; S1:=B
C1	2D	S:=4	C1	19	т:=S; S:=S1; S1:=т
C2	0A	R2[C2]:=A	C2	1A	R2[C2]:=S; S:=:=M2[C2]+S+1
C3	10		C3	3C	(S:=S+1; L:=П)×L$\vee$(R3[C3]:=4; S:=:=4)×$\overline{L}$

Окончание табл. 4.7

Такт СП	Адрес МК	Обработка данных	Такт СП	Адрес МК	Обработка данных
АСП=0C			АСП=0D		
A1	1C	R1 [A] : =R1 [A]+F+L	A1	17	R1 [A] : =R1[A]+$\overline{L}$
A2	30	S : =8	A2	10	
A3	29	S : =R3 [A]+S+1	A3	16	R3 [A] : =F; S : =F
B1	2A	R1 [B] : =$\overline{L}$	B1	2A	R1 [B] : =$\overline{L}$
B2	39	S : =M2 [B]+S+1	B2	18	S : =M2 [B]
B3	3C	(S : =S+1; L : =П)× ×L$\vee$(R3 [B] : =4; S : = : =4)×$\overline{L}$	B3	20	S1 : =4; L : =0
C1	2B	S : =C	C1	19	т : =S; S : =S1; S1 : =т
C2	1A	R2 [C]:=S; S:=M2[C]+S+1	C2	0D	R2 [C] : =D
C3	3C	(S : =S+1; L : =П)× ×L$\vee$(R3[C] : =4; S : = : =4)×L	C3	15	R3 [C] : =F; L : =0
АСП=0E			АСП=08		
A1	3F	(R1 [A] : =S1+1; L : =П)× ×L$\vee$ (R1 [A] : =R1 [A]+F; L : =П)×$\overline{L}$	A1	3D	R1 [A] : =(SR1 [A])× ×L$\vee$ (S+1)×$\overline{L}$; S : = :=(S+1)×$\overline{L}$; S1:=H×L
A2	3D	(R2 [A] : =SR2 [A]; S1 : = : =H)×L$\vee$(R2 [A] : =S+ +1; S : =S+1)×$\overline{L}$	A2	26	R3 [C] : =R2 [A]; R2 [A] : =R2 [B]
A3	1B	R2 [A] : =S1+1; L : =П	A3	19	R3 [A] : =F; S : =F
B1	28	S1 : =0	B1	3D	R1 [B] : =(SR1 [B])×L$\vee$ $\vee$ (S+1)×$\overline{L}$; S : = :=(S+1)×$\overline{L}$; S1:=H×L
B2	3D	(R2 [B] : =SR2 [B]; S1 : = : =H)×L$\vee$(R2 [B] : = : =S+1; S : =S+1)×$\overline{L}$	B2	1E	S : =4; L : =0
B3	1B	R2 [B] : =S1+1; L : =П	B3	2E	S1 : =$\overline{(R3\,[B]\vee 4)}$+6
C1	19	т : =S; S : =S1; S1 : =т	C1	10	
C2	0E	R2 [C] : =E	C2	08	R2 [C] : =8
C3	1E	S : =4; L : =0	C3	1E	S : =4; L : =0

томатически увеличивает значение LL до величины, соответствующей целому числу страниц. Это необходимо учитывать при распределении памяти ВЗУ.

Текст драйвера пульта управления и программы лексического разбора приведен в табл. 4.8. Драйвер индикаторного устройства обращается к командам с адресами 33, 35, F8, FA для отображения информации на индикаторе. Эти команды являются завершающими при выво-

Адрес команды	АСП К[3]	АСП К[2]	АСП К[1]	МОД К[0]	Обработка данных	Определение адреса перехода
33	5B	5C	1	1	Отображение содержимого R1(Д12:Д1) на индикаторе. Ожидание отпускания клавиши	Если T=0, то PC:=:=PC+2
F8	5B	5C	1	1		
35	5D	5E	0	0	Отображение содержимого R1(Д12:Д1) на индикаторе	Если K1∧K2=0, то (если K1∨K2=0, то R1(Д14):=R1(Д14)+9), иначе (если K1=0, то R1(Д14):=R1(Д14)+1, иначе R1(Д14):R1(Д14)+8)
FA	5D	5E	0	0	Реакция на нажатие клавиши	
45	20	21	79	0	R3(Д10):=R3(Д10)+S1; R3(Д11):=R3(Д11)+1; S1:=0	PC:=K(1)+12=8B
B5	62	6B	71	1	S:=R3(Д10)+$\overline{R3(Д1)}$; S:=S+B; L:=П; S1:=0	Если L=1, то PC:=:=K(1)+12=83, иначе PC:=PC+2=B7
C5	20	21	78	0	R3(Д10):=R3(Д10)+S1; R3(Д11):=R3(Д11)+1; S1:=0	PC:=K(1)+12=8A
B7	20	21	33	0	То же	PC:=K(1)+12=45
83	14	20	33	1	S1:=5; L:=1	PC:=K(1)+12=45
0A	14	20	A5	1	То же	PC:=K(1)+12=B7
7A	62	7D	0D	1	S:=R3(Д10)+$\overline{R3(Д1)}$; S:=S+A; L:=П; S1:=0	PC:=PC+L+1
8A	20	21	53	0	R3(Д10):=R3(Д10)+S1; R3(Д11):=R3(Д11)+1; S1:=0	PC:=K(1)+12=65
65	20	21	69	0	То же	PC:=K(1)+12=7B
7B	20	21	A5	0	»	PC:=K(1)+12=7B
7C	72	40	5	1	т:=S; S:=S1; S1:=т; L:=1	R1(Д13):=S1
76	73	53	D0	0	R3(Д8:Д1)=11111111; R3(Д9):=0; L:=1	PC:=K(1)+12=E2
77	5C	20	5B	0	R3(Д1):=F; L:=1; R3(Д8:Д2):=0	PC:=K(1)+12=6D
78	62	20	6D	0	т:=R3(Д1); R3(Д1):=:=R3(Д1); R3(Д2):=т; L:=1	PC:=K(1)+12=7F
79	C	20	33	1	S1:=6; L:=1	PC:=K(1)+12=45
8B	62	55	36	0	S1:=R3(Д10)+$\overline{R3(Д1)}$; т:=R3(Д1); R3(Д1):=:=R3(Д2); R3(Д2):=т; L:=1	PC:=K(1)+12=48
48	40	11	AF	1	R3(Д11)+C; L:=П	Если L=1, то PC:=:=K(1)+12=B1, иначе PC:=PC+2=4A

Адрес команды	АСП К [3]	АСП К [2]	АСП К [1]	МОД К [0]	Обработка данных	Определение адреса перехода
4A	4C	32	39	0	R3(Д8):=R3(Д8)+F; L:=П; т:=R3(Д11); R3(Д11):=R3(Д12); R3(Д12):=т	Если L=1, то PC:= :=K(1)+12=4B, иначе PC:=PC+2=4C
4B	4C	32	B	0	То же	PC:=PC+1+L
4C	6D	32	8	0	R3(Д8):=0; т:=R3(Д11); R3(Д11):= :=R3(Д12); R3(Д12):=т	PC:=PC+2=4E
4D	4C	40	D	0	R3(Д8):=R3(Д8)+F; L:=П; S:=6	R1(Д13):=R1(Д13)+ +S×L+1=4×L∨E× ×$\bar{L}$; R1(Д14):=R1 (Д14)+1=5
54	7E	4C	A5	0	R3(Д2):=R3(Д1); R3(Д1):=0; S1:=7; R3(Д11):=R3(Д11)+F; L:=П; R3(Д8):=0	Если L=1, то PC:= =K(1)+12=B7, иначе PC:=PC+2=56
56	0	69	53	1	R3(Д10)+D; L:=П; S1:=E	Если L=1, то PC:= :=PC+12=65, иначе PC:=PC+2=58
58	62	52	3C	0	т:=R3(Д1); R3(Д1):= :=R3(Д2); R3(Д2):=т; R3(Д10):=R3(Д10)+ +4; R3(Д11):=5; L:=1	PC:=PC+12=4E
4E	40	4D	85	0	R1(Д12:Д11):= :=R3(Д12:Д11); L:=1; R3(Д11):=0	PC:=K(1)+12=97
97	78	40	51	1	R3(Д1)+8; L:=П	Если L=1, то PC:= :=K(1)+12=63, иначе PC:=PC+2=99
99	6	22	62	1	R3(Д9)+F; L:=П; S1:=R1(Д11)	Если L=1, то PC:= :=K(1)+12=74, иначе PC:=PC+2=9B
74	40	F	74	0	R2(Д12:Д11):= =R3(Д12:Д11); L:=1	PC:=K(1)+12=86
86	40	53	C1	0	R3(Д9):=0; L:=1	PC:=K(1)+12=Д3
B1	40	78	B	0	R3(Д9):=0; R3(Д10)+ +8; L:=П	Если L=1, то PC:= :=PC+2=B3, иначе PC:=PC+1=B2
B2	E	40	7C	1	R3(Д8)+B; L:=П; S1:=7	Если L=1, то PC:= :=K(1)+12=8E, иначе PC:=PC+2=B4
B3	6D	6F	5	0	R3(Д8):=0; S1:= :=R3(Д10); R3(Д12):= :=R3(Д10)	R1(Д13):=S1
B4	40	E	47	1	R3(Д11)+B; L:=П; S1:=7	Если L=1, то PC:= :=K(1)+12=59, иначе PC:=PC+2=B6
B6	20	21	6B	0	R3(Д10):=R3(Д10)+S1; L:=1; R3(Д11):= :=R3(Д11)+1	PC:=K(1)+12=7D

Адрес команды	АСП К [3]	АСП К [2]	АСП К [1]	МОД К [0]	Обработка данных	Определение адреса перехода
B8	4C	20	6D	0	R3(Д8):=R3(Д8)+F; L:=1	PC:=K(1)+12=7F
B9	62	73	E6	0	т:=R3(Д1); L:=1; R3(Д1):=R3(Д2); R3(Д2):=т	PC:=K(1)+12=F8
7D	13	40	49	1	R3(Д1)+B; L'=П	Если L=1, от PC:= :=K(1)+12=5B, иначе PC:=PC+2=7F
7F	62	73	21	0	т:=R3(Д1); L:=1; R3(Д1):=R3(Д2); R3(Д2):=т	PC:=K(1)+12=33
8E	20	21	4C	0	R3(Д10):=R3(Д10)+S1; R3(Д11):=R3(Д11)+1; L:=1	PC:=K(1)+12=5E
59	6D	71	5	0	R3(Д8):=0; S1:=R3 (Д10)$\vee$A	R1(Д13):=S1
5A	13	40	3C	1	R3(Д1)+B; L:=П	Если L=1, то PC:= :=K(1)+12=4E, иначе PC:=PC+2=5C
5B	70	40	10	0	R3(Д8):=2	PC:=PC—1=5A
5E	40	6F	10	0	R3(Д12):=R3(Д10)	PC:=PC—1=5D
5D	70	20	A6	0	R3(Д8):=2; L:=1	PC:=K(1)+12=B8
5F	40	6F	5	1	S1:=R3(Д10)	R1(Д13):=S1
55	40	4E	F4	0	R1(Д12:Д10):=FFF; R3(Д10):=0; L:=1	PC:=K(1)+12=06
57	E	58	C	0	R2(Д12:Д11):= :=R2(Д12:Д11)—2; десятичное вычитание	PC:=PC—2=55
5C	6D	6F	5	0	R3(Д8):=0; S1:= :=R3(Д10); R3(Д12):= :=R3(Д10)	R1(Д13):=S1
50	4C	40	9	0	R3(Д8):=R3(Д8)+F; L:=1	PC:=PC+3=53
51	70	77	6D	0	R3(Д8):=2;R3(Д9):=F; R3(Д12):=0; L:=1	PC:=K(1)+12=7F

де содержимого регистра индикации РИ (11:0) = R1(Д12:Д1) на индикатор и вместе с тем обеспечивают проверку состояния клавиатуры и формирование кодов нажимаемых клавиш. Адреса синхропрограмм обработки адреса на этих командах имеют значения, удовлетворяющие условию $0 \leqslant K(1) < 4$, а следовательно, синхронно с текущим разрядным сигналом Д*k* производится обращение содержимого регистра R1 (Д*k*) к таблице символов и вывод сегментных сигналов I1, ..., I7. Сигнал I8 для отображения десятичной запятой может быть выведен в любом разряде мантиссы в зависимости от порядка числа. Для этого используется ячейка П3 = R3 (Д1), содержимое которой определяется из условия

если $0 \leqslant PX(11:9) < 8$, то R3 (Д1) := PX (9) + 8, иначе R3 (Д1):= := PX (9).

Все команды драйвера пульта управления проверяют состояние клавиатуры, но команды с адресами 35, FA позволяют определить, какая клавиша нажата, а остальные — нажата ли какая-нибудь клавиша или нет. К команде с адресом 33 происходит обращение при выполнении программы, инициализирующей работу ПМК после включения напряжения питания и после выполнения операторов или директив. К команде с адресом F8 происходит обращение только после нажатия клавиши F. На этих командах в области мантиссы и порядка выполняются синхропрограммы, которые приведены в табл. 4.9. Если ни одна из нажатых клавиш не коммутирует выход сигнала Д2 на входы К1 или К2, то T := 0, при нажатии клавиши T:=1. Рассмотрим работу синхропрограммы с АСП = 5 В, если порядок числа в регистре X равен 6 и, следовательно, R3 (Д1) = E.

Таблица 4.9

Такт СП	Адрес МК	Обработка данных	Такт СП	Адрес МК	Обработка данных
АСП=5B			АСП=5C		
A1	0E	S: =F	A1	00	
A2	1E	S: =$\overline{S}$	A2	0E	S: =F
A3	15	S: =R3[A]+S+1; L: =П	A3	08	т: =S; S: =R3[A]; R3[A]: =т
B1	00		B1	0E	S: =F
B2	00		B2	1D	S1: =0
B3	02	S: =S+1; L: =П	B3	23	R3[B]: =S1
C1	00		C1	1E	S: =$\overline{S}$
C2	00		C2	3A	S: =S+$\overline{T}$; L: =П
C3	02	S: =S+1; L: =П	C3	3A	S:=S+$\overline{T}$; L: =П
АСП=5D			АСП=5E		
A1	1D	S1: =0	A1	00	
A2	04	S: =0; L: =1	A2	00	
A3	15	S: =R3[A]+S+1; L: =П	A3	3A	S: =S+$\overline{T}$; L: =П
B1	00		B1	00	
B2	00		B2	0D	R3[B—1]: =S
B3	3A	S: =S+$\overline{T}$; L: =П	B3	0E	S: =F
C1	00		C1	03	R3[C—1]: =S
C2	00		C2	0F	т: =S; S: =S1; S1: =т
C3	3A	S: =S+$\overline{T}$; L: =П	C3	00	

По временному адресу Д1Е3 выполнится присвоеиие S := R3(Д1)+ +1 = F; L := 0. Во временном интервале Д2 на индикатор будут выведены сигналы I1 ... I7, соответствующие содержимому R1 (Д1). Поскольку L :=0, то сигнал I8 будет единичного уровня, который не отображает сегмент дисплея.

По временному адресу Д2Е3 выполнится присвоение S := S + 1 = = F + 1 = 0; L := 1. Во временном интервале Д3 вместе с сигналами I1 ... I7, соответствующими содержимому R1 (Д2), будет выведен сигнал I8 уровня логического 0, который будет индицировать запятую.

По временному адресу Д3Е3 выполнится присвоение S := S + 1 = = 1; L := 0 и с начала временного интервала Д4 сигнал I8 уровня логической 1 выключает индикацию сегмента. Он будет выключен до конца исполнения команды, так как L не изменит своего значения. По временному адресу Д9Е3 выполнится присвоение S := S + 1 = 7; L := 0. При исполнении синхропрограммы с АСП = 5С проверяется состояние разряда Т. Если Т = 0, то S := 2, иначе S := 0. Значение L = 0, и не произойдет ложного подсвета сегмента на индикаторе. Синхропрограмма обработки адреса использует содержимое аккумулятора S для вычисления адреса следующей команды. Если ни одна клавиша не нажата (Т := 0), то содержимое счетчика команд уменьшится на 2. Если перед исполнением этой команды или во время ее исполнения была нажата любая клавиша, то команда будет исполняться до ее отпускания, так как при Т := 1 содержимое РС не будет изменяться.

В табл. 4.9 приведены синхропрограммы, которые выполняются на командах с адресами 35, FA в области мантиссы и порядка содержимого регистров памяти. При выполнении синхропрограммы с АСП =5D отличие от предыдущих команд состоит в том, что накопление содержимого аккумулятора S с изменением содержимого ячейки L происходит не безусловно, а в зависимости от состояния Т. Если ни одиа клавиша на клавиатуре не нажата, то изменение содержимого S и L во времени будет совпадать с их измеиением на предыдущих командах и так же будет осуществляться вывод сегментных сигналов I1, ..., I8. Если какая-нибудь клавиша нажата, то соответствующий разрядный сигнал будет передан на вход К1 или К2 и в зависимости от этого изменит содержимое S1. Кроме того, начиная с момента Д*k* перестанет изменяться содержимое аккумулятора S. Это условие позволяет определить, какой разрядный сигнал Д*k* был подан на входы микроЭВМ, и идентифицировать на клавиатуре столбец, в котором была нажата клавиша. Поскольку матрица клавиатуры содержит 10 столбцов с разрядными сигналами Д2, ..., Д11, то анализ состояния клавиатуры продолжается и в течение интервала Д10 при исполнении синхропрограммы обработки области порядка. Содержимое аккумулятора пересылается в R3 (Д10), S := S1, R3 (Д11) := F. Для получения младшего разряда кода клавиши выполняется вычитание: R3 (Д10) — — R3 (Д1) — 1. Содержимое S, куда пересылается содержимое из S1, используется для вычисления адреса следующей команды. Если какая-нибудь клавиша нажата, то адрес перехода будет вычислен как

сумма старшего разряда содержимого счетчика команд и старшего разряда кода клавиши (табл. 4.2). Если ни одна из клавиш не нажата, то содержимое РС не изменится и будет продолжаться индикация содержимого РИ.

На командах с адресами 0А, 45, 58, 65, 7А, 7В, 83, 8А, 8В, В7, С5 происходит формирование кодов операторов за один или несколько проходов. На команде с адресом 48 происходит выделение кодов символов, получаемых при нажатии в строке матрицы клавиатуры клавиш, которые коммутируют разрядные сигналы на входы К1 и К2 БИС ИК1302. Эти коды адресуются команде В1, где из них выделяются коды символов F и К, которые идентифицируются на команде В3. Для символа F переход производится на команду с адресом В9, а для К — на команду с адресом В8. Остальные коды символов из этой строки клавиатуры передаются на команду с адресом В2. Здесь проверяется, введен ли символ К перед текущим, т. е. формируется оператор с косвенной адресацией или нет. В первом случае управление передается команде с адресом 8Е. На этой ветви программы производится формирование старшей тетрады кода оператора с косвенной адресацией и осуществляется переход к опросу клавиатуры для получения адреса регистра, который записывается в младшую тетраду кода оператора. Если символ К перед текущим не вводился, то выполняется переход к команде с адресом В4. По этой команде проверяется, введен ли символ F перед текущим. Если введен, выполняется переход к команде с адресом В6, где формируются коды таких операторов, а далее на команде с адресом 7D проверяется режим, в котором находится ПМК. В режиме программирования осуществляется ввод сформированного кода в программную память и подготовка к вводу адреса прямого перехода.

В режиме исполнения выполнение этого оператора блокируется и управление передается драйверу пульта управления с индикацией прежнего содержимого РИ.

Для текущего символа, которому не предшествовал ввод символа F, выполняется переход к команде с адресом 59, по которой выполняется ветвление по четырем направлениям: для символов П → х и х → → П — на команду с адресом 5Е, для символов БП и ПП—5В, В/О и С/П—5А, $\overrightarrow{\text{ШГ}}$ и $\overleftarrow{\text{ШГ}}$—5F. Для первого направления ветвления выполняется формирование старшей тетрады кода оператора, для последнего — окончательная идентификация и выполнение директив по изменению содержимого программного счетчика. По команде с адресом 5А проверяется режим ПМК. В режиме программирования сформированный код оператора или слова операторов БП ЕА, ПП ЕА записывается в программную память по текущему содержимому РС. В режиме исполнения на команде с адресом 5С производится идентификация директив, составленных из этих символов. Для директивы БП *hd* переход выполняется на команду с адресом 51, для директивы С/П — на команду 50, для директивы ПП — 53, для директивы В/О — на

команду 52. По команде с адресом 7С производится ветвление на команду с адресом 79 при выполнении оператора FBx и на команды с адресом 78, 77 или 76 при выполнении директив FCF, FПРГ или FABT.

После выполнения проверки по команде с адресом 48 строки матрицы клавиатуры, где была нажата клавиша, основная группа символов адресуется к команде 4А. Если код оператора формируется из текущего кода символа, то выполняется переход на команду с адресом 4С, которая восстанавливает значение текущего кода оператора в старшей тетраде R3 (Д11), измененное на команде 4А.

Если требуется получить младшую тетраду кода оператора с косвенной адресацией или младший разряд абсолютного адреса перехода ЕА, то к младшей тетраде текущего кода R3(Д10) на команде 4С присоединяется значение старшей тетрады R3(Д11):=R3(Д12), полученное на предыдущем шаге, и выполняется переход к команде с адресом 4Е. Для получения значения старшего разряда абсолютного адреса перехода ЕА в операторах прямых программных ветвлений с команды с адресом 4D выполняется переход на команду с адресом 5Е, где производится хранение младшей тетрады текущего кода символа: R3(Д12):= :=R3(Д10).

По командам с адресами 54, 56, 58 производится анализ текущего символа и начальный этап формирования кода оператора, начинающегося с символа К и не являющегося оператором с косвенной адресацией. Сформированный код оператора по команде с адресом 4Е пересылается в ячейку ОП:=R1(Д12:Д11). По командам с адресами 94, 86 происходит изменение содержимого программного счетчика РС = = R2(Д12:Д11) при выполнении операторов программных ветвлений или директивы БП *hd*. По команде с адресом 97 анализируется режим, сформированный код оператора передается для исполнения или запоминания в программной памяти по текущему содержимому РС.

В качестве примера рассмотрим трассу адресов при формировании оператора Кх → ПС в режиме исполнения и оператора FL1A2 в режиме программирования. В обоих случаях полагаем, что ПМК завершил выполнение предыдущего оператора или директивы и находится в состоянии анализа клавиатуры на команде с адресом 35. При нажатии клавиши К выполняются команды с адресами С5, 8А, 65, 7В, В7, 45, 8В, 48, В1, В3, В8, 7F, 33. До отпускания клавиши будет выполняться последняя команда, после чего — исходная команда 35. При нажатии клавиши х → П выполняются команды С5, 8А, 65, 7В, В7, 45, 8В, 48, В1, В2, 8Е, 5Е, 5D, В8, 7F, 33, а после отпускания клавиши — команда 35. Старшая тетрада кода формируемого оператора будет записана в R3(Д12):=В. После нажатия клавиши С выполняются команды с адресами В5, 83, 45, 8В, 48, 4А, 4В, 4С, 4Е, 97, 99 и далее на исполнение сформированного кода оператора ОП:=ВС.

Для ввода оператора F L1 A2 в программную память нажимается клавиша F, после чего выполняется следующая последовательность команд: С5, 8А, 65, 7В, В7, 45, 8В, 48, В1, В3, В9, F8 и после отпускания клавиши — команда FA. После нажатия клавиши L1 выполня-

ются команды с адресами 8A, 95, 7B, B7, 45, 8B, 48, B1, B2, B4, B6, 7D, 5B, 5A, 4E, 97 и запись сформированного кода слова оператора ОП:= :=5B в программную память по текущему содержимому РС. После записи индикация текущей области программной памяти будет выполняться по команде с адресом 35. После нажатия клавиши А выполняются команды с адресами B5, 83, 45, 8B, 48, 4A, 4B, 4D, 5E, 5D, B8, 7F, 33 и после отпускания клавиши — команда 35. Старший разряд адреса перехода будет записан в R3(Д12) = A. После нажатия клавиши 2 выполняются команды с адресами 45, 8B, 48, 4A, 4B, 4C, 4E, 97 и запись сформированного второго слова оператора ОП:=A2, которое является адресом перехода в память Мр по адресу РС.

Фрагмент программы интерпретатора входного языка ПМК, записанный в ПЗУ БИС ИК1303 и обеспечивающий выполнение операторов «+» и «—», приведен в табл. 4.10. При обращении к этому фрагменту содержимое регистра X скопировано в R2(Д12:Д1), а регистра Y — в R1(Д12:Д1).

При выполнении оператора «+» после декодирования его кода происходит обращение к команде с адресом 0D, на которой счетчику в S1 присваивается значение 3. При выполнении оператора «—» происходит обращение к команде с адресом 1D, изменяющей на обратное значение знакового разряда в копии содержимого регистра X и переход на команду с адресом 0D фрагмента программы, обеспечивающего интерпретацию оператора «+». По команде FC проверяется значение знакового разряда R2(Д9), и в случае его отрицательного значения (код 9) переводится содержимое мантиссы R2 (Д9:Д1) в дополнительный код. На команде с адресом FD выполняется обмен содержимым между блоками регистра сверхоперативной памяти, где хранятся мантиссы операндов. По этой команде производится вычитание 2 из содержимого ячейки S1, выполняющей в данном случае функции счетчика; при первом выполнении команды с адресом FD происходит переход на команду FC(L = 1). Этим обеспечивается проверка знакового разряда и перевод в дополнительный код мантиссы второго операнда. При втором выполнении команды FC(L = 0) происходит переход на команду с адресом FF, причем операнды находятся в тех же ячейках регистра R, в каких они хранились перед обращением к этому фрагменту программы десятичного суммирования операндов. По команде с адресом FF выполняется сравнение порядков операндов. Если порядок операнда, находящегося в R2, будет больше или равен порядку операнда, записанного в R1, то выполняется переход на команду с адресом F0. На этой команде операнды поменяются местами и произойдет безусловный переход к команде с адресом 2F, выполняющей такое же сравнение порядков операндов, как и команда с адресом FF. Она является завершающей в процедуре уравнивания порядков операндов. При равенстве порядков операндов осуществляется переход к команде с адресом 21, в противном случае — к команде F1. Ветвление на эту команду производится и после первого сравнения порядков слагаемых при R2(Д12:Д10)<R1(Д12:Д10).

Таблица 4.10

Адрес команды	АСП К [3]	АСП К [2]	АСП К [1]	МОД К [0]	Обработка данных	Определение адреса перехода
1D	76	50	13	0	R2 (Д9) : = R2 (Д9) + A; L : = 0	R1 (Д14) : = R1 (Д14) + + F = 0
0D	50	52	FB	1	S1 : = 3; L : = 1	PC : = K (1) + 1 = FC
FC	6A	50	17	0	R2 (Д9) + F; L : = П; R2 (Д9) : = 0	PC : = PC + 1 + L
FE	47	3C	A	0	R2 (Д2:Д1) : = — R2 (Д9 : : Д1)	R1 (Д13) : = R1 (Д13) + + F = D
FD	20	50	FB	0	т : = R1 (Д9 : Д1); R1 (Д9 : Д1) : = R2 (Д9 : : Д1); S1 : = S1 + F; R2 (Д9 : Д1) : = т; S1 : = : = S1 + F; L : = П	Если L = 1, то PC : = : = K (1) + 1 = FC, иначе PC : = PC + 2 = FF
FF	50	4D	17	0	R2 (Д11 : Д10) : = R1 (Д11 : Д10 : L : = П; S : = : = R2 (Д12) + R1 (Д12) + L S : = S + 8; L : = П	Если L = 1, то R1 (Д13) : = : = R1 (Д13) + 2 = 1, иначе R1 (Д13) : = R1 (Д13) + 1 = 0
F0	20	20	2E	0	т : = R1 (Д12 : Д1); R1 (Д12 : Д1) : = R2 (Д12 : : Д1); R2 (Д12 : Д1) : = т; L : = 1	PC : = K (1) + 1 = 2F
F1	48	50	20	1	R1 (Д9 : Д1) — R1 (Д9 : Д1); L : = П; S1 : = R1 (Д9)	Если L = 1, то PC : = : = K (1) + 1 = 21, иначе PC : = PC + 2 = F3
F3	4C	60	2D	0	R2 (Д9 : Д1) : = R2 (Д9 : : Д1) + 5; L : = 1, десятичное сложение	PC : = K (1) + 1 = 2E
2E	6C	3D	9	0	Для i = 1 до 8 (R2 (Дi) : = : = R2 (Дi + 1); S1 : = S1 + + 8; S1 : = S1 + 8; L : = : = П; R2 (Д9) : = 9 × $\overline{L}$; R2 (Д12 : Д10) : = R2 (Д12 : Д10) + 1	RC : = PC + 1 = 2F
2F	50	4D	F0	0	R2 (Д11 : Д10) : = R1 (Д11 : : Д10); L : = П; S : = : = R2 (Д12) + $\overline{\text{R1 (Д12)}}$ + + L; S : = S + 8; L : = П	Если L = 1, то PC : = : = K (1) + 1 = F1, иначе R1 (Д13) : = 1
21	2E	D	31	0	R2 (Д9 : Д1) : = R2 (Д9 : Д1) + R1 (Д9 : Д1); S1 : = R2 (Д9); S1 : = S1 + + 8; S1 : = S1 + 8; L : = П; R3 (Д9) : = 0	Если L = 1, то PC : = : = K (1) + 1 = 32, иначе PC : = PC + 2 = 23
23	47	6B	31	0	R2 (Д9 : Д1) : = — R2 (Д9 : Д1); R3 (Д9) : = 9; L : = 1	PC : = K (1) + 1 = 32
32	47	50	B4	1	R2 (Д9 : Д1) — R2 (Д9 : Д1); L : = П	Если L = 1, то PC : = : = K (1) + 1 = 75, иначе PC : = PC + 2 = 34
34	6A	2B	AA	1	R2 (Д9) + F; L : = П	Если L = 1, то PC : = : = K (1) + 1 = AB, иначе PC : = PC + 2 = 36

Окончание табл. 4.10

Адрес команды	АСП К [3]	АСП К [2]	АСП К [1]	МОД К [0]	Обработка данных	Определение адреса перехода
36	33	49	1D	0	Для $i=1$ до 8 (R2 (10—i) : = R2 (9—i); R2 (Д12 : Д10) : = R2 (Д12 : Д1)—1; R2 (Д1): = : = 0)	PC : = PC—2 = 34
B5	4B	26	BA	0	R3 (Д9) : = 0; L : = 1; R3 (Д11 : Д10) : = R2(Д11: : Д10) R2 (Д12 : Д10) : = 0	PC : = K (1) + 1 = BB
AB	4C	3D	0D	0	R2(Д9 : Д1): = : = R2 (Д9:Д1) +5; R2 (Д12 :Д10): = R2(Д12 : : Д10)	PC : = PC + 10 = BB

Таким образом, если порядки операндов не равны, то меньший из операндов будет всегда находиться в R2. На команде с адресом F1 выполняется проверка мантиссы слагаемого, находящегося в R1. Если мантисса операнда равна нулю, то производится выход из процедуры уравнивания порядков, а если иначе, то переход на команду с адресом F3. Эта проверка позволяет исключить искажение результата при суммировании нулевого значения с операндом, абсолютное значение которого меньше единицы. На команде с адресом F3 выполняется округление значения старших семи разрядов мантиссы в зависимости от содержимого ее младшего разряда. По команде с адресом 2E выполняется денормализация операнда, хранимого в R2, т. е. сдвиг мантиссы на один десятичный разряд вправо с расширением знака и увеличение порядка на единицу. После этого производится переход на команду 2F и по команде с адресом 21 выполняется суммирование мантисс слагаемых с равными порядками. Если результат получается отрицательным, то по команде с адресом 23 он преобразуется из дополнительного кода в прямой с запоминанием знака мантиссы в R3(Д9).

При выполнении операции суммирования слагаемых возможны переполнение разрядной сетки ПМК в области мантиссы и порядка или ненормализованный результат при нормализованных исходных операндах. При переполнении в области мантиссы старший разряд результата не теряется, и необходимо только выполнить его денормализацию. Если результат ненормализованный, то необходимо выполнять его нормализацию.

Перед выполнением любой из этих процедур по команде с адресом 32 выполняется проверка мантиссы результата на равенство нулю, чтобы исключить бесконечный цикл на командах с адресами 34, 36. По этим командам выполняется нормализация результата, а при первом вхождении по команде 34 производится проверка переполнения

разрядной сетки и в области мантиссы. Нормализация мантиссы результата заключается в повторении операции сдвига мантиссы на один десятичный разряд с обнулением младшего разряда, а также в уменьшении порядка на единицу до тех пор, пока в знаковом разряде R2(Д9) не будет находиться наибольшая значащая цифра мантиссы. После выполнения этого условия по команде с адресом AB выполняется округление мантиссы и увеличение порядка результата на единицу, а по команде BB — сдвиг мантиссы вправо на один десятичный разряд с присвоением знаковому разряду его значения, ранее сохраненного в R3(Д9) (денормализация результата с округлением значения мантиссы). При переполнении разрядной сетки в области мантиссы выполнение этой процедуры может привести к переполнению разрядной сетки в области порядка.

Приводимый фрагмент программы завершается командой с адресом A7, по которой выполняется копия результата в R1 и переход по адресу, ранее записанному в стек SPC.

4.5. УРОВЕНЬ МИКРОКОМАНД

Синхропрограмма определяет последовательность адресов микрокоманд, выполняемых в течение временного интервала E и реализуемых параллельным формированием напряжений уровня логического 0 или 1, управляющих логическими элементами операционных устройств микроЭВМ. Микрокоманды делятся на *условные* и *безусловные*. Адрес условной микрокоманды содержит разряд L слова состояния процесса, в зависимости от содержимого которого выполняется одна из двух микрокоманд. Это обеспечивает гибкость системы микрокоманд без увеличения ее объема.

Сокращенная запись микрокоманды отображает ее назначение без излишней детализации. Запись L:=П означает, что разряду присвоено значение переноса после обработки старшего разряда тетрады. Запись $A\overline{L}$ означает обращение к шестнадцатеричной константе A только при L = 0. Запись S:=S+L соответствует добавлению единицы к содержимому аккумулятора только при L = 1, а запись S:= :=S + $\overline{T}$ — только при T:=0.

В микроЭВМ К745ИК13 может храниться 68 микрокоманд, в том числе 60 безусловных с адресами от 00 до 3 B и 4 условных с адресами от 3C до 3F. В ПЗУ микроЭВМ К745ИК18 хранится 40 микрокоманд, в том числе 24 безусловных с адресами от 00 до 17 и 8 условных с адресами от 18 до 1F.

Микрокоманды микроЭВМ К745ИК1302 в основном обеспечивают пересылку данных и управление работой пульта управления и индикаторного устройства. По назначению эти микрокоманды (табл. 4.11) можно отнести к одной из групп:

1. Микрокоманды с адресами 07, 24, 09, 0E, 01, 21, 1E, 1D, 2B, 04, 0F загружают в аккумулятор S и ячейку S1 ряд констант, прямое и ин-

Таблица 4.11

АДРЕС МК	Описание МК	Сокращенная запись МК
0	Нет операции	
01	$\Sigma := R[I]$; $S := \Sigma$	$S := R[I]$
02	$\Sigma := S+1$; $L := \Pi$; $S := \Sigma$	$S := S+1$; $L := \Pi$
03	$\Sigma := S$; $R[I-1] := \Sigma$	$R[I-1] := S$
04	$\Sigma := (S \vee \overline{S}) + (L := \overline{L})$; $L := \Pi$; $S := \Sigma$	$S := 0$; $L := 1$
05	$\Sigma := R[I]+S+(L \vee \overline{L})$; $L := \Pi$	$R[I]+S+1$; $L := \Pi$
06	$\Sigma := R[I]+(S \vee \overline{S})$; $L := \Pi$; $S := \Sigma$	$S := R[I]+F$; $L := \Pi$
07	$\Sigma := 1+(L \vee \overline{L})$; $S := \Sigma$	$S := 2$
08	$\Sigma := R[I]$; $R[I] := S$; $S := \Sigma$	
09	$\Sigma := 6$; $S := \Sigma$	$S := 6$
0A	$\Sigma := (R[I] \vee \overline{R[I]})+S$; $L := \Pi$; $S := \Sigma$	$S := S+F$; $L := \Pi$
0B	$\Sigma := S+(6 \vee 1)+(L \vee \overline{L})$; $L := \Pi$; $S := \Sigma$	$S := S+8$; $L := \Pi$
0C	$\Sigma := S+1$; $S := \Sigma$	$S := S+1$
0D	$\Sigma := S$; $R[I-2] := \Sigma$	$R[I-2] := S$
0E	$\Sigma := S \vee \overline{S}$; $S := \Sigma$	$S := F$
0F	$\Sigma := S$; $S := S1$; $S1 := \Sigma$	
10	$\Sigma := R[I]+S$; $S := \Sigma$	$S := R[I]+S$
11	$\Sigma := R[I]+1$; $L := \Pi$; $R[I] := \Sigma$	$R[I] := R[I]+1$; $L := \Pi$
12	$\Sigma := 0$; $R[I-1] := \Sigma$	$R[I-1] := 0$
13	$\Sigma := R[I]$; $R[I] := S$; $R[I-1] := \Sigma$	$R[I-1] := R[I]$; $R[I] := S$
14	$\Sigma := R[I]$; $S := \Sigma$; $R[I] := R[I+3]$	$S := R[I]$; $R[I] := R[I+3]$
15	$\Sigma := R[I]+S+(L \vee \overline{L})$; $L := \Pi$; $S := \Sigma$	$S := R[I]+S+1$; $L := \Pi$
16	$\Sigma := R[I]+S+L$; $L := \Pi$; $S := \Sigma$	$S := R[I]+S+L$; $L := \Pi$
17	$\Sigma := R[I]+(S \vee \overline{S})+L$; $L := \Pi$; $S := \Sigma$	$S := R[I]+F+L$; $L := \Pi$
18	$\Sigma := S+A \times \overline{L}$; $R[I-1] := \Sigma$	$R[I-1] := S+A \times \overline{L}$
19	$\Sigma := R[I]+6$; $S := \Sigma$	$S := R[I]+6$
1A	$\Sigma := R[I]+S$; $L := \Pi$; $S := \Sigma$	$S := R[I]+S$; $L := \Pi$
1B	$\Sigma := R[I]$; $R[I-1] := \Sigma$	$R[I-1] := R[I]$
1C	$\Sigma := R[I]+1$; $S := \Sigma$	$S := R[I]+1$
1D	$\Sigma := 0$; $S1 := \Sigma$	$S1 := 0$

Адрес МК	Описание МК	Сокращенная запись МК
1E	$\Sigma := \overline{S}$; $S := \Sigma$	$S := \overline{S}$
1F	$\Sigma := R[I]+1$; $L := \Pi$; $S1 := \Sigma$	$S1 := R[I]+1$; $L := \Pi$
20	$\Sigma := R[I]+(6 \vee 1)+(L \vee \overline{L})$; $R[I] := \Sigma$	$R[I] := R[I]+8$
21	$\Sigma := \overline{R[I]}$; $S := \Sigma$	$S := \overline{R[I]}$
22	$\Sigma := \overline{R[I]}+S$; $L := \Pi$; $S := \Sigma$	$S := R[I]+S$; $L := \Pi$
23	$\Sigma := S1$; $R[I] := \Sigma$	$R[I] := S1$
24	$\Sigma := 4$; $S := \Sigma$	$S := 4$
25	$\Sigma := (\overline{S} \vee S1)+L$; $S := \Sigma$	$S := (\overline{S} \vee S1)+L$
26	$\Sigma := S1+L$; $S := \Sigma$; $S1 := \Sigma$	$S := S1+L$; $S1 := S$
27	$\Sigma := \overline{R[I]}+S1$; $S1 := \Sigma$	$S1 := \overline{R[I]}+S1$
28	$\Sigma := R[I]$; $R[I-2] := \Sigma$	$R[I-2] := R[I]$
29	$\Sigma := M[I]+S$; $L := \Pi$; $S := \Sigma$	$S := M[I]+S$; $L := \Pi$
2A	$\Sigma := \overline{R[I]}+L$; $L := \Pi$; $S := \Sigma$	$S := \overline{R[I]}+L$; $L := \Pi$
2B	$S := R[I]$; $S1 := \Sigma$	$S1 := R[I]$
2C	$\Sigma := \overline{R[I]}+1$; $L := \Pi$; $S := \Sigma$	$S := \overline{R[I]}+1$; $L := \Pi$
2D	$\Sigma := R[I]$; $M[I] := S$; $S := \Sigma$	$M[I] := S$; $S := R[I]$
2E	$\Sigma := SR[I]$; $R[I] := \Sigma$	$R[I] := SR[I]$
2F	$\Sigma := S+1$; $R[I-2] := \Sigma$	$R[I-2] := S+1$
30	$\Sigma := M[I]$; $S := \Sigma$	$S := M[I]$
31	$\Sigma := M[I]$; $R[I-1] := \Sigma$; $M[I] := S$	$R[I-1] := M[I]$; $M[I] := S$
32	$R[I] := R[I+3]$	
33	$\Sigma := S+A \times \overline{L}$; $L := \Pi$; $S := \Sigma$	$S := S+A \times \overline{L}$; $L := \Pi$
34	$\Sigma := S+S1$; $L := \Pi$; $S := \Sigma$	$S := S+S1$; $L := \Pi$
35	$\Sigma := R[I]+L$; $S := \Sigma$	$S := R[I]+L$
36	$\Sigma := S1+(L \vee \overline{L})$; $L := \Pi$; $S1 := \Sigma$	$S1 := S1+1$; $L := \Pi$
37	$\Sigma := R[I]$; $S := \Sigma$; $SR[I] := \Sigma$	$S := R[I]$; $SR[I] := S$
38	$\Sigma := R[I]+1+L$; $R[I] := \Sigma$	$R[I] := R[I]+1+L$
39	$\Sigma := R[I]$; $R[I] := R[I+3]$; $S := \Sigma$; $S1 := \Sigma$	$S := R[I]$; $S1 := S$; $R[I] := R[I+3]$
3A	$\Sigma := S+\overline{T}$; $L := \Pi$; $S := \Sigma$	$S := S+\overline{T}$; $L := \Pi$
3B	$\Sigma := R[I]+1$; $S := \Sigma$; $SR[I] := \Sigma$	$S := R[I]+1$; $SR[I] := S$
3CL	$\Sigma := R[I]+1$; $S := \Sigma$; $R[I-1] := \Sigma$	$S := R[I]+1$; $R[I-1] := S$
$3C\overline{L}$	$\Sigma := S$; $S := \Sigma$; $R[I-1] := \Sigma$	$R[I-1] := S$

Окончание табл. 4.11

Адрес МК	Описание МК	Сокращенная запись МК
3DL	$\Sigma := R[I]+S+(L \vee \overline{L})$; $R[I] := \Sigma$; $S := \Sigma$	$S := R[I]+S+1$; $R[I] := S$
$3D\overline{L}$	$\Sigma := R[I]+1$; $R[I] := \Sigma$; $S := \Sigma$	$S := R[I]+1$; $R[I] := S$
3EL	$\Sigma := S \vee \overline{S}$; $R[I] := S$; $S := \Sigma$	$S := F$; $R[I] := S$
$3E\overline{L}$	$\Sigma := S \vee \overline{S}$; $S := \Sigma$	$S := F$
3FL	$\Sigma := R[I]+S$; $L := П$; $S := \Sigma$	$S := R[I]+S$; $L := П$
$3F\overline{L}$	$\Sigma := R[I]$; $S := \Sigma$	$S := R[I]$

версное содержимое тетрады операционного регистра R, выполняют обмен содержимым S и S1, а также инвертирование содержимого аккумулятора S. Эти микрокоманды (за исключением микрокоманды 0F) не изменяют содержимого ячейки L.

2. Микрокоманды с адресами 23, 12, 03, 1B, 0D, 28, 13, 08, 3E выполняют очистку содержимого текущей и предыдущей ячеек регистра R и содержимого S и S1, не изменяя содержимого ячейки L. Микрокоманда с адресом 3E условная, загружает текущую тетраду в регистр R в зависимости от содержимого ячейки L.

3. Микрокоманды с адресами 37, 3B и 2E (обращения к стековому регистру SR) обеспечивают обмен информацией между регистрами R и SR, не изменяя содержимого ячейки L.

4. Микрокоманды с адресами 30, 29, 31 и 2D (обращения к системной магистрали) обеспечивают обмен информацией между регистром R, аккумулятором S и системной магистралью, причем команда с адресом 29 используется для непосредственного анализа информации в системной магистрали и записи в ячейку L текущего значения переноса.

5. Микрокоманда с адресами 32, 14 и 39 обеспечивают сдвиг информации из последующей тетрады в текущую в одной из ячеек регистра R без изменения содержимого ячейки L, причем содержимое текущей ячейки может быть стерто или передано в S или S1.

6. Арифметические микрокоманды (адреса 02, 0B, 0A, 34, 33, 11, 36, 06, 17, 1A, 15, 16, 22, 2C, 0C, 10, 1C, 35, 19, 27, 20, 26, 3D, 3C, 2A, 1F, 3F, 38, 18, 2F) выполняют суммирование содержимого S, S1 и текущей тетрады R с рядом безусловных констант, единицей или константой, зависящей от содержимого ячейки L. Ряд микрокоманд суммирует прямое или инверсное содержимое текущей тетрады R с S или S1 и засылает результат в текущую или предыдущую тетраду R. Некоторые из микрокоманд не изменяют содержимого ячейки L, но остальные заносят в L значение переноса. Условные микрокоманды выполняют операции в зависимости от исходного содержимого ячейки L.

7. Логическая микрокоманда с адресом 25 использует содержимое ячейки L без его изменения.

8. Микрокоманда ввода с адресом 3A фиксирует положительный фронт входного сигнала с точностью до интервала разрядного сигнала, и после ее выполнения перенос засылается в ячейку L, инверсное содержимое которой подается на выход I8 микроЭВМ.

Ввод информации по входам K1 и K2 производится по команде с полем $0 \leqslant K(1) \leqslant 3$, а синхропрограммы обработки областей мантиссы и порядка могут иметь произвольные адреса. В этом случае исходное содержимое ячейки T равно нулю, но в случае исполнения команды нулевое значение сигнала на входах K1 или K2 в течение 8 тактов приводит к занесению единицы в ячейку T.

9. Микрокоманда с адресом 05 обеспечивает перенос при суммировании содержимого текущей тетрады регистра R, аккумулятора S, константы 1 и его запись в ячейку L.

МикроЭВМ К745ИК1303 в основном обеспечивает интерпретацию функциональных операторов прикладной программы в десятичной системе счисления. По назначению микрокоманды этой микроЭВМ (табл. 4.12) разбиваются на следующие группы:

1. Микрокоманды с адресами 12, 35, 0A, 11, 24, 1F, 01, 07, 06, 05, 3F, 33, 08 загружают в S и S1 прямое и инверсное значения текущей тетрады R и констант, значения которых безусловны или зависят от содержимого ячейки L. Микрокоманда с адресом 06 присваивает ячейке L единицу, остальные микрокоманды не изменяют ее содержимого.

2. Микрокоманды с адресами 34, 3B, 20, 02, 3A, 2D, 0C, 18, 0B, 3C выполняют обмен содержимым тетрады R и S, безусловную или условную загрузку тетрады R константой или содержимым S или S1. Микрокоманда с адресом 0B выполняет кольцевую пересылку информации ($S:=S1$; $S1:=R(Дk)$; $R(Дk):=S$).

3. Микрокоманды с адресами 2C, 37, 31, 3E обеспечивают условный и безусловный ввод информации в стековый регистр из S и текущей тетрады R и вывод данных из стека в аккумулятор S.

4. Микрокоманды с адресами 22, 32 и 23 обеспечивают обмен информацией между системной магистралью и регистром R и передачу информации из системной магистрали в аккумулятор S без изменения содержимого слова состояния.

5. Микрокоманды с адресом 1C и 2F сдвигают информацию в тетрадах одного блока регистра R без изменения содержимого слова состояния.

6. Арифметические микрокоманды с адресами 10, 25, 14, 16, 04, 2B, 17, 36, 1D, 39, 15, 09, 21, 2A, 30, 26, 27, 13, 19, 29, 0D, 38, 03 обеспечивают вычисления в двоичной и десятичной системах счисления. Микрокоманда 03 выполняет десятичную коррекцию результата с записью его в регистр R вместо одного из операндов и в S1 с пересылкой прежнего содержимого S1 в S. Это позволяет анализировать содержимое двух последних разрядов мантиссы после суммирования или вычитания.

Таблица 4.12

Адрес МК	Описание МК	Сокращенная запись МК
00	Нет операции	
01	$\Sigma := R[I]$; $S := \Sigma$	$S := R[I]$
02	$\Sigma := S$; $R[I-1] := \Sigma$	$R[I-1] := S$
03	$\Sigma := S + A \times \overline{L}$; $S1 := \Sigma$; $S := S1$; $R[I-1] := \Sigma$	
04	$\Sigma := R[I] + S$; $L := \Pi$; $S := \Sigma$	$S := R[I] + S$; $L := \Pi$
05	$\Sigma := 0$; $S1 := \Sigma$	$S1 := 0$
06	$\Sigma := S$; $S := S1$; $S1 := \Sigma$	
07	$\Sigma := \overline{R[I]}$; $S := \Sigma$	$S := \overline{R[I]}$
08	$\Sigma := (S \vee \overline{S}) + (L \vee \overline{L})$; $L := \Pi$; $S := \Sigma$	$S := 0$; $L := 1$
09	$\Sigma := S1 + \overline{L}$; $S1 := \Sigma$	$S1 := S1 + \overline{L}$
0A	$\Sigma := 6$; $S := \Sigma$	$S := 6$
0B	$\Sigma := R[I]$; $R[I] := S$; $S := S1$; $S1 := \Sigma$	
0C	$\Sigma := S$; $R[I-2] := \Sigma$	$R[I-2] := S$
0D	$\Sigma := S + L$; $S := \Sigma$; $R[I-1] := \Sigma$	$S := S + L$; $R[I-1] := S$
0E	$\Sigma := \overline{S} + (L \vee \overline{L})$; $L := \Pi$	$\overline{S} + 1$; $L := \Pi$
0F	$\Sigma := \overline{R[I]} + S + (L \vee \overline{L})$; $L := \Pi$	$\overline{R[I]} + S + 1$; $L := \Pi$
10	$\Sigma := S + 1$; $L := \Pi$; $S := \Sigma$	$S := S + 1$; $L := \Pi$
11	$\Sigma := S + \overline{S}$; $S := \Sigma$	$S := F$
12	$\Sigma := 4 + (S \vee \overline{S})$; $L := \Pi$; $S := \Sigma$	$S := 3$
13	$S := R[I] + S$; $S := \Sigma$; $R[I] := \Sigma$	$S := R[I] + S$; $R[I] := S$
14	$\Sigma := (R[I] \vee \overline{R[I]}) + S$; $L := \Pi$; $S := \Sigma$	$S := S + F$; $L := \Pi$
15	$\Sigma := R[I] + 6$; $S := \Sigma$	$S := R[I] + 6$
16	$\Sigma := S + S + L$; $L := \Pi$; $S := \Sigma$	$S := S + S + L$; $L := \Pi$
17	$\Sigma := R[I] + S + L$; $L := \Pi$; $S := \Sigma$	$S := R[I] + S + L$; $L := \Pi$
18	$\Sigma := R[I]$; $R[I] := S$; $S := \Sigma$	
19	$\Sigma := S + S1$; $L := \Pi$; $S := \Sigma$; $S1 := \Sigma$	$S := S + S1$; $L := \Pi$; $S1 := S$
1A	$\Sigma := \overline{S} + L$; $L := \Pi$	$\overline{S} + L$; $L := \Pi$
1B	$\Sigma := S + 6 + (L \vee \overline{L})$; $L := \Pi$	$S + 7$; $L := \Pi$
1C	$R[I] := R[I+3]$	
1D	$\Sigma := S + L$; $S := \Sigma$	$S := S + L$

Продолжение табл. 4.12

Адрес МК	Описание МК	Сокращенная запись МК
1E	$\Sigma := S+6+L;\ L := \Pi$	$S+6+L;\ L := \Pi$
1F	$\Sigma := A \times \overline{L} + (S \vee \overline{S}) + L;\ S := \Sigma$	$S := 9 \times \overline{L}$
20	$\Sigma := 0;\ R[I-1] := \Sigma$	$R[I-1] := 0$
21	$\Sigma := S+1;\ R[I-2] := \Sigma$	$R[I-2] := S+1$
22	$\Sigma := M[I];\ S := \Sigma$	$S := M[I]$
23	$\Sigma := M[I];\ M[I] := S;\ R[I-1] := \Sigma$	$R[I-1] := M[I];\ M[I] := S$
24	$\Sigma := \overline{S};\ S := \Sigma$	$S := \overline{S}$
25	$\Sigma := S+(6 \vee 1)+(L \vee \overline{L});\ L := \Pi;$ $S := \Sigma$	$S := S+8;\ L := \Pi$
26	$\Sigma := \overline{R[I]}+1;\ L := \Pi;\ S := \Sigma$	$S := \overline{R[I]}+1;\ L := \Pi$
27	$\Sigma := \overline{R[I]}+L;\ L := \Pi;\ S := \Sigma$	$S := \overline{R[I]}+L;\ L := \Pi$
28	$\Sigma := 4+(\overline{S} \vee 6);\ L := \Pi$	$4+(\overline{S} \vee 6);\ L := \Pi$
29	$\Sigma := (R[I[\vee \overline{R[I]})+S1+L;\ L := \Pi;$ $S := S1$	$S1+F+L;\ L := \Pi;\ S := S1$
2A	$\Sigma := \overline{S}+(L \vee \overline{L});\ R[I-2] := \Sigma$	$R[I-2] := \overline{S}+1$
2B	$\Sigma := R[I]+S+(L \vee \overline{L});\ L := \Pi;$ $S := \Sigma$	$S := R[I]+S+1;\ L := \Pi$
2C	$\Sigma := SR[I];\ S := \Sigma$	$S := SR[I]$
2D	$\Sigma := R[I];\ R[I] := S;\ R[I-1] := \Sigma$	$R[I-1] := R[I];\ R[I] := S$
2E	$\Sigma := (S \vee S1)+L;\ S1 := \Sigma$	$S1 := (S \vee S1)+L$
2F	$\Sigma := R[I];\ R[I] := R[I[+3];\ S1 :=$ $:= \Sigma$	$S1 := R[I];\ R[I] := R[I+3]$
30	$\Sigma := (R[I] \vee \overline{R[I]})+S1;\ L := \Pi;$ $S1 := \Sigma$	$S1 := S1+F;\ L := \Pi$
31	$\Sigma := R[I];\ R[I] := S;\ SR[I] := \Sigma$	$SR[I] := R[I];\ R[I] := S$
32	$\Sigma := M[I];\ R[I-1] := \Sigma$	$R[I-1] := M[I]$
33	$\Sigma := R[I],\ S1 := \Sigma$	$S1 := R[I]$
34	$\Sigma := S1;\ R[I] := \Sigma$	$R[I] := S1$
35	$\Sigma := 4+1;\ S := \Sigma$	$S := 5$
36	$\Sigma := R[I]+(S \vee \overline{S})+L;\ L := \Pi;$ $S := \Sigma$	$S := R[I]+F+L;\ L := \Pi$
37	$\Sigma := R[I]+1;\ SR[I] := \Sigma$	$SR[I] := R[I]+1$
38	$\Sigma := R[I]+(S \vee \overline{S});\ L := \Pi;\ S := \Sigma;$ $R[I] := \Sigma$	$S := R[I]+F;\ L := \Pi;$ $R[I] := S$
39	$\Sigma := R[I]+1;\ S := \Sigma$	$S := R[I]+1$
3A	$\Sigma := R[I];\ R[I-1] := \Sigma$	$R[I-1[:= R[I]$

Окончание табл. 4.12

Адрес МК	Описание МК	Сокращенная запись МК
3B	$\Sigma := A \times \overline{L} + (S \vee \overline{S}) + L$; $R[I] := \Sigma$	$R[I] := 9 \times \overline{L}$
3CL	$\Sigma := R[I]$; $R[I] := S$; $S := \Sigma$; $R[I-1] := \Sigma$	
3C$\overline{\text{L}}$	$\Sigma := S$; $R[I-1] := \Sigma$; $S := \Sigma$	$R[I-1] := S$
3DL	$S1 := S1 \vee H$; $\Sigma := (R[I] \vee \overline{R[I]}) + S1$; $L := \Pi$	$S1 := S1 \vee H$; $S1 + F$; $L := \Pi$
3D$\overline{\text{L}}$	$\Sigma := 4 + S$; $S := \Sigma$	$S := S + 4$
3EL	$\Sigma := S$; $SR[I] := \Sigma$	$SR[I] := S$
3E$\overline{\text{L}}$	$\Sigma := R[I] + S$; $L := \Pi$; $S := S1$	$R[I] + S$; $L := \Pi$; $S := S1$
3FL	$\Sigma := 0$; $S1 := \Sigma$	$S1 := 0$
3F$\overline{\text{L}}$	$\Sigma := S \vee \overline{S}$; $S1 := \Sigma$	$S1 := F$

7. Логические микрокоманды с адресами 28 и 2E обеспечивают проверку определенного условия.

8. Микрокоманда с адресом 3D обеспечивает ввод информации через вывод H, что позволяет фиксировать положительный фронт входного сигнала во времени с точностью до 4-тактного интервала времени E.

9. Микрокоманды с адресами 0E, 1A, 1B, 1E, и 0F используются для проверки разрядов операндов в синхропрограммах многоразрядной обработки информации с изменением содержимого ячейки L в зависимости от результата проверки.

Микрокоманды микроЭВМ эмулятора накопителя ВЗУ (табл. 4.13) по назначению также разбиваются на несколько групп:

1. Микрокоманды с адресами 1D, 2D, 30, 2B, 28, 13, 1E, 20 и 19 обеспечивают обмен содержимым S и S1 и загрузку в эти ячейки ряда констант.

2. Микрокоманды с адресами 00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B, 0C, 0D, 0E, 0F засылают в текущую тетраду регистра R константы, равные адресам. Поэтому запись адресов этих микрокоманд в синхропрограммах для хранения содержимого сегментов является записью в определенном порядке части некоторой страницы сегмента в шестнадцатеричном коде. Перед выполением микрокоманд с адресами 09, 0D содержимое ячейки L должно быть равным нулю. Микрокоманды этой группы с адресами 2F, 35, 2A, 15, 16, 23, 22, 25, 11 обеспечивают выполнение вспомогательных операций.

3. Микропрограммы с адресами 1F и 21 используются для сохранения в стеке программного счетчика адреса очередной команды загружаемого сегмента перед возвратом в загружающую программу.

Таблица 4.13

Адрес МК	Описание МК	Сокращенная запись МК
00	$\Sigma := 0$; $R[I] := \Sigma$	$R[I] := 0$
01	$\Sigma := 1$; $R[I] := \Sigma$	$R[I] := 1$
02	$\Sigma := 1 + (L \vee \overline{L})$; $R[I] := \Sigma$	$R[I] := 2$
03	$\Sigma := 4 + (S \vee \overline{S})$; $R[I] := \Sigma$; $S1 := \Sigma$	$R[I] := 3$; $S1 := 3$
04	$\Sigma := 4$; $R[I] := \Sigma$	$R[I] := 4$
05	$\Sigma := 4 + 1$; $L := \Pi$; $R[I] := \Sigma$	$R[I] := 5$; $L := 0$
06	$\Sigma := 6$; $R[I] := \Sigma$	$R[I] := 6$
07	$\Sigma := 6 + (L \vee \overline{L})$; $R[I] := \Sigma$	$R[I] := 7$
08	$\Sigma := (6 \vee 1) + (L \vee \overline{L})$; $R[I] := \Sigma$	$R[I] := 8$
09	$\Sigma := A \times \overline{L} + (S \vee \overline{S})$; $R[I] := \Sigma$	$R[I] := 9 \times \overline{L} \vee F \times L$
0A	$\Sigma := 4 + 6$; $R[I] := \Sigma$	$R[I] := A$
0B	$\Sigma := 4 + (6 \vee 1)$; $R[I] := \Sigma$; $S1 := \Sigma$	$R[I] := B$; $S1 := B$
0C	$\Sigma := 4 + (6 \vee 1) + (L \vee \overline{L})$; $R[I] := \Sigma$	$R[I] := C$
0D	$\Sigma := (4 \vee A \times \overline{L}) + (S \vee \overline{S})$; $R[I] := \Sigma$; $S1 := \Sigma$	$R[I] := D \times \overline{L} \vee 3 \times L$; $S1 := D \times \overline{L} \vee 3 \times L$
0E	$\Sigma := (R[I] \vee \overline{R[I]}) + (S \vee \overline{S})$; $R[I] := \Sigma$	$R[I] := E$
0F	$\Sigma := R[I] \vee \overline{R[I]}$; $R[I] := \Sigma$; $S1 = \Sigma$	$R[I] := F$; $S1 := F$
10	Нет операции	
11	$\Sigma := R[I]$; $R[I] := S$; $S := \Sigma$	
12	$\Sigma := R[I] + 1$; $R[I] := \Sigma$; $S := S1$; $S1 := \Sigma$	
13	$\Sigma := 6 \vee 1$; $S1 := \Sigma$	$S1 := 7$
14	$\Sigma := (R[I] \vee \overline{R[I]}) + S1$; $L := \Pi$; $S1 := \Sigma$	$S1 := S1 + F$; $L := \Pi$
15	$\Sigma := R[I] \vee \overline{R[I]}$; $L := \Pi$; $R[I] := \Sigma$	$R[I] := F$; $L := 0$
16	$\Sigma := R[I] \vee \overline{R[I]}$; $R[I] := \Sigma$; $S := \Sigma$	$R[I] := F$; $S := F$
17	$\Sigma := R[I] + \overline{L}$; $R[I] := \Sigma$	$R[I] := R[I] + \overline{L}$
18	$\Sigma := M[I]$; $S := \Sigma$	$S := M[I]$
19	$\Sigma := S$; $S := S1$; $S1 := \Sigma$	
1A	$\Sigma := M[I] + S + (L \vee \overline{L})$; $R[I] := S$; $S := \Sigma$	$R[I] := S$; $S := M[I] + S + 1$
1B	$\Sigma := S1 + 1$; $L := \Pi$; $R[I-1] := \Sigma$	$R[I-1] := S1 + 1$; $L := \Pi$
1C	$\Sigma := R[I] + (S \vee \overline{S}) + L$; $R[I] := \Sigma$	$R[I] := R[I] + F + L$

Продолжение табл. 4.13

Адрес МК	Описание МК	Сокращенная запись МК
1D	$\Sigma := 1$; $S := \Sigma$	$S := 1$
1E	$\Sigma := 4$; $L := \Pi$; $S := \Sigma$	$S := 4$; $L := 0$
1F	$\Sigma := R[I]+1$; $L := \Pi$; $R[I] := S$; $SR[I] := \Sigma$	$SR[I] := R[I]+1$; $L := \Pi$; $R[I] := S$
20	$\Sigma := 4$; $L := \Pi$; $S1 := \Sigma$	$S1 := 4$; $L := 0$
21	$\Sigma := R[I]+L$; $R[I] := S$; $SR[I] := \Sigma$	$SR[I] := R[I]+L$; $R[I] := S$
22	$\Sigma := \overline{R[I]}$; $L := \Pi$; $R[I-1] := \Sigma$	$R[I-1] := \overline{R[I]}$; $L := 0$
23	$\Sigma := \overline{R[I]}$; $L := \Pi$; $R[I] := \Sigma$	$R[I] := \overline{R[I]}$; $L := 0$
24	$\Sigma := R[I]+S1$; $S1 := \Sigma$	$S1 := R[I]+S1$
25	$\Sigma := R[I]$; $R[I-1] := \Sigma$; $S := S1$	$R[I-1] := R[I]$; $S := S1$
26	$\Sigma := R[I]$; $R[I-2] := \Sigma$; $R[I] := := R[I+3]$	$R[I-2] := R[I]$; $R[I] := := R[I+3]$
27	$\Sigma := R[I]+L$; $R[I] := \Sigma$	$R[I] := R[I+L]$
28	$\Sigma := 0$; $S1 := \Sigma$	$S1 := 0$
29	$\Sigma := R[I]+S+(L \vee \overline{L})$; $S := \Sigma$	$S := R[I]+S+1$
2A	$\Sigma := \overline{L}$; $R[I] := \Sigma$	$R[I] := \overline{L}$
2B	$\Sigma := 4+(6 \vee 1)+(L \vee \overline{L})$; $S := \Sigma$	$S := C$
2C	$\Sigma := (\overline{R[I]} \vee 4)+6$; $S1 := \Sigma$	$S1 := (\overline{R[I]} \vee 4)+6$
2D	$\Sigma := 4$; $S := \Sigma$	$S := 4$
2E	$\Sigma := S+1$; $R[I] := \Sigma$	$R[I] := S+1$
2F	$\Sigma := (R[I] \vee \overline{R[I]})+1$; $L := \Pi$; $R[I] := \Sigma$	$R[I] := 0$; $L := 1$
30	$\Sigma := (6 \vee 1)+(L \vee \overline{L})$; $S := \Sigma$	$S := 8$
31	$\Sigma := R[I]+S$; $L := \Pi$	$R[I]+S$; $L := \Pi$
32	$\Sigma := S+S1$; $S1 := \Sigma$	$S1 := S+S1$
33	$\Sigma := R[I] \vee 4$; $S := \Sigma$	$S := R[I] \vee 4$
34	$\Sigma := S+(6 \vee 1)+(L \vee \overline{L})$; $L := \Pi$	$S+8$; $L := \Pi$
35	$\Sigma := (R[I] \vee \overline{R[I]})+1+(L \vee \overline{L})$; $L := \Pi$; $R[I] := \Sigma$	$R[I] := 1$; $L := 1$

Окончание табл. 4.13

Адрес МК	Описание МК	Сокращенная запись МК
36	$\Sigma := R[I]+S+(L \vee \overline{L})$; $R[I-1] := \Sigma$	$R[I-1] := R[I]+S+1$
37	$\Sigma := R[I]+S$; $R[I-1] := \Sigma$	$R[I-1] := R[I]+S$
38	$R[I] := R[I+3]$	
39	$\Sigma := M[I]+S+(L \vee \overline{L})$; $S := \Sigma$	$S := M[I]+S+1$
3A	$\Sigma := R[I]+L$; $R[I] := \Sigma$; $S := S1$; $S1 := \Sigma$	$S := S1$; $S1 := R[I]+L$; $R[I] := S1$
3B	$\Sigma := S+S+(L \vee \overline{L})$; $L := \Pi$; $R[I-1] := \Sigma$	$R[I-1] := S+S+1$; $L := \Pi$
3CL	$\Sigma := S+1$; $L := \Pi$; $S := \Sigma$	$S := S+1$; $L := \Pi$
$3C\overline{L}$	$\Sigma := 4$; $R[I] := \Sigma$; $S := \Sigma$	$R[I] := 4$; $S := 4$
3DL	$\Sigma := SR[I]$; $R[I] := \Sigma$; $S1 := S1 \vee H$	$R[I] := SR[I]$; $S1 := S1 \vee H$
$3D\overline{L}$	$\Sigma := S+1$; $R[I] := \Sigma$; $S := \Sigma$	$S := S+1$; $R[I] := S$
3EL	$\Sigma := \overline{R[I]}+(S \vee \overline{S})$; $L := \Pi$; $M[I] := S$	$M[I] := S$; $R[I]+F$; $L := \Pi$
$3E\overline{L}$	$\Sigma := 4$; $S := \Sigma$	$S := 4$
3FL	$\Sigma := S1+(L \vee \overline{L})$; $L := \Pi$; $R[I] := \Sigma$	$R[I] := S1+1$; $L := \Pi$
$3F\overline{L}$	$\Sigma := R[I]+(S \vee \overline{S})$; $L := \Pi$; $R[I] := \Sigma$	$R[I] := R[I]+F$; $L := \Pi$

4. Микрокоманды с адресами 18, 39, 1A, 3E вводят содержимое тетрады с системной магистрали в аккумулятор S. При этом микрокоманда 3E позволяет разрабатывать синхропрограммы для сопровождения ввода сигналом нулевого уровня на выводе I8.

5. Микрокоманды с адресами 38 и 26 перемещают в область мантиссы R2 содержимое страницы сегмента, введенное в область порядка и адреса регистра R.

6. Арифметические микрокоманды с адресами 29, 32, 14, 24, 2E, 27, 17, 1C, 1B, 37, 36, 12, 3A, 3C, 3F, 3B используются для организации работы счетчиков, проверки условий при работе программы и вычисления адреса следующей команды.

7. Логические микрокоманды с адресами 2C и 33 формируют содержимое S и S1, используемое для получения адресов команд.

8. Микрокоманда с адресом 1D вычисляет адрес БИС, вводимый по входу H, и при L:=1 используется для загрузки адреса в программный счетчик из стека, где находится текущий адрес команды, с содержимым страницы сегмента, загружаемого в программную память ПМК.

9. Микрокоманды с адресами 31 и 34 заносят перенос П в ячейку L, не изменяя содержимого ячеек памяти.

Глава 5

ОСОБЕННОСТИ ПОЛЬЗОВАНИЯ ПРОГРАММИРУЕМЫМИ МИКРОКАЛЬКУЛЯТОРАМИ

5.1. КРИТЕРИИ КАЧЕСТВА ПРИКЛАДНЫХ ПРОГРАММ

Технические характеристики, входной язык и особенности эксплуатации достаточно подробно описаны в руководствах по применению ПМК каждого типа. Однако в руководствах, как правило, отсутствует описание общего подхода к составлению, оптимизации и отладке программ решения прикладных задач. Между тем именно качество программ в основном и определяет эффективность применения ПМК любого типа и соответственно производительность труда его пользователя.

Общим критерием целесообразности применения ЭВМ является экономический эффект. Стоимость решения задачи обработки информации на универсальной ЭВМ складывается из стоимости машинного времени (включающего и стоимость машинного времени обслуживающего персонала) и стоимости рабочего времени пользователя, затрачиваемых на решение задачи. Производительность ЭВМ, зависящая от ее технических характеристик, определяет лишь потенциальную возможность решения сложных задач с малыми затратами времени, тогда как реальная производительность ЭВМ связана с качеством программ решения прикладных задач.

Стоимость одного часа работы (машинного времени) ПМК равна отношению суммы всех затрат на покупку, ремонт и питание к сумме числа часов работы ПМК ежегодно в течение срока амортизации. Приняв этот срок равным 7 лет и полагая, что ПМК в среднем работает 200 ч в год, получим стоимость часа машинного времени массовых отечественных ПМК, не превышающую 10 коп. Этот результат изменится при другом числе часов работы ПМК, но он будет меньше стоимости часа рабочего времени пользователя. Независимо от этого полная стоимость решения прикладной задачи* с помощью ПМК будет складываться из стоимости машинного времени и рабочего времени пользователя, пропорциональной сумме затрат времени на составление, оптимизацию и отладку ($T_с$), ввод в оперативную память ($T_к$) и исполнение ($T_м$) программы решения задачи, а также дополнительных затрат времени ($T_д$) в основном на вход исходных данных и регистрацию результатов выполнения программы:

$$T = T_с + T_к + T_м + T_д \qquad (5.1)$$

*Прикладная задача может рассматриваться как решенная, когда результат решения получен с требуемой точностью. Поэтому в дальнейшем при отсутствии оговорок предполагается, что сравниваемые программы обеспечивают решение задач с одинаковой требуемой или достижимой точностью.

Сумма (5.1) определяет производительность пользователя при решении задачи с помощью ПМК и является основным критерием качества (оптимальности) программы решения задачи. Наилучшим будет тот ее вариант, который соответствует наименьшим затратам времени и, следовательно, материальных средств на решение задачи. Оптимизация прикладной программы заключается в выборе ее наилучшего варианта по минимуму суммы (5.1), причем этой цели обычно достигают методом минимакса, последовательно минимизируя наибольшие слагаемые суммы до получения ее минимума.

Слагаемое T_c этой суммы минимально при использовании готовой программы, взятой из справочных пособий, подобных [1, 2, 5, 6, 8, 19—21, 23, 25—27], или вводимой в оперативную память из внешних модулей прикладных программ, подготовленных профессиональными программистами. В этом случае слагаемое T_c определяется затратами времени на выбор нужной программы, оценку ее соответствия (в частности, точности результатов вычислений) условиям прикладной задачи и при необходимости составления согласованной с выбранной программой математической модели, связывающей исходные данные с искомым результатом решения задачи. При использовании готовых программ часто приходится учитывать и их стоимость. Она минимальна для программ, публикуемых в справочных пособиях, но при увеличении длины программы резко возрастает вероятность типографских ошибок, существенно снижающих целесообразность публикации, так как затраты времени на обнаружение и устранение таких ошибок часто превышают затраты времени на составление новой программы самим пользователем. Стоимость пакетов прикладных программ на покупных внешних модулях, подготовленных профессиональными программистами (например, для ПМК типа HP-41C или HP-71B), часто больше стоимости самого ПМК.

Слагаемое T_c максимально и обычно значительно превышает остальные слагаемые суммы (5.1) в тех случаях, когда пользователю приходится самостоятельно составлять, оптимизировать и отлаживать прикладные программы. При этом затраты времени существенно зависят от характера задачи и назначения программы. Если составляемая программа предназначена для разового применения без ее последующего хранения и использования, то оптимальным по суммарным затратам времени обычно оказывается первый же вариант программы, обеспечивающий требуемую точность результатов решения задачи. Исключение могут составлять программы численной оптимизации или численного решения задач методами последовательных приближений при медленной сходимости вычислительного процесса к искомым результатам. Однако и в этих случаях после получения в процессе составления программы первого же результата решения задачи с требуемой точностью дальнейшая оптимизация программы не имеет смысла.

Большинство прикладных программ, в частности программ решения типовых математических задач, к которым сводится решение приклад-

ных задач, целесообразно хранить в библиотеке пользователя для их последующего применения. В этих случаях оправданы и большие начальные затраты времени $T_{со}$ на составление, оптимизацию и отладку таких библиотечных программ с минимизацией суммы затрат времени $T_к + T_м + T_д$, так как при очередном использовании программы составляющая затрат времени $T_с = T_{со}/k$, где k — число обращений к библиотечной программе. Чем больше число k таких обращений, тем меньше затраты времени $T_с$, и, следовательно, наиболее часто употребляемые программы библиотеки пользователя необходимо оптимизировать особенно тщательно. Полные затраты времени $T_{со}$ на составление библиотечной программы зависят не только от вида и назначения задачи, но и в значительной степени от знаний и творческих способностей пользователя, составляющего эти программы. Эти качества требуются на каждом этапе составления программы — при составлении математической модели исходных условий, выборе метода и алгоритма решения задачи, представления алгоритма программой на входном языке, отладке программы и оценке приемлемости полученных результатов. Поиск оптимального варианта сложной задачи в общем случае является итеративным процессом, так как при этом приходится на каждом этапе уточнять и оптимизировать выбор решений на предыдущих этапах.

Следует отметить относительную простоту составления программ на алгебраических входных языках, форма записи которых достаточно близка к стандартизованной форме записи алгоритма решения задачи. В связи с этим профессиональные программисты обычно составляют алгоритм решения задачи непосредственно в виде программы на алгоритмическом входном языке и часто отождествляют алгоритм и программу. Простота представления программой алгоритмов решения задач особенно характерна для языка программирования Бейсик (по крайней мере при программировании относительно простых задач), чем и объясняется преимущественное использование этого входного языка для микроЭВМ, предназначенных для широкого круга пользователей, не имеющих специальной подготовки по вычислительным технике и математике. Значительно сложнее представление алгоритмов решения прикладных задач на компактных входных языках. Это связано как с различной формой записи алгоритмов решения задач и программ на компактных входных языках, так и с относительным разнообразием способов отображения операторов алгоритма фрагментами программ на таких входных языках, вызывающих дополнительные затраты времени на выбор оптимального способа отображения операторов алгоритма.

Таким образом, основным критерием оптимальности библиотечной программы является минимум суммы затрат времени $T_к + T_м + T_д$, и при составлении оптимальной программы приходится тщательно оценивать вклад каждой из этих составляющих, минимизируя в первую очередь наибольшую из них.

Затраты времени $T_к$ минимальны при непосредственном вводе программы, отображенной последовательностью машинных кодов, в оперативную память ПМК с внешних ЗУ (ВЗУ), например встроенного накопителя ПМК «Электроника МК-52». Эти же затраты максимальны при вводе прикладной программы с клавиатуры. В обоих случаях затраты времени примерно пропорциональны длине программы (числу ее шагов). При увеличении длины программы, вводимой с клавиатуры, возрастает вероятность ошибок, вследствие чего затраты времени увеличиваются нелинейно. Поэтому при вводе программы с клавиатуры затраты времени $T_к$ зависят от опыта пользователя и его навыков четкого и безошибочного нажатия клавиш и контроля правильности ввода программы по ее отображению на дисплее в режиме программирования (режиме ввода программы). Таким образом, в общем случае минимальное значение $T_к$ соответствует минимальной длине программы.

Затраты времени $T_м$ на исполнение программы в автоматическом режиме также пропорциональны длине неразветвленной программы, и в этом случае при обычно малых затратах времени $T_д$ критерием оптимальности, соответствующим миниимуму суммы $T_к + T_и$, является минимальная длина программы. Уменьшение длины программы часто удается достичь рациональным выбором математической модели, метода и алгоритма решения задачи, но более эффективно использование обращений к подпрограммам и итерациониых циклов. При решении многих задач (например, решении нелинейных уравнений или численной оптимизации) примеиение итерационных циклов принципиально необходимо, но на выполиение переходов в программе затрачивается зиачительная часть машинного времени. Таким образом, в общем случае умеиьшение длины программы путем использования условных и безусловных переходов (включая внутренние переходы в операторах цикла) приводит к уменьшению составляющей $T_к$ и увеличению составляющей $T_м$ суммы (5.1). Выбор оптимального решения в таких случаях зависит от характера задачи и назначения программы, причем для решения одной и той же задачи могут существовать различные критерии оптимальности. Так, если программа предназначена для многократного исполнения в процессе решения задачи (например, при вычислении функции в задаином интервале аргумента), целесообразно минимизировать затраты машинного времени: в этом случае составляющая $T_м$ будет преобладающей. Однако, если та же программа предназначена для однократного или во всяком случае небольшого числа исполнеиий в процессе решения задачи, составляющая $T_к$ может оказаться преобладающей, и в этом случае целесообразно минимизировать длину программы, используя для этого все средства входного языка. Примером могут служить программы вычисления многочленов действительного аргумента с малыми затратами машиниого времени при большой длине и большими затратами времени при малой длине [23], которые целесообразно использовать соответственно при многократном и однократном вычислении многочлена в процессе решения задачи.

Время счета $T_м$ обычно максимально для программ решения прикладных задач методами последовательных приближений с итерационными циклами. В таких случаях приходится тщательно анализировать возможность уменьшения затрат времени на выполнение каждой итерации, выбирая наиболее подходящие методы и алгоритмы с быстрой сходимостью, а также условия выхода из цикла, устраняющие избыточные затраты времени на получение результата с точностью, превышающей заданную.

Значительные потери времени возникают при решении прикладной задачи с помощью нескольких последовательно используемых программ — пакета программ. Если такой пакет состоит из готовых программ, которые хранятся в ВЗУ и не могут быть изменены пользователем, то обычно приходится составлять дополнительную программу для согласования имен переменных и способа ввода исходных данных в готовые программы на алгоритмических языках или для согласования размещения в памяти данных промежуточных результатов. Если пакет состоит из готовых программ, взятых из справочных пособий или библиотеки пользователя (когда они не связаны заранее в пакет), то их также приходится предварительно корректировать, чтобы согласовать результаты выполнения предыдущей программы с исходными данными для следующей программы. Все это связано со значительными дополнительными потерями времени, и следует использовать все возможности решения задачи и средства входного языка, обеспечивающие уменьшение длины программы решения сложной задачи для размещения ее в оперативной памяти.

Особенно актуальной является минимизация длины программы в том случае, когда лишь небольшая ее часть не помещается в оперативную память ПМК или когда (например, при решении задач методами последовательных приближений) принципиально нельзя использовать пакет программ или его использование приводит к неоправданно большим затратам времени. Теоретически решение любой задачи можно разбить на ряд этапов, реализуемых отдельными программами, но часто это приводит к очень большим затратам машинного времени и рабочего времени пользователя ПМК.

Дополнительные затраты времени $T_д$ в основном связаны с потерями времени на ввод в память исходных данных, а также на вызов из памяти и регистрацию результатов выполнения программы. Обычно эти затраты можно уменьшить, повысив степень автоматизации ввода и вывода данных. Например, число клавиш, нажимаемых при вводе данных, и, следовательно, время $T_д$ можно уменьшить, организовав в начале программы фрагменты для автоматического распределения в нужные регистры данных набираемых чисел. Аналогично в конце программы можно ввести фрагмент, автоматически вызывающий на индикатор очередной результат выполнения программы при дополнительных пусках программы после ее останова для регистрации очередного результата. Однако в этом случае увеличиваются длина программы и составляющие $T_к$ и $T_м$ суммы (5.1). Поэтому повышение степе-

ни автоматизации ввода и вывода данных оправдано лишь в том случае, когда это не приводит к увеличению суммарных затрат времени.

Дополнительные затраты времени на ввод и вывод данных значительно возрастают, если все данные не удается разместить в оперативной памяти и для хранения промежуточных данных приходится использовать внешнюю память в виде накопителя информации или вычислительного бланка для регистрации и считывания промежуточных результатов вычислений. Во многих случаях эти затраты удается устранить или уменьшить, используя различные приемы программирования, обеспечивающие снижение требований к емкости памяти данных. Однако, если необходимо одновременно хранить в памяти большое число данных, эти приемы могут оказаться непригодными и приходится использовать внешнюю память. Например, при решении систем из n линейных уравнений приходится использовать бланки, когда необходимо хранить одновременно (постоянно или на каком-либо из этапов решения задачи), как правило, $n^2 + n$ коэффициентов, число которых превышает число регистров памяти данных. Это число можно уменьшить для систем уравнений определенного вида или определенных методов решения систем линейных уравнений [19, 21], но даже в этом случае при большом n ресурс памяти данных может оказаться недостаточным.

Большое значение имеет форма вычислительного бланка, обеспечивающая минимальное число ошибок при регистрации и считывании данных и возможность контроля правильности вычислений, своевременного обнаружения и устранения ошибок. Обычно для этой цели целесообразно использовать бланки, разработанные для решения соответствующих задач без применеиия ЭВМ, часто их форму можно оптимизировать с учетом особенностей применяемого ПМК.

При использовании в качестве внешней памяти накопителя информации или внешнего расширителя оперативной памяти необходнмо тщательно организовать обмен информацией между оперативной и внешней памятью, обеспечивая минимальное число пересылок данных и остановов программы для обмена информацией.

Дополнительные затраты времени зависят также от четкости инструкции к программе. В ней должны быть ясно и однозначно описаны все операции (ввод и вывод данных, а также пуск и останов программы в процессе решения задачи), содержание и последовательность выполнения которых должны обеспечивать минимальные затраты времени.

Большое значение при составлении и вводе с клавиатуры имеет форма записи программы, особенно при использовании ПМК с компактным входным языком.

5.2. ФОРМЫ ЗАПИСИ ПРИКЛАДНЫХ ПРОГРАММ

В процессе составления и особенно оптимизации библиотечных прикладных программ приходится многократно переписывать их текст в поисках наилучшего варианта. В этом случае затраты времени (5.1)

существенно зависят от формы записи программы. Запись готовых оптимизированных и отлаженных программ в библиотеке пользователя должна быть максимально информативной, чтобы уменьшить затраты времени на оценку целесообразности использования программы для решения конкретной задачи с требуемой точностью по данной программе, ввод в оперативную память, а также ввод исходных данных и регистрацию результатов.

Для иллюстрации выбора формы записи программ, удовлетворяющей упомянутым противоречивым требованиям, рассмотрим следующий пример. В прикладных задачах часто встречается необходимость решения алгебраических уравнений второго порядка $a_2x^2 + a_1x + a_0 = 0$. С этой целью обычно используют прямой метод вычисления корней уравнения по традиционной формуле

$$x_{1,2} = \alpha \pm \sqrt{D}, \tag{5.2}$$

где $\alpha = -a_1/2a_2$; $D = \alpha^2 - \beta$, $\beta = a_0/a_2$.

Эта формула (реализуемая приведенным в гл. 1 алгоритмом) обеспечивает вычисление корней с требуемой точностью при отсутствии ограничений на разрядность операидов. Однако при вычислениях по этой формуле на ПМК с ограниченной разрядностью операндов возникают погрешности округления, часто приводящие к значительной потере точности вычисляемых корней уравиения. Так, при вычислении значения D под знаком радикала в формуле (5.2) при разрядности r мантиссы машинного представления чисел и условии $\alpha^2 > 10^r\beta$ число β попадает в область машинного нуля отиосительно числа α^2 и не влияет на результат вычисления действительиых корней уравнения $x_1 = 2\alpha$, $x_2 = 0$. Если подставить эти вычисленные значения корней в левую часть исходного уравнеиия, то ее значение для обоих корней окажется равным β, а не нулю, соответствующему точному значению корней. Например, при решении на ПМК с разрядностью $r = 8$ мантиссы числа в показательной форме уравнения $x^2 + 10\,000\,x + 9 = 0$ по формуле (5.2) получим значения корней $x_1 = -10\,000$, $x_2 = 0$. Подставив эти значения корней в исходное уравнение, получим в обоих случаях значения левых частей уравнения отличными от нуля и равными 9.

Если рассмотренное неравенство выполняется лишь для части цифр β, отличающейся значительно от α^2 по порядку, также возникает значительная погрешность в определении меньшего действительного корня. Между тем, если при решении прикладных задач алгебраическое уравнение второго порядка моделирует свойства физического объекта, требуется достаточно точное определение именно меньшего действительного корня, ближайшего к началу координат. При $x \geqslant 0$ моделируемый объект становится неустойчивым, а при $x < 0$ значение этого корня определяет запас устойчивости моделируемого объекта.

Низкая точность вычисления корня x_2 по формуле (5.2) связана с округлением результата вычитания под знаком радикала. Следовательно, при устранеиии этой операции вычитания близких чисел при разных знаках составляющих правой части формулы (5.2) точность может быть повышена. В соответствии с основной теоремой алгебры коэффициент степенного многочлена при нулевой степени аргумента равен произведению всех корней многочлена или $\beta = a_0/a_2 = x_1x_2$. Поэтому точность меньшего действительного корня можно повысить, вычисляя его по формуле $x_2 = \beta/x_1$, где больший действительный корень $x_1 = \pm(|\alpha| + \sqrt{D})$ со знаком, совпадающим со знаком коэффициента α.

Так, для уравнения $x^2 + 100\,000x + 9 = 0$ при $r = 8$ получим кории $x_1 = -100\,000$, $x_2 = -9 \cdot 10^{-4}$ со значительно меньшей погрешностью меньшего корня, определением его положения относительно точки $x = 0$ и значеиием левой части уравнения $P(x_2) \approx 8 \cdot 10^{-7}$.

При использовании стандартных функций определения модуля ABS(x) и знака SGN(x) числа вычисление корней алгебраического

уравнения второй степени с повышенной точностью вычисления меньшего действительного корня по приведенным формулам описывается алгоритмом:

1. Задать коэффициенты a_2, a_1, a_0.
2. Принять $\alpha = -a_1/2a_2$.
3. Принять $\beta = a_0/a_2$.
4. Принять $D = \alpha^2 - \beta$.
5. Если $D < 0$, то перейти к шагу 9.
6. Принять $x_1 = (\mathrm{ABS}(\alpha) + \sqrt{D}) \times \mathrm{SGN}(\alpha)$.
7. Принять $x_2 = \alpha\beta/x_1$.
8. Стоп.
9. Принять $\mathrm{Re}\ x_{1,2} = \alpha$, $\mathrm{Im}\ x_{1,2} = \sqrt{-D}$.
10. Закончить вычисления.

Этот алгоритм можно непосредственно представить на основной версии языка Бейсик следующей программой:

```
10  DATE a0, a1, a2
20  READ A0, A1, A2;
30  LET A=-.5*A1/A2;
40  LET B=A0/A2;
50  LET D=A↑2-B;
60  IF D<0 THEN 110;
70  LET X1=(ABS(A)+SQR(D))*SGN(A);
80  LET X2=B/X1;
90  PRINT «X1=»X1, «X2=»X2;
100 STOP;
110 LET C=SQR(-D);
120 PRINT «REX=»A, «IMX=»C;
130 END
```

В этой записи знаки ; использованы как символы разделения строк операторов (символы перевода каретки). Подобная запись программы на алгоритмическом языке может использоваться как при составлении, так и для хранения библиотечной программы. В последнем случае она должна содержать все символы входного языка, используемые для ввода программы в оперативную память с клавиатуры. Кроме того, библиотечная программа должна быть снабжена заголовком с четким указанием назначения и (при необходимости) отличия программы от других программ решения подобных задач, а также инструкцией (которая может записываться в виде комментариев) и контрольными примерами для проверки правильности ввода и выполнения программы с указанием в необходимых случаях точности результатов вычислений и времени выполнения программы. Контрольный пример должен обеспечивать проверку всех разветвлений в программе.

Если алгоритм содержит операторы, отсутствующие во входном языке, их заменяют последовательностями более простых операторов из словарного запаса входного языка ПМК. Так, при отсутствии во входном языке операторов определения модуля и знака чисел решение рассматриваемого алгебраического уравнения с повышенной точностью вычисления меньшего действительного корня можно описать, например, следующим алгоритмом:

1. Задать коэффициенты a_0, a_1 и a_2.
2. Принять $\alpha = -a_1/2a_2$.
3. Принять $\beta = a_0/a_2$.
4. Принять $D = \alpha^2 - \beta$.
5. Если $D < 0$, то перейти к шагу 12.
6. Если $\alpha < 0$, то перейти к шагу 9.
7. Принять $x_1 = \alpha + \sqrt{D}$.
8. Перейти к шагу 10.
9. Принять $x_1 = \alpha - \sqrt{D}$.
10. Принять $x_1 = \beta/x_2$.
11. Стоп.
12. Принять $\mathrm{Re}\, x_{1,2} = \alpha$, $\mathrm{Im} x_{1,2} = \sqrt{-D}$.
13. Закончить вычисления.

Библиотечную программу, составленную по этому алгоритму, можно записать, в частности, со следующим текстом на входном языке ПМК «Электроника МК-85», являющимся упрощенной версией языка Бейсик:

```
1  DATA a0, a1, a2 EXE
2  READ A∅, A1, A2 EXE
3  A=—A1/2/A2 EXE
4  B=A∅/A2 EXE
5  D=A↑2—B EXE
6  IF D<0 THEN 14 EXE
7  IF A<0 THEN 10 EXE
8  X1=A+SQR(D) EXE
9  GOTO 11 EXE
10 X1=A—SQR(D) EXE
11 X2=B/X1 EXE
12 PRINT «X1=» X1, «X2=» X2 EXE
13 STOP EXE
14 C=SQR(—D) EXE
15 PRINT «REX=» A, «IMX=» C EXE
16 END EXE
```

Для контроля правильности ввода этой программы вместо символов a_0, a_1 и a_2 в первой строке следует ввести численные значения ко-

эффициентов из контрольного примера для действительных корней, проверить правильность их вычисления и повторить эти операции для проверки правильности вычисления комплексно-сопряженных корней. После этого, если ошибки не обнаружены, можно использовать программу для решения задачи.

При составлении программ на алгоритмических языках можно использовать сокращенные обозначения операторов, причем для оптимизации программы можно использовать как алгоритм, если он отображается достаточно однозначно операторами программы, так и запись программы.

Значительно более разнообразны формы записи программ на компактных входных языках ПМК. В руководствах по их применению и учебных пособиях прикладные программы обычно представляют в виде таблиц, каждая строка которых содержит достаточно полную информацию о каждом шаге программы — его адресе, индицируемом при вводе программы коде, нажимаемых клавишах, назначении каждого шага и содержимом операционного стека. При этом в руководствах по применению ПМК символы нажимаемых клавиш для однозначности отображают прямоугольниками с указанием всех символов, обозначенных на и около нажимаемой клавиши. В учебных пособиях в табличной записи программ обычно указывают лишь символы элементов алфавита, обозначенные на клавише, или при предварительном вводе префиксной клавиши символы алфавита, обозначенные около клавиши и соответствующие вводимому оператору.

При необходимости в инструкции или контрольных примерах приводят информацию о точности вычисления результатов и времени выполнения программы, что особенно существенно для программ с большим временем счета, содержащих циклы.

Для примера в табл. 5.1 приведена программа решения алгебраического уравнения на ПМК «Электроника Б3-34», реализованная по второму из приведенных алгоритмов в связи с отсутствием в словарном запасе входного языка операторов определения модуля и знака чисел.

В первых столбцах этой записи программы указаны адреса, символы нажимаемых клавиш и коды (шестнадцатеричные цифры показаны в форме, индицируемой на дисплее) шагов программы, а также их назначение, отображаемое операторами присваивания для регистров X, Y и в необходимых случаях X1, а также содержимое регистров X, Y, Z, T и X1 операционного стека после выполнения каждого шага.

Заметим, что на входном языке ПМК «Электроника Б3-34» можно реализовать и первый из приведенных выше алгоритмов, отображая функции определения модуля ABS(x) и знака SGN(x) соответственно фрагментами программы $x^2\sqrt{}$ и $\uparrow$ x^2 $\sqrt{}$ $\div$, но в этом случае программа будет несколько длиннее.

Программа, записанная в табл. 5.1, составлена с автоматическим распределением в памяти коэффициентов a_2, a_1 и a_0 и пуском програм-

Таблица 5.1

Решение уравнения $a_2x^2+a_1x+a_0=0$ с повышенной точностью вычисления меньшего действительного корня на ПМК «Электроника Б3-34»

Адрес	Клавиши	Код	Назначение	X	Y	Z	T	X1
00	П 2	42	$P2:=x$	a_2	0	0	0	0
01	С/П	50	Стоп	a_1	a_2	0	0	0
02	ИП 2	62	$y:=x$, $x:=P2$	a_2	a_1	a_2	0	0
03	/—/	0L	$x:=-x$	$-a_2$	a_1	a_2	0	0
04	2	02	$y:=x$, $x:=2$	2	$-a_2$	a_1	a_2	0
05	×	12	$x:=y\times x$, $y:=z$	$-2a_2$	a_1	a_2	a_2	2
06	÷	13	$x:=y/x$, $y:=z$	α	a_2	a_2	a_2	$-2a_2$
07	П 1	41	$P1:=x$	α	a_2	a_2	a_2	$-2a_2$
08	С/П	50	Стоп	a_0	α	a_2	a_2	$-2a_2$
09	ИП 2	62	$y:=x$, $x:=P2$	a_2	a_0	α	a_2	$-2a_2$
10	÷	13	$x:=y/x$, $y:=z$	β	α	a_2	a_2	a_2
11	П 0	40	$P0:=x$	β	α	a_2	a_2	a_2
12	ИП 1	61	$y:=x$, $x:=P1$	α	β	α	a_2	α
13	F x^2	22	$x:=x^2$	α^2	β	α	a_2	a
14	ИП 0	60	$y:=x$, $x:=P0$	β	α^2	β	α	α
15	—	11	$x:=y-x$, $y:=z$	D	β	α	α	β
16	F $x\geqslant 0$	59	Если $x<0$, то перейти на	D	β	α	α	β
17	2 9	29	адрес 29					
18	F $\sqrt{}$	21	$x:=\sqrt{x}$	$\sqrt{D}$	β	α	α	D
19	ИП 1	61	$y:=x$, $x:=P1$	α	$\sqrt{D}$	β	α	D
20	F $x<0$	5C	Если $x\geqslant 0$, то перейти на	α	$\sqrt{D}$	β	α	D
21	2 5	25	адрес 25					
22	XY	14	$x1:=x$, $y:=x$, $x:=y$	$\sqrt{D}$	α	β	α	α
23	—	11	$x:=y-x$, $y:=z$	x_1	β	α	α	$\sqrt{D}$
24	0	00	$y:=x$, $x:=0$	0	x_1	β	α	$\sqrt{D}$
25	+	10	$x:=y+x$, $y:=z$	x_1	β	α	α	0
26	÷	13	$x:=y/x$, $y:=z$	x_2	α	α	α	x_1
27	F Bx	0	$y:=x$, $x:=x1$	x_1	x_2	α	α	x_1
28	С/П	50	Стоп	x_1	x_2	α	α	x_1
29	F $\sqrt{}$	21	$x:=$ ЕГГОГ	D	β	α	α	D
30	↑	0E	$y:=x$, $x:=x1$	D	D	β	α	D
31	/—/	01	$x:=-x$	$-D$	D	β	α	D

Окончание табл. 5.1

Адрес	Клавиши	Код	Назначение	X	Y	Z	T	X1
32	F √	21	$x := \sqrt{x}$	$\sqrt{-D}$	D	β	α	D
33	ИП 1	61	$y := x$, $x :=$ P1	α	$\sqrt{-D}$	D	β	D
34	С/П	50	Стоп	α	$\sqrt{-D}$	D	β	D

Инструкция:
1. Включить ПМК.
2. Нажать клавиши F и ПРГ.
3. Ввести текст программы нажатием клавиш.
4. Нажать клавиши F и АВТ.
5. Набрать a_2 и нажать клавиши В/О и С/П.
6. Набрать a_1 и нажать клавишу С/П.
7. Набрать a_0 и нажать клавишу С/П.
8. Если индицировано число, то оно равно действительному корню x_1, для вызова x_2 нажать клавишу XY.
9. Если индицировано сообщение ЕГГОГ, нажать клавишу С/П, зарегистрировать индицируемое значение Re $x_{1,2}$ и, нажав клавишу XY, вызвать значение Im $x_{1,2}$.

мы после ввода каждого коэффициента, что несколько (на время нажатия трех клавиш) сокращает затраты на ввод коэффициентов, но удлиняет программу. Для уменьшения длины программы шаги 5—9 алгоритма реализованы фрагментом x < 0 25 XY — 0 + ÷ Bx.

Для идентификации действительных или комплексно-сопряженных корней в программе предусмотрена непосредственная засылка действительных корней в регистры X и Y и предварительное высвечивание сообщения ЕГГОГ для комплексных корней с их засылкой в регистры X и Y после дополнительного пуска программы. При этом для вызова в регистр X корня x_2 или мнимой части комплексно-сопряженных корней $\mathrm{Im}x_{1,2}$ достаточно нажать клавишу XY с сохранением информации в операционном стеке.

Табличная форма записи, подобная приведенной в табл. 5.1, содержит достаточно полную информацию о программе, дополняемую контрольным примером вида: уравнение $x^2 + 10\,000x + 9 = 0$ имеет действительные корни $x_1 = -9999{,}999$, $x_2 = -9{,}0000009 \cdot 10^{-4}$; уравнение $x^2 + 2x + 5 = 0$ имеет комплексно-сопряженные корни $x_{1,2} = 1 \pm j2$ ($t \approx 10$ с). Однако подобная таблица весьма громоздка, и ее составление связано со значительными затратами времени. Поэтому для библиотечных программ используют упрощенные таблицы, чаще всего содержащие только информацию об адресах, кодах и нажимаемых клавишах для каждого шага программы. Так как составление библиотечной программы само по себе предполагает определенный опыт пользователя в применении ПМК, то в инструкциях к программе в такой табличной записи описывают лишь те операции, которые необходимы для выполнения только данной программы. При этом иногда исходные данные указывают их именами (символами), а операции

пуска программы — символами нажимаемых клавиш. В этом случае определенная неоднозначность возникает при описании размещения в памяти результатов операций, что требует дополнительных комментариев [25].

Программа может быть записана в несколько столбцов, что уменьшает ее объем и объем библиотеки с большим числом программ. Подобная запись программы на входном языке ПМК семейства «Электроника МК-61», представляющая первый из приведенных алгоритмов, приведена в табл. 5.2. На ПМК этого семейства может быть выполнена и программа, приведенная в табл. 5.1, если учесть соответствие символов П, ИП, ↑ и XY входного языка ПМК «Электроника Б3-34» символам x → П, П → x, B ↑ и ↔ остальных ПМК расширяющегося ряда. Контрольный пример для этой программы такой же, как и для предыдущей при приблизительно одинаковом времени счета.

Таблица 5.2

Решение уравнения $a_2x^2+a_1x+a_0=0$ с повышенной точностью вычисления меньшего действительного корня на ПМК семейства «Электроника МК-61»

Адрес	Клавиши	Код	Адрес	Клавиши	Код	Адрес	Клавиши	Код
00	x → П 2	42	12	П → x 1	61	24	×	12
01	С/П	50	13	F x^2	22	25	B ↑	0E
02	П → x 2	62	14	П → x 0	60	26	П → x 0	60
03	/—/	0L	15	—	11	27	↔	14
04	2	02	16	F x ⩾ 0	59	28	÷	13
05	×	12	17	3 0	30	29	С/П	50
06	÷	13	18	F √	21	30	F √	21
07	x → П 1	41	19	П → x 1	61	31	B ↑	0E
08	С/П	50	20	| x |	31	32	/—/	0I
09	П → x 2	62	21	+	10	33	F √	21
10	÷	13	22	П → x 1	61	34	П → x 1	61
11	x → П 0	40	23	ЗН	32	35	С/П	50

Инструкция. a_2 В/О С/П a_1 С/П a_0 С/П $x_1 \leftrightarrow x_2$ или ЕГГОГ С/П Re $x_{1,2} \leftrightarrow$ Im $x_{1,2}$. Время счета около 10 с.

Форма, подобная приведенной в табл. 5.2, удобна для библиотечных программ, так как содержит достаточно полную информацию для их ввода и выполнения. Однако для составления и оптимизации программ она громоздка и содержит избыточную информацию. В подобных случаях необходим только столбец с символами нажимаемых клавиш, соответствующих символам вводимых операторов. Адреса ша-

гов программы определяются их порядковыми номерами и по существу нужны только при организации переходов, а коды операторов, высвечиваемые в режиме программирования, не нужны при составлении и оптимизации самого текста программы.

Значительные затраты времени при многократном переписывании текста программ связаны с записью сложных по начертанию операторов входного языка. Поэтому целесообразно в составляемых текстах программ использовать их упрощенные варианты. Это относится прежде всего к символу оператора поворота стека по часовой стрелке, обозначаемого на клавиатуре окружностью со стрелками, который целесообразно заменить направленной вправо стрелкой →. Особенно неудобны для записи и идентификации стандартизованные символы операторов обращения к памяти большинства ПМК расширяющегося ряда. Поэтому при составлении программ целесообразно использовать достаточно простые обозначения на клавишах ПМК «Электроника Б3-34» П, ИП, ↑ вместо более сложных используемых для остальных ПМК расширяющегося ряда x→П, П→x и В↑.

Кроме того, в текстах программ целесообразно опускать символы префиксных клавиш F и К, за исключением символа К в операторах косвенной адресации, где он необходим для отличия от соответствующих операторов прямой адресации. Заметим, что префиксные клавиши для набора знаков верхних рядов никогда не указывают в текстах на алгоритмических языках, как не указывают в машинописных текстах символы клавиши поднятия каретки для печатания заглавных букв. Отсутствие символов префиксных клавиш может смутить только начинающего пользователя, но и он быстро привыкает к тому, что без предварительного нажатия префиксных клавиш нельзя ввести символы (как и при машинописи) неосновного ряда.

Символы клавиш каждого шага целесообразно записывать слитно при составлении программы по 10 шагов в строке. В этом случае адрес ab любого шага программы легко определить по номерам строки $a = 0, 1, 2, \ldots$ и столбца $b = 0, 1, 2, \ldots$, на пересечении которых находится нужный шаг программы.

Подобная компактная запись может использоваться и для библиотечных программ. В этом случае текст программы необходимо дополнить соответствующим заголовком и инструкцией. Для стандартизации инструкций целесообразно размещение исходных данных в регистрах памяти или операционного стека описывать формулами вида $a = PN$, означающими, что перед пуском программы число a должно быть занесено в регистр памяти или операционного стека с адресом или номером N, причем символ PN является именем переменной, хранящейся в регистре N с присвоенным числовым значением a. Операции пуска программы целесообразно описывать символами нажимаемых клавиш, как и вспомогательные операции по перемещению данных, если они необходимы. Размещение в памяти результатов вычислений целесообразно описывать формулами присваивания вида $PN = a$, означающими, что после выполнения программы переменной, храня-

щейся в регистре N, с именем PN, присвоено числовое значение a. Так как во многих случаях целесообразно сохранять исходные данные после выполнения программы для последующего использования, то их указывают в скобках, заключающих соответствующие формулы присваивания.

Примером библиотечной программы, составленной по этим правилам, может служить следующая.

Решение уравнения $a_2x^2 + a_1x + a_0 = 0$ с повышеиной точиостью меньшего действительиого корня на ПМК расширяющегося ряда

П2	С/П	ИП2	/—/	2	×	÷	П1	С/П	ИП2
÷	П0	ИП1	x^2	ИП0	—	x≥0	29	$\sqrt{}$	ИП1
x<0	25	↔	—	0	+	÷	Вх	С/П	$\sqrt{}$
↑	/—/	$\sqrt{}$	ИП1	С/П					

Инструкция. a_2=РХ В/0 С/П a_1=РХ С/П a_0=РХ С/П РХ=x_1, РУ=x_2 или РХ=ЕГГОГ С/П РХ=Re $x_{1,2}$, РУ=Im $x_{1,2}$ ($t \approx 10$ с).

В подобной компактной записи адрес ab любого шага программы легко определить по номеру строки $a = 0, 1, \ldots$ и столбца $b = 0, 1, 2, \ldots$, на пересечении которых в программе находится нужный шаг. Определенным недостатком, в особенности для начинающих программистов, является отсутствие в этой записи кодов шагов программы, индицируемых в режиме ввода программы (программирования). Однако пользователь легко запоминает коды арифметических операторов 10, 11, 12 и 13 и коды операторов обращения к памяти данных $4N$ и $6N$, наиболее часто встречающиеся в программах. Коды остальных шагов легко проверить, нажав клавишу $\overleftarrow{\text{ШГ}}$ и четко повторив ввод шага.

Наиболее частой ошибкой при вводе программ в оперативную память ПМК с компактным входным языком является пропуск шага программы вследствие недостаточно четкого нажатия клавиши. Поэтому ввод программы целесообразно контролировать по адресам $ab = 10, 20, 30, \ldots$, индицируемыми после ввода крайнего правого в каждой строке программы шага. Если индицируется меньший требуемый адрес, то в строке пропущен шаг программы, который должен быть восстановлен. При таком контроле для ввода любой программы в память ПМК расширяющегося ряда требуется не более 1—2 мин.

При необходимости пользователь может дополнить компактную запись программы кодами шагов [18], представив программу в табличной форме, подобной показанной в табл. 5.3. Однако подобную форму записи целесообразно использовать лишь для библиотечных программ, причем в этом случае перечень кодов обеспечивает дополнительный контроль.

Некоторые пользователи ПМК записывают прикладные программы в виде последовательности кодов шагов, но такая форма записи затрудняет понимание алгоритма решения задачи и пригодна лишь в виде исключеиия для пользователей, безошибочно сопоставляющих в собственной памяти коды операторов с нажимаемыми клавишами.

Таблица 5.3

Решение уравнения $a_2x^2+a_1x+a_0=0$ с повышенной точностью вычнслеиия меиьшего действнтельного кория на ПМК расширяющегося ряда

П2 42	С/П 50	ИП2 62	/—/ 0L	2 02	× 12	÷ 13	П1 41	С/П 50	ИП2 62
÷ 13	П0 40	ИП1 61	x^2 22	ИП0 60	— 11	$x\geqslant 0$ 59	29 29	$\sqrt{}$ 21	ИП1 61
$x<0$ 5C	25 25	↔ 14	— 11	0 00	+ 10	÷ 13	Bx 0	С/П 50	$\sqrt{}$ 21
↑ 0E	/—/ 01	$\sqrt{}$ 21	ИП1 61	С/П 50					

Инструкция. a_2=РХ В/О С/П a_1=РХ С/П a_0=РХ С/П РХ=x_1, РУ=x_2 или РХ=ЕГГОГ С/П РХ=Re $x_{1,2}$, РУ=Im $x_{1,2}$ ($t\approx 10$ с).

В литературе встречается запись программ на компактных входных языках, в которой каждый символ шага программы записан после отделяемого точкой адреса этого шага, например:

00.П2 01.С/П 02.ИП2 03./—/ 04.2 05.× 06.÷ 07.П1 08.С/П ...

Однако такая запись неудобна для чтения текста программы, так как адреса «затеняют» символы шагов и она содержит больше символов, чем приведенная ранее компактная запись программы с легко определяемыми адресами по номерам строк и столбцов.

Общий недостаток приведенных способов записи прикладных программ на компактных входных языках ПМК — сложность понимания представленного программой алгоритма. Этот недостаток можно несколько ослабить, записав программу по строкам, соответствующим отдельным операторам алгоритма или другим его частям, как рассмотрено в гл. 1. Например, программа из табл. 5.3 может быть записана в форме

(00) П2 С/П
(02) ИП2 /—/ 2 × ÷ П1 С/П
(09) ИП2 ÷ П0
(12) ИП1 x^2 ИП0 —
(16) $x\geqslant 0$ 29
(17) $\sqrt{}$ ИП1
(20) $x<0$ 25

(22) ↔ — С

(25) ⁻| ÷ Вх С/П

(29) √ ↑ /—/ √ ИПI С/П

В скобках указаны адреса начальных шагов каждой строки, содержащей самостоятельный фрагмент. Подобная запись особенно удобна при составлении и оптимизации программ решения сложных задач с большим числом переходов и различными вариантами представления операторов алгоритма, который обычно также приходится уточнять в процессе составления подобных программ.

Выбор формы записи программы зависит и от собственных критериев пользователя, тем более, что требование минимальных затрат времени на решение задачи является первостепенным лишь при использовании ПМК в служебных целях.

5.3. ТОЧНОСТЬ РЕЗУЛЬТАТОВ ВЫЧИСЛЕНИЙ

Необходимым условием окончания решения задачи с помощью ПМК является получение результатов с требуемой точностью. Численные результаты подавляющего большинства прикладных задач могут быть получены лишь приближенно. Источниками отклонения результата решения задач от точного значения (погрешности) являются погрешности математической модели и исходных данных, а также методические и операционные погрешности вычислений.

Точность приближенного результата a решения задачи теоретически оценивают по истинной абсолютной погрешности $\Delta_{ист} = a - a^*$ или истинной относительной погрешности $\delta_{ист} = \Delta_{ист}/|a^*|$. Однако точное значение результата a^* обычно неизвестно, так как в противном случае отпала бы необходимость в вычислении a. Исключение составляют лишь вычисления, выполняемые для экспериментального выбора метода, алгоритма или программы решения по получаемой точности результата вычислений.

В общем случае погрешности результатов решения задач оценивают их предельными погрешностями, равными половине ширины интервала, в котором по имеющейся информации находится точное значение

$$a - \Delta a \leqslant a^* \leqslant a + \Delta a.$$

При этом предельная абсолютная погрешность Δa должна быть как можно ближе к истинной. В противном случае оценка точности результата окажется заниженной и, если она не удовлетворяет условиям задачи, потребуется дополнительное время на уменьшение не существующей в действительности погрешности.

Точность результатов вычислений часто оценивают по их предельной относительной погрешности

$$\delta a = \Delta a / |a| \text{ или } \delta a[\%] = 100\Delta a / |a|.$$

При описании погрешностей слово «предельный» обычно опускают, подразумевая под погрешностями их предельные значения.

Для определения погрешности, состоящей из нескольких слагаемых, знаки которых могут изменяться, используют мажоритарную оценку погрешности как суммы модулей ее слагаемых. Однако такая оценка часто оказывается завышенной, и при определении полной погрешности учитывают неизменяющиеся знаки всех или некоторых слагаемых. При большом числе слагаемых погрешности используют статистические оценки погрешности, принимая ее значение равным математическому ожиданию (обычно среднему арифметическому значению) с учетом вероятности отклонений от этого значения.

Точность результатов вычислений часто удобно оценивать по числу верных цифр в мантиссе десятичного представления чисел. Верными (в узком смысле) называют первые слева k значащих (без нулей слева) цифр мантиссы, если погрешность числа не превышает половины единицы содержимого разряда с k-й цифрой. Так, число $a = 0{,}1234567$ с абсолютной погрешностью $\Delta a \leqslant 0{,}5 \cdot 10^{-6}$ содержит не менее 5 верных цифр. Верные цифры приближенного числа не всегда совпадают с верными цифрами точного значения этого числа. Например, для точного числа 0,996 число 1,00 будет его приближением с тремя верными цифрами, так как погрешность этого приближения $0{,}4 \cdot 10^{-3} < 0{,}5 \cdot 10^{-3}$. Относительная погрешность числа с $k \geqslant 2$ верными цифрами

$$\delta a = 10^{1-k} / 2a_m,$$

где a_m — первая значащая цифра мантиссы. По известной относительной погрешности $\delta a = \Delta a/|a|$ результата вычислений число его верных цифр определяется формулой

$$k = 1 - \mathrm{INT}\ (\lg(2a_m \delta a)).$$

Иногда точность приближенных чисел оценивают числом верных в широком смысле цифр, когда абсолютная погрешность не превышает единицы содержимого k-го разряда. Для чисел с $k \geqslant 2$ верными в широком смысле цифрами относительная погрешность

$$\delta a \leqslant 10^{1-k}/a_m,$$

а по известной относительной погрешности приближенного числа можно определить и число его верных в широком смысле цифр

$$k = 1 - \mathrm{INT}(\lg\ (a_m \delta a)).$$

Высокая точность результатов вычислений обычно требуется лишь при решении чисто математических задач, когда исходные данные и математическая модель задачи предполагаются точно заданными. В этих случаях (например, при вычислении специальных функций) для уменьшения затрат времени вычисления выполняют не с максимально достижимой точностью, а с наперед заданным числом верных цифр или предельными погрешностями.

При решении подавляющего большинства прикладных задач требуемая точность результата решения определяется возможной точностью измерения или определения физических величин, моделируемых численными результатами решения прикладной задачи. Обычно точность измерения физических величин определяется небольшим числом (от 2 до 4) верных цифр, а в результате вычислений требуется число верных цифр, не более чем на единицу превышающее число верных цифр в значении соответствующей физической величины. Однако при низкой требуемой точности результатов решения прикладных задач не всегда удается оценить и обеспечить с необходимой достоверностью даже эту точность.

Точность результатов решения прикладнх задач в первую очередь зависит от точности исходных данных и выбранной модели условий задачи, но уменьшение погрешностей модели и исходных данных не связано с применением ПМК, и в дальнейшем будем рассматривать лишь погрешности вычислений.

Методические погрешности вычислений возникают, когда отбрасывается часть расчетных выражений с большим или теоретически бесконечным числом членов. Такие погрешности возникают, в частности, при аппроксимации функций бесконечными рядами, произведениями или цепными дробями, численном интегрировании, численной оптимизации и решении задач итерационными методами последовательных приближений. Теоретически методическую составляющую погрешности можно уменьшить, увеличив число членов расчетных формул и, следовательно, число операций. Однако при вычислениях с ограниченной разрядностью операндов методическая погрешность не может быть меньше погрешности результата вычислений, соответствующей его максимальной точности и определяемой разрядностью r мантиссы в машинном представлении чисел, от которой зависит погрешность округления.

В вычислительной математике чаще всего используют три основных способа округления чисел: отбрасывание и несимметричное или симметричное дополнение. При округлении отбрасыванием до r сохраняемых цифр они не зависят от значения отбрасываемой части числа, и в этом случае погрешность округления $\delta \leqslant 1 \cdot 10^{1-r}$ определяется единицей последнего разряда в сохраняемой части числа. При несимметричном округлении по дополнению сохраняемая часть числа увеличивается на единицу последнего разряда, если отбрасываемая часть числа не меньше половины единицы последнего разряда сохраняемой части числа, и не изменяется в противном случае. Симметричное округление по дополнению отличается от несимметричного тем, что при равенстве отбрасываемой части числа половине единицы последнего сохраняемого разряда его нечетное содержание увеличивают на единицу, а четное не изменяют. При округлении по дополнению относительная погрешность $\delta \leqslant 0{,}5 \cdot 10^{1-r}$ и ее предельное значение в 2 раза меньше предельной погрешности округления отбрасыванием.

В ПМК могут использоваться различные варианты этих способов округления. В частности, в подавляющем большинстве массовых отечественных ПМК с компактными входными языками для результатов деления используется округление отбрасыванием, а для результатов выполнения остальных арифметических операций — округление по дополнению с предварительным поразрядным округлением отбрасываемой части результата операции. Например, при сложении

$$10\,000\,000 + 0{,}4444445 = 10\,000\,001; \quad 10\,000\,000 + 0{,}4444444 = 10\,000\,000,$$

а при вычитании

$$10\,000\,000 - 0{,}55 = 10\,000\,000 - (1 - 0{,}45) = 9999999 + 0{,}45 = 10\,000\,000; \quad 10\,000\,000 - 0{,}56 = 20\,000\,000 - (1 - 0{,}44) = 9999999 + 0{,}44 = 9999999.$$

Погрешности округления являются основным источником операционных погрешностей, так как все исходные данные и результаты вычислений округляются в ПМК до r цифр мантиссы и при выполнении нескольких операций погрешности накапливаются вследствие как округления, так и увеличения погрешности результата операций над округленными значениями операндов. Следовательно, при увеличении числа операций методическая погрешность $\delta_м$ результата вычислений уменьшается, но одновременно возрастает операционная погрешность $\delta_{оп}$. Поэтому, если методическую погрешность можно уменьшить, увеличив число операций, приходится выбирать их оптимальное значение, соответствующее минимальному значению полной погрешности δ_Σ (рис. 5.1).

Потеря точности результата вследствие накопления малых погрешностей округления характерна не только для ПМК, но в еще большей степени для универсальных ЭВМ высокой производительности. Если предположить, что ЭВМ выполняет миллион арифметических операций над десятичными числами за секучду, то только из-за округления результатов операций (без учета значительно больших погрешностей результата операции иад округленными операндами) через 1 с погрешность результата возрастает в миллион раз и его шесть последних цифр могут оказаться ошибочными.

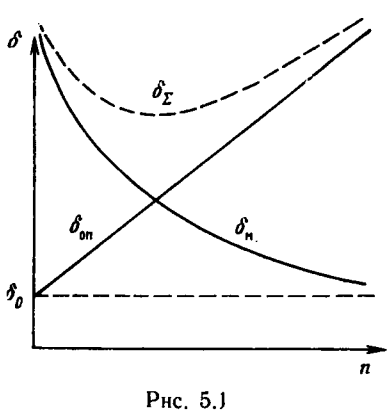


Рис. 5.1

Влияние малых погрешностей округления может привести не только к их постепенному накоплению, но и катастрофической потере точности, при которой результат вычислений стремится к нулю или бесконечности или все цифры результата становятся ошибочными. Например, вычисление по формуле $y = ((10/3 - - 2) \cdot 6 - 8) \times 10^9$ даст результат $y = -2 \cdot 10^{10-r}$, где r — разрядность мантиссы. Между тем точное значе-

ние $y = 0$, в чем легко убедиться, предварительно преобразовав выражение в скобках и представив расчетную формулу в виде $y = ((10 - 6) \cdot 2 - 8) \cdot 10^9 = 0$. Причиной полностью ошибочного результата при вычислениях по исходной расчетной формуле является малая погрешность округления частного 10/3, приводящая к отличию от нуля результата вычислений в скобках, умножаемого затем на большое число.

Когда результат вычислений стремится к бесконечности, регистры памяти данных или операционного стека ПМК переполняются, и обычно выполнение программы автоматически прекращается. Это позволяет обнаружить и устранить ошибку, приводящую к потере точности. Значительно опаснее попадание результата вычислений в область машинного нуля. В этом случае значение отличающегося от нуля промежуточного результата принимается равным нулю независимо от дальнейших операций, причем попадание конечного малого результата вычислений в область машинного нуля в современных ПМК (как и в большинстве универсальных ЭВМ других классов) не диагностируется. Это требует особого внимания программиста, который должен предусмотреть и устранить возможность возникновения подобных ситуаций в процессе вычислений.

Например, при вычислении на ПМК с разрядностью порядка $m = 2$

$$10^{-50} \cdot 10^{-50} \cdot 10^{50} \cdot 10^{50} = (0).$$

Поменяв местами множители, получим другой результат с катастрофической потерей точности:

$$10^{50} \cdot 10^{50} \cdot 10^{-50} \cdot 10^{-50} = (\infty),$$

и только при правильном выборе последовательности множителей получим верный результат

$$10^{50} \cdot 10^{-50} \cdot 10^{50} \cdot 10^{-50} = 1.$$

Следовательно, при умножении на ПМК с ограниченной разрядностью представления чисел нарушается закон коммутативности (результат умножения не зависит от последовательности множителей), а также другие законы арифметики, справедливые для операций над точными числами. Так, в нарушении ассоциативного закона (сумма не зависит от последовательности слагаемых) можно убедиться, изменив при использовании ПМК с разрядностью $r = 8$ мантиссы порядок слагаемых суммы

$0{,}4 + 0{,}4 + 10\,000\,000 = 10\,000\,001$; $0{,}4 + 10\,000\,000 \cdot 0{,}4 + 10\,000\,000$.

Аналогично можно убедиться в нарушении дистрибутивного закона, вычисляя на ПМК с разрядностью порядка $m = 2$:

$$20\,000 \cdot 10^{95} - 18\,000 \cdot 10^{95} = (\infty) \text{ и } (20\,000 - 18\,000) \cdot 10^{95} = 2 \cdot 10^{98}.$$

В связи с отмеченными особенностями вычислений при ограниченной разрядности операндов программист должен тщательно контролировать последовательности выполняемых операций, устраняя ситуации, приводящие к катастрофической потере точности или резкому ее снижению. Подобный анализ необходим и для обеспечения результата вычислений с требуемой точностью.

Результат решения задачи прямыми методами можно рассматривать как функцию $F = F(a_1, \ldots, a_n)$ исходных данных и разультатов

промежуточных вычислений a_i. Такая функция раскладывается в многомерный ряд Тейлора

$$F = F_0 + \sum_{i=1}^{n} \frac{\partial F}{\partial a_i} \Delta a_i + \sum_{j \neq i}^{n} \frac{\partial^2 F}{\partial a_i \, \partial a_j} \Delta a_i \Delta a_j + \ldots, \tag{5.3}$$

где F_0 — значение функции при точных значениях аргументов a_i, а Δa_i — отклонение i-го аргумента от точного значения. При малых отклонениях Δa_i члены этого ряда с высшими производными достаточно малы, и для оценки отклонений функции от точного значения ограничиваются линейными членами ряда. В этом случае абсолютная погрешность результата вычислений

$$\Delta F = F - F_0 = \sum_{i=1}^{n} \frac{\partial F}{\partial a_i} \Delta a_i, \tag{5.4}$$

где Δa_i — абсолютные погрешности исходных данных и результатов промежуточных вычислений, а частные производные результата вычислений по величинам a_i, являются коэффициентами пропорциональности между абсолютными отклонениями Δa_i и результата вычислений. Разделив правую и левую части формулы (5.4) на F, получим выражение для оценки относительной погрешности результата вычислений

$$\delta F = \frac{\Delta F}{F} = \sum_{i=1}^{n} \frac{a_i}{F} \frac{\partial F}{\partial a_i} \frac{\Delta a_i}{a_i} = \sum_{i=1}^{n} S_F(a_i) \, \delta a_i, \tag{5.5}$$

где коэффициент $S_F(a_i) = (a_i/F)\partial F/\partial a_i$ называют чувствительностью (первого порядка) результата вычислений F к относительным изменениям δa_i операндов.

По формулам (5.4) и (5.5) можно оценить чувствительность к изменениям операндов результатов всех операций, автоматически выполняемых ПМК [19—21]. Заметим, что в ПМК для автоматического вычисления элементарных трансцендентных функций используют методы последовательных приближений, методическая погрешность которых зависит от числа операций. В массовых ПМК с небольшим быстродействием для уменьшения затрат времени на автоматическое вычисление элементарных функций выполняется ограниченное число операций с результатом, имеющим методическую погрешность, обычно значительно превышающую погрешность округления (значения погрешностей вычисления элементарных функций указывают в руководствах по применению ПМК). Так как пользователь не может увеличить число операций, чтобы уменьшить методическую погрешность результата автоматического вычисления элементарных функций по встроенным программам операционной системы, то полную погрешность таких результатов можно рассматривать при анализе накопления погрешностей как операционную.

Анализ чувствительности результатов различных операций с помощью формул (5.5) показывает, что одной из самых опасных операций

приводящих к потере точности вычислений, является вычитание близких чисел. В этом случае результат вычитания содержит небольшое число отличающихся от нуля цифр, что и определяет его низкую точность.

Для примера рассмотрим вычисление функции

$$f(x) = x/(\sqrt{10^6+x} - 10^3). \tag{5.6}$$

При вычислениях по этой формуле уменьшение аргумента x приводит к тому, что члены разности в знаменателе становятся близкими по модулю и погрешность результата вычислений быстро возрастает. Действительно, выполняя вычисления по этой формуле на ПМК с разрядностью мантиссы $r = 8$, для значений аргумента $x = 10; 5; 2; 1; 0{,}4; 0{,}2$ соответственно получим $f(x) = 2040{,}8163; 2083{,}3333; 2222{,}2222; 2500; 4000; \infty$. Полученные результаты явно ошибочны, так как при уменьшении аргумента значение рассматриваемой функции, как свидетельствуют результаты математического анализа, асимптотически стремятся к конечному числу. Для получения результата с приемлемой точностью устраним вычитание близких чисел в исходной формуле, домножив ее числитель и знаменатель на сумму членов исходного знаменателя:

$$f(x) = \frac{x(\sqrt{10^6+x}+10^3)}{(\sqrt{10^6+x})^2 - 10^6} = \sqrt{10^6+x}+10^3.$$

При вычислениях по этой формуле для тех же значений аргумента, что и при вычислениях по исходной формуле, соответственно получим

$f(x) = 2000{,}0049; 2000{,}0024; 2000{,}0009; 2000{,}0004; 2000{,}0001; 2000.$

Правильность этих результатов подтверждается и значением функции $f(0) = 2000$.

Приведенный пример подтверждает необходимость тщательного анализа погрешностей результатов вычислений даже по простым расчетным формулам. Накопление погрешностей при вычислениях по сложным формулам приходится анализировать путем последовательного анализа результатов операций, разбивая расчетную формулу на части.

Громоздкость анализа погрешностей несколько упрощается, если использовать графы накопления абсолютных или относительных погрешностей [17, 20] с более наглядным отображением связей между ними. При таком анализе можно, например, определить относительную погрешность результата вычислений по формуле (5.6)

$$\delta y = \frac{3\sqrt{x+10^6}}{2(\sqrt{x+10^6}-10^3)} \cdot 10^{-7}.$$

предположив, что исходные данные заданы точно, а погрешности округления $\delta = 1 \cdot 10^{-7}$ одинаковы для результатов всех операций.

Анализируя подобным образом накопление погрешностей при вычислениях по преобразованиой формуле с исключением вычитания близких чисел, получаем отиосительиую погрешность результата

$$y = (1 + \sqrt{x+10^6}/2(\sqrt{x+10^6}+10^3)) \cdot 10^{-7},$$

которая при $x \to 0$ стремится к предельному значению $1{,}5 \cdot 10^{-7}$, что подтверждается и результатамн вычислений.

Решение задач итерационными методами сводится к многократному выполнению однотипных последовательностей операций (процедур). В этих случаях необходимо оценить лишь накопление погрешностей на одной итерации, так как в большинстве методов дальнейшего накопления ошибок не происходит. Однако может получиться, что погрешности двух очередных итераций несколько отличаются и при приближении к погрешности округления их разность будет конечной величиной. В таком случае условие максимальной точности, соответствующее попаданию в область машинного нуля разности значений результатов вычислений на двух очередных итерациях, не будет выполняться, и при выборе этого условия для выхода из цикла возникает «зацикливание» вычислительного процесса. Чтобы исключить эту возможность, выбор условий выхода из цикла, обеспечивающий получение максимально точного результата, должен тщательно оцениваться.

Специфическими являются требования к выбору приращений аргумента (шага) при решении численными методами нелинейных уравнений, численной оптимизации и в особенности при численном интегрировании и решении дифференциальных уравнений. В последнем случае результат вычислений на каждом шаге зависит от результатов вычислений на предыдущих шагах, что может привести к накоплению ошибки, а при выборе большого шага — к катастрофической потере точности. В то же время выбор малого шага приводит к быстрому накоплению операционных погрешностей и увеличению времени счета.

Значительные трудности связаны с оценкой и обеспечением требуемой точности уравнений и особенно систем уравнений. Точность решения уравнений оценивают по величине невязки — разности правой и левой частей уравнений при подстановке вычисленных значений корней. Точность решения систем уравнений оценивают по норме вектора невязок уравнений системы.

Нормой вектора с n элементами x_i называют величину

$$\|x\|_p = \left(\sum_{i=1}^{n} x_i^p \right)^{1/p},$$

где p — действительное число от 1 до ∞, причем $\|x\|_\infty = \max\limits_{1 \leqslant i \leqslant n} |x_i|$.

Для оценки точности решения систем уравнений чаще всего используют норму

$$\|x\|_1 = \sum_{i=1}^{n} |x_i| \text{ или } \|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

(эвклидову норму).

Значительные трудности возникают при вычислении определителей плохо обусловленных квадратных матриц и систем линейных уравнений. (*Плохо обусловленной* называют задачу, погрешности результатов решения которой очень сильно зависят от погрешностей исходных данных.) При плохой обусловленности задачи полученное решение

может оказаться полностью ошибочным, причем во многих случаях норму вектора невязок нельзя использовать для оценки погрешности результатов плохо обусловленных задач. Причиной плохой обусловленности задачи может оказаться как близость к неустойчивому состоянию моделируемого матрицей или системой уравнений физического объекта, так и неудачный выбор математической модели. Степень обусловленности квадратной матрицы A коэффициентов и приближенно системы уравнений с такой матрицей коэффициентов оценивают числом, называемым мерой обусловленности:

$$\text{cond } A = \|A\| \cdot \|A^{-1}\|,$$

причем матрица A или система с матрицей коэффициентов A тем хуже обусловлены, чем больше это число.

Норму матрицы обычно вычисляют по формулам

$$\|A\|_1 = \max_{1 \leqslant i \leqslant n} \sum_{i=1}^{n} |a_{ij}| \text{ или } \|A\|_2 = \max_{1 \leqslant i \leqslant n} \sum_{j=1}^{n} |a_{ij}|.$$

Например, для системы уравнений $x_1 + x_2 = 3$; $x_1 + 5x_2 = 11$ с корнями $x_1 = 1$, $x_2 = 2$ матрицы

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 5 \end{bmatrix} \qquad A^{-1} = \begin{bmatrix} 1{,}25 & 0{,}25 \\ 0{,}25 & 0{,}25 \end{bmatrix}$$

и мера обусловленности cond $A = 6 \cdot 1{,}5 = 9$. Следовательно, система уравнений хорошо обусловлена, и небольшие изменения ее коэффициентов не приводят к значительным изменениям корней.

Для системы уравнений

$$\begin{bmatrix} 1 & 1{,}9999 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 4{,}9998 \\ 5 \end{bmatrix}$$

с корнями $x_1 = 1$, $x_2 = 2$ мера обусловленности cond$A = 3{,}9999 \times 39999 = 159992$, и, следовательно, можно ожидать, что при небольших изменениях коэффициентов значения корней могут существенно изменяться.

Для определенности рассматриваемые системы из двух линейных уравнений

$$a_{11} x_1 + a_{12} x_2 = q_1, \quad a_{21} x_1 + a_{22} x_2 = q_2$$

будем решать методом исключений по формулам

$$x_2 = (a_{11} q_2 - a_{12} q_1) / (a_{11} a_{22} - a_{12} a_{21});$$

$$x_1 = (q_1 - a_{12} x_2) / a_{11}.$$

При вычислении по этим формулам корней рассматриваемой плохо обусловленной системы уравнений на ПМК с разрядностью $r = 8$ мантиссы получим точные значения корней $x_1 = 1$, $x_2 = 2$.

Изменим на 0,005 % коэффициент a_{21}, изменив соответственно q_2, чтобы сохранить значения корней:

$$x_1 + 1{,}9999 x_2 = 4{,}9998; \quad 1{,}00005 x_1 + 2 x_2 = 5{,}00005.$$

Решая эту систему по приведенным формулам на том же ПМК, получаем значения корней $x_1 = 2{,}9999$, $x_2 = 1$. Подставив эти значения в уравнения системы, получим их невязки $\alpha_1 = 0$, $\alpha_2 = 1 \cdot 10^{-7}$. Хотя эти невязки, как и их векторная норма, достаточно малы, что должно свидетельствовать о высокой точ-

ности вычисления корней, известные нам точные значения корней ни в одном знаке не совпадают с вычисленными. Действительно, вычисление корней рассматриваемой системы уравнений при удвоенной разрядности с устранением погрешностей округления приводит к точным значениям $x_1 = 1$, $x_2 = 2$. Следовательно, для плохо обусловленных систем уравнений векторная норма невязок не может служить критерием точности корней.

Так как режим вычислений с повышенной точностью в ПМК не предусмотрен, решение плохо обусловленных систем уравнений требует особого внимания. Имеется и множество других более частных математических задач, при решении которых необходим особенно тщательный анализ влияния погрешностей на результаты вычислений.

5.4. ОПТИМИЗАЦИЯ ПРИКЛАДНЫХ ПРОГРАММ

Составление оптимальной программы решения сложной прикладной задачи является многоступенчатым итерационным процессом, начинающимся с выбора математической модели условий задачи и метода ее решения, в наибольшей степени соответствующих возможности используемого ПМК и особенностям его входного языка. Этот выбор часто приходится многократно уточнять как при составлении алгоритма решения задачи, так и при его представлении программой на входном языке и ее отладке.

Профессиональные программисты при составлении сложных программ операционной системы на внутренних языках ЭВМ или сложных прикладных программ на входных языках обычно руководствуются принципами структурированного программирования. В соответствии с этими принципами программу составляют из типовых структурных элементов (модулей), в основном представляющих собой типовые неразветвленные фрагменты программ, циклы и разветвления. Такие модули должны быть максимально независимыми с минимальным количеством данных, получаемых от других модулей и передаваемых им, а связи между модулями должны быть как можно более простыми. В качестве одного из основных критериев степени структурированности программы многие теоретики рассматривают отсутствие или во всяком случае минимальное число безусловных переходов между модулями.

Структурирование прикладных программ по существу основано на разбиении сложной прикладной задачи на ряд типовых математических подзадач, решение которых обеспечивается модулями структурированной программы. Использование типовых моделей, оптимизированных по требуемому критерию качества, позволяет значительно уменьшить затраты времени на составление прикладной программы, что отвечает основному критерию ее качества — минимум затрат времени на решение задачи.

Библиотеки пользователя, содержащие оптимизированные программы решения типовых математических задач, и являются библиотеками моделей, из которых формируются структурированные программы решения сложных прикладных задач. Использование таких библио-

течных программ-модулей аналогично использованию программ решения стандартных задач в системах, например, автоматизированного проектирования (САПР), хранящихся в накопителях информации с вызовом нужной программы по ее имени. Для эффективного использования библиотечных программ в качестве модулей структурированных программ решения прикладных задач они должны быть предварительно оптимизированы по требуемым критериям оптимальности и обеспечивать согласованный обмен данными.

Следует добавить, что методы структурированного программирования основаны на теоретических исследованиях и накопленном опыте программирования универсальных ЭВМ высокой производительности с большой емкостью запоминающих устройств. При приложении этих методов к программированию прикладных задач на ПМК приходится учитывать ограниченную емкость его оперативной памяти.

Поэтому библиотечные программы пользователя, которые могут быть применены в качестве модулей прикладных программ, должны быть прежде всего оптимизированы по критерию минимальной длины, но должны иметь и варианты с минимальным временем счета для обеспечения соответствующего критерия оптимальности конкретной прикладной программы. Так как при решении прикладных задач могут предъявляться различные требования к точности результатов, то библиотечные программы-модули должны храниться также в различных вариантах, обеспечивающих требуемую точность. Попытка ограничиться набором программ, обеспечивающих максимальную точность, приводит к излишним потерям времени при решении прикладных задач с относительно небольшой требуемой точностью результатов.

Критерий минимальной длины библиотечных программ-модулей особенно актуален, когда общая длина модулей, используемых для решения прикладной задачи, оказывается большой и приходится использовать несколько последовательно выполняемых программ (пакет), что приводит к значительным затратам времени. Эти затраты уменьшаются при размещении пакета в накопителе информации, но в этом случае программы пакета должны быть тщательно согласованы по именам или адресам данных, передаваемых следующей программе.

Следует добавить, что при использовании внешнего накопителя может быть значительно повышено и качество многих библиотечных программ-модулей. Например, для аппроксимации большинства специальных функций используют бесконечные ряды, произведения или цепные дроби с вычислением функции методами последовательных приближений, что связано со значительными затратами времени, так как для уменьшения затрат времени на ввод данных в таких аппроксимирующих выражениях используют однозначные или во всяком случае немногозначные коэффициенты. Между тем можно использовать эффективную аппроксимацию специальных функций целыми или дробными рациональными функциями с многозначными коэффициентами. Если хранить такие коэффициенты в накопителе информации, то затраты времени на их ввод существенно сократятся и ока-

жется возможным использование более эффективной аппроксимации специальных функций рациональными выражениями.

В общем случае при программировании на ПМК решения сложной прикладной задачи целесообразно использовать следующую методику. После выбора математической модели и метода решения задачи составляют алгоритм ее решения с шагами, соответствующими решению подзадач, на которые разбивается сложная прикладная задача. Затем каждый такой шаг отображают модулем программы на алгоритмическом входном языке или при необходимости алгоритмами решения подзадач с учетом их реализации программами-модулями на компактном входном языке. При этом следует стремиться к тому, чтобы алгоритм решения задачи содержал как можно больше однотипных фрагментов, отображающих одинаковые процедуры преобразования информации. После этого каждый фрагмент или шаг алгоритма отображают модулем или фрагментом программы на входном языке, вначале не заботясь об общей длине фрагментов. Минимизировав пересылки данных между модулями, используют все стандартные приемы и особенности входного языка, обеспечивающие минимизацию длины отдельных фрагментов и программы в целом, учитывая и требования минимального времени счета.

Основными средствами минимизации длины программы, предусмотренными практически во всех входных языках, являются обращения к подпрограммам и организация циклов. Если в программе несколько одинаковых процедур, то целесообразно вынести один фрагмент с такой процедурой в подпрограмму, оканчивающуюся оператором возврата из подпрограммы, а на месте фрагментов в исходной программе записать операторы обращения к подпрограмме. Так как время выполнения подпрограмм возрастает за счет затрат времени на переходы, то организация обращений к подпрограммам оказывается целесообразной лишь при допустимом увеличении времени счета и только в том случае, если введение подпрограмм уменьшает общую длину программы.

Если в исходной программе на компактном входном языке n одинаковых фрагментов по m шагов, то при вынесении такого фрагмента в подпрограмму длина программы уменьшается при выполнении условия $mn > m + 1 + 2n$; подпрограмма вместе с оператором возврата занимает $m + 1$ шагов, а на месте каждого фрагмента исходной программы, вынесенного в подпрограмму, записывается занимающий два шага оператор обращения к подпрограмме. Если это условие не выполняется, то организация обращения к подпрограмме ухудшает качество программы, так как увеличивается время счета и длина программы.

Подобная оценка может быть использована и для организации подпрограмм в программах на алгоритмических языках. Дополнительное сокращение длины программы в обоих случаях часто достигается при обращении из подпрограммы к подпрограмме, но время счета в таких случаях возрастает еще больше.

Длину программы, содержащую последовательность из n одинаковых фрагментов, можно сократить, охватив один из таких фрагментов циклом, вычисления в котором повторяются n раз. В программах на алгоритмических языках для этой цели можно использовать как операторы цикла общего вида: «пока <условие>, выполнять <тело цикла>» или «повторять <тело цикла>, пока <условие>», так и специализированные операторы цикла вида «для $I = n$ до 1 выполнить <тело цикла>», где телом цикла является многократно выполняемый фрагмент программы.

В программах на компактных входных языках массовых ПМК расширяющегося ряда для этой цели предназначены операторы цикла вида LN ab, где $N = 0, 1, 2$ или 3 — адрес регистра памяти, в который записывается число n итераций, уменьшающееся на единицу после каждого выполнения оператора цикла, а ab — адрес начала тела цикла, которому передается управление, когда содержимое регистра N больше единицы. При использовании такого оператора, который можно описать предложением «выполнить <тело цикла> для PN:=PN — — 1 > 1» с телом цикла длиной m шагов, программа сокращается при практически всегда выполняемом условии $mn > m + 2$, где n — число однотипных последовательных фрагментов в исходной программе.

Если регистры 0, 1, 2 и 3 не могут быть использованы для организации цикла, то его можно организовать на любом регистре памяти с адресом $N \leqslant 3$ с помощью следующего фрагмента:

(↑) → <тело цикла> ИПN 1 — ПN x = 0 ab →

В таком фрагменте содержимое счетчика итераций на регистре памяти с адресом N уменьшается на единицу после каждой итерации и после выполнения n итераций оператор условного перехода выполняет операцию выхода из цикла. Операторы поворота стека в начале и конце фрагмента (этот оператор в начале тела цикла имеет адрес ab) обеспечивают устранение из входного (индикаторного) регистра содержимого регистра-счетчика, а указанный в скобках оператор ↑ в конце предыдущей части программы обеспечивает ввод в тело цикла результата выполнения предыдущей части программы. Этот оператор не нужен, если операции в теле цикла не связаны непосредственно с содержимым входного регистра операционного стыка, равным результату выполнения предыдущей части программы.

При использовании подобного составного оператора цикла условие уменьшения длины программы при введении цикла описывается неравенством $mn > m + 8$ с учетом всех указанных операторов входного языка, используемых при организации цикла, кроме оператора ↑, который может оказаться ненужным. Если результат операции в цикле заносится в память данных, то могут оказаться ненужными и операторы поворота стека, а условие уменьшения длины программы в этом случае менее жестко ($mn > m + 6$), если только не увеличивается длина тела цикла.

При решении задач методами последовательных приближений с заранее неизвестным числом итераций использование циклов стано-

вится необходимым. В этом случае используются операторы цикла общего вида, реализуемые на компактных входных языках фрагментами вида

<тело цикла> $x \geqslant 0$ *ab*,

где *ab* — адрес начала тела цикла, содержащего процедуру оценки достигнутой точности результата.

Точность результата вычислений в цикле определяется разностью двух очередных значений результатов с условием выхода из цикла

$$|x_{i+1} - x_i| \leqslant \varepsilon \text{ или } |(x_{i+1} - x_i)|/x_{i+1}| \leqslant \delta$$

при заданной предельной абсолютной ε или относительной δ погрешности результата вычислений.

Максимально достижимая точность вычислений в цикле определяется разрядностью r мантиссы машинного представления чисел и соответствует в общем случае предельной погрешности округления. Если разность очередных значений результатов вычислений в цикле меньше этой погрешности, то она попадает в область машинного нуля, и в качестве условия выхода из цикла при достижении максимальной точности можно принять неравенства

$$|x_{i+1} - x_i| = 0 \text{ или } |(x_{i+1} - x_i)/x_{i+1}| = 0.$$

Эти условия, как отмечалось, могут оказаться невыполненными, если погрешность вычислений на двух очередных итерациях отличается не меньше, чем на единицу последнего разряда мантиссы. Выполнение программы может «зациклиться» и в тех случаях, когда результат вычислений в теле цикла (например, при использовании стандартных элементарных функций, вычисляемых автоматически со значительной погрешностью) имеют большую и изменяющуюся при изменении аргумента погрешность. В подобных случаях необходимо тщательно исследовать погрешности вычислений в теле цикла и при необходимости вместо приведенного условия максимальной точности выбрать условие выхода из цикла при несколько большей предельной погрешности.

Длину программы на компактном входном языке можно сократить, рационально используя возможные разнообразные способы реализации одних и тех же операторов алгоритма. Следует учитывать, что уменьшение длины программы складывается из небольших уменьшений числа шагов в отдельных фрагментах, а если длина оптимизируемой программы лишь на несколько шагов превышает число ячеек программной памяти, выигрыш от сокращения длины программы и устранения необходимости использования пакета программы оказывается значительным. Если при уменьшении длины отдельных фрагментов удается устранить и операторы безусловных переходов (что соответствует требованиям структурирования программ), то заметно сократится и время выполнения программы.

Для примера рассмотрим фрагмент ранее рассмотренного алгоритма решения алгебраического уравнения:

5. Если $\alpha \geqslant 0$, то перейти к шагу 8.
6. Принять $x_1 = \alpha - \sqrt{D}$.
7. Перейти к шагу 9.
8. Принять $x_1 = \alpha + \sqrt{D}$.

Этот фрагмент на входном языке ПМК расширяющегося ряда непосредственно реализуется фрагментом программы (с адресом 20 начального шага)

x<0 26 ↔ — БП 27 +...

Однако этот же фрагмент алгоритма можно представить и более коротким фрагментом программы (см. табл. 5.1)

x<0 25 ↔ — 0 + ...,

не содержащим оператора безусловного перехода.

Следует отметить еще один прием сокращения длины программы — «жаргонное» использование элементов входного языка, не предусмотренное руководствами по применению ПМК. Например, если в операторах прямого или косвенного обращения к памяти данных набор адресов 0, 1, 2, 3 и 4 заменить нажатием соответственно клавиш ввода символов +, —, ×, ÷ и ↔, то будут выполняться те же самые операторы обращения к памяти, что можно использовать для упрощения вызова нужных фрагментов составных программ [19, 21, 23]. В программах на входном языке ПМК семейства «Электроника МК-54» при вводе оператора КИП- ↑ вместо оператора КИП0 не происходит уменьшение на единицу содержимого регистра 0 при выполнении такого «усовершенствованного» оператора, что удобно использовать в различных ситуациях. Подобные находки имеются в арсенале каждого пользователя, имеющего опыт программирования ПМК с компактными входными языками. В то же время входные алгоритмические языки отличаются более жесткими лексическими правилами, ограничивающими разнообразие вариантов программной реализации операторов алгоритма и тем самым уменьшающими затраты времени на поиск наилучших вариантов.

Иногда решение задачи приходится реализовать с помощью пакета программ вследствие большого числа входных данных или результатов промежуточных вычислений, не размещающихся в памяти данных. В таких случаях часто удается уменьшить требуемое число регистров данных, используя различные способы, и в первую очередь нормирование чисел.

Нормирование чисел часто используют при составлении программ для универсальных ЭВМ, чтобы предупреждать возможность выхода результатов вычислений из области машинного представления чисел, так как многие традиционные алгоритмические языки предусматривают относительно небольшой диапазон представления чисел в показательной форме. Подобное нормирование иногда приходится исполь-

зовать и при составлении программ для ПМК, но чаще цель нормирования — приведение коэффициентов или переменных к единице или другому немногозначному значению, которое можно записать в текст программы. В частности, число регистров памяти для хранения коэффициентов степенного многочлена уменьшают на единицу при замене переменной (аргумента) x переменной z путем подстановки $x = z/\sqrt{a_n}$, где a_n — коэффициент при старшей степени x. Если вычисляются корни многочлена, то достаточно нормировать его делением на один из коэффициентов (например, на коэффициент a_n), что не изменяет значения корней с точностью, определяемой погрешностями округления при делении коэффициентов. Аналогично значение дробно-рациональной функции (отношения степенных многочленов) не изменяется при делении ее числителя и знаменателя на один из коэффициентов. Еще большего уменьшения требуемого числа регистров памяти можно достичь при нормировании аргументов и функций различных других выражений [20].

Требуемое число регистров памяти можно также уменьшить, используя одни и те же регистры в качестве буферных для временного хранения различных последовательно вычисляемых промежуточных результатов, а регистры операционного стека для хранения текущих значений аргумента вычисляемых функций или части исходных данных. Во многих случаях требуемое число регистров данных можно уменьшить, организовав ввод части исходных данных и вывод части результатов с автоматическим остановом программы в соответствующих местах ее текста. Заметим, что в некоторых ПМК предусмотрен специальный оператор PAUSE, обеспечивающий автоматический останов программы на несколько секунд для регистрации или контроля промежуточных результатов вычислений.

Требования к памяти данных можно также ослабить, записав однозначные или даже многозначные коэффициенты непосредственно в текст программы. В этих случаях целесообразно, когда это можно, использовать символы операций для уменьшения ввода таких данных или длины их описания. Например, вместо постоянной e = 2,7182818 в текст программы можно ввести короткий фрагмент ее вычисления 1 e^x, а вместо чисел 1,4142135; 3,3333333 · 10^{-1} или 6,25 · 10^{-2} — соответственно более короткие фрагменты 2 $\sqrt{}$, 3 1/x или 4 x^2 1/x.

Повышение автоматизации ввода исходных данных или вывода результатов должно быть согласовано с основной задачей оптимизации — уменьшением полных затрат времени на решение задачи. Поэтому начальный фрагмент программы, подобный использованному в табл. 5.1 или 5,2, не всегда целесообразен; экономится лишь время нажатия трех клавиш, тогда как затраты времени на ввод более длинного фрагмента подобного типа и на его исполнение могут оказаться бо́льшими полученной экономии. В то же время при многократном повторении вычислений по подобной программе введение такого фрагмента может быть выгодным с точки зрения уменьшения общих затрат

времени. В этих же случаях целесообразно заменять оператор С/П в конце программы фрагментом С/П БП 00 или С/П БП *ab*, обеспечивающим при повторных пусках программы нажатием только клавиши С/П выполнение программы с начального шага или с шага *ab*.

В случае затруднений в теоретической оценке времени выполнения различных вариантов программ или их фрагментов, как и затруднений в оценке погрешностей результатов вычислений и способов обеспечения требуемой точности, целесообразно решать эти задачи экспериментально в процессе отладки программы.

5.5. ОТЛАДКА ПРИКЛАДНЫХ ПРОГРАММ

Отладка прикладной программы заключается в ее проверке на ЭВМ с целью устранения ошибок и при необходимости дополнительной оптимизации программы по получаемым экспериментальным данным. При составлении прикладной программы для универсальной ЭВМ высокой производительности в связи с высокой стоимостью машинного времени к отладке программы переходят лишь после ее всесторонней теоретической оптимизации. При отладке прикладной программы для ПМК с низкой стоимостью машинного времени выгодно заменить сложные теоретические оценки непосредственными экспериментальными данными, что позволит уменьшить и общие затраты времени на решение задачи.

При отладке готовых программ в первую очередь проверяют правильность выполнения программы по контрольному примеру и в случае несоответствия получаемых и контрольных результатов отыскивают и устраняют причины расхождения. Простейшими являются синтаксические и семантические ошибки в тексте программы, составленной на входном языке и введенной в оперативную память. При использовании ПМК с алгоритмическими входными языками многие из синтаксических ошибок диагностируются операционной системой, автоматически останавливающей в случае обнаружения такой ошибки выполнение программы и вырабатывающей сообщение ERROR, обычно сопровождаемое номером класса или типа ошибок.

Диагностируемые синтаксические ошибки в первую очередь связаны с ошибочным набором слов или операторов из элементов алфавита, ошибочным размещением или пропуском элементов алфавита в тексте. Типичным примером является легко диагностируемый пропуск одной из пары закрывающей и открывающей скобок. Обычно диагностируются и ошибки, связанные с пропуском имен переменных в местах текста, где они должны быть объявлены, например в записи

```
10 FOR I TO 100 ... 20 NEXT;
```

с пропущенным именем переменной после ключевых слов FOR и NEXT. В большинстве ПМК диагностируются ошибки, связанные с обозначением одной и той же переменной различными именами, например в записи

```
10 FOR I = 1 TO 100 ... 20 NEXT A
```

в заголовке и окончании оператора цикла должно указываться одинаковое имя изменяющейся переменной.

Не диагностируются ошибки, допускаемые синтаксическими правилами входного языка и являющиеся смысловыми (семантическими), в особенности ошибка в операторе, символе операции или имени переменной, если они не противоречат правилам входного языка. Особенно опасны контекстовые синтаксические ошибки, когда допустимость использования элемента алфавита зависит от его места в тексте, причем правила входного языка не предусматривают формальных запретов на использование различных элементов алфавита, изменяющих смысл соответствующей части текста. Типичным примером подобной ошибки для европейских пользователей ПМК с алгоритмическим входным языком является запись действительного числа с запятой после целой части (например, 2,35Е—2), тогда как в большинстве алгоритмических языков правила записи чисел предусматривают использование в качестве разделительного знака точки (например, 2.35Е—2). В этом случае при использовании запятой, применяемой в алгоритмических языках для разделения элементов числовых или символьных массивов, диагностическая система примет такую запись, как массив из двух чисел (например, 2 и 35Е—2) с последующим ошибочным результатом выполнения программы. В программах на компактном входном языке подобная ошибка исключается, так как алфавиты таких входных языков содержат только один десятичный разделительный знак (точку или запятую).

Сообщения об ошибке вырабатываются операционными системами многих ПМК с алгоритмическими языками и в других критических случаях, например при неверной работе внешних устройств вследствие снижения напряжения питания, при переполнении оперативной памяти или отдельных запоминающих устройств — стека возврата из подпрограмм или даже операционного стека.

При выполнении контрольных примеров почти во всех случаях диагностируется переполнение входного регистра операционного стека, в частности, при делении на нуль. Обычно диагностируется и попадание аргумента вне области определения стандартных функций, например при попытке извлечения корня квадратного или логарифма отрицательного числа. В некоторых случаях в режиме автоматического выполнения программы переполнение может не диагностироваться, если оно устраняется при выполнении ближайшей операции (примеры приведены в гл. 4).

Если при выполнении контрольного примера результат выполнения программы совпадает с требуемым, то текст программ не содержит ошибок и дальнейшая отладка целесообразна лишь для подбора более оптимального ее варианта. Если результат выполнения программы не совпадает с контрольным, то требуется устранить причину этого расхождения. Если оно связано лишь с обеспечением требуемой точности, то следует попытаться найти вариант текста программы (например, изменением последовательности операций или устранением

наиболее опасных для накопления погрешностей операций) или, если этого недостаточно, выбрать другой метод решения задачи.

В случае грубой ошибки, когда результат выполнения программы полностью отличается от контрольного, причиной обычно является семантическая ошибка в тексте программы. Для ее обнаружения текст программы разбивают на самостоятельные фрагменты и определяют по контрольному примеру значения промежуточных результатов, соответствующих выполнению этих фрагментов. Поочередно записывая в конце каждого такого фрагмента оператор С/П, выполняют фрагмент и сверяют полученный результат с контрольным. При их совпадении восстанавливают шаг программы, временно занятый оператором С/П, и проверяют следующий фрагмент. При обнаружении содержащего ошибку фрагмента его выполнение проверяют в режиме пошагового прохождения программы, предварительно определив содержимое регистра индикации на каждом шаге проверяемого фрагмента при выполнении контрольного примера. Нажимая при использовании ПМК отечественного производства клавишу ПП, сверяют индицируемые результаты выполнения каждого шага с контрольными до обнаружения ошибки.

Чтобы определить контрольные значения содержимого регистра индикации, обычно приходится исследовать перемещение содержимого всех регистров операционного стека в процессе выполнения рассматриваемого фрагмента или даже всей программы. С этой целью можно составить таблицу содержимого регистров операционного стека на каждом шаге выполняемой программы, соответствующую последним столбцам табл. 5.1. Примером может служить также приведенная в табл. 5.4 программа решения алгебраического уравнения второй степени без обращения к памяти данных, при составлении которой приходится тщательно контролировать перемещение информации в операционном стеке.

При проверке отдельных фрагментов или даже небольших программ составление подобной таблицы для экономии времени целесообразно заменить записью чернового текста фрагмента или программы по строкам с обозначением содержимого операционных регистров над каждым шагом программы. Например, для начальной части программы из табл. 5.4 такая запись может быть символической:

X1	0	a_2	a_2	0	a_1/a_2	$-a_1/a_2$	2	2	α	α^2	α	
T	0	0	0	a_0	0	a_0	a_0	a_0	a_0	α^2	α	
Z	a_0	0	a_0	a_2	a_0	a_2	a_0	a_2	a_2	a_0	α^2	
Y	a_1	a_0	a_1/a_2	a_1/a_2	a_2	$-a_1/a_2$	a_2	α	α	a_2	a_0	
X	a_2	a_1/a_2	a_2		$-a_1/a_2$	2	α	α	α^2	α	a_2	
		÷	Bx	↔	/—/	2	÷	↑	x²	→	→	...

Подобная запись с символами содержимого операционного стека удобна для составления программы. Для проверки составленной программы символы переменных следует заменить числами из контрольного примера и проверять на каждом шаге или только содержимое регистра X,

Таблица 5.4

Решение уравнения $a_2x^2+a_1x+a_0=0$ для действительных корней на ПМК расширяющегося ряда без обращения к памяти данных

Адрес	Шаг	Код	X	Y	Z	T	X1
			a_2	a_1	a_0	0	0
00	÷	13	a_1/a_2	a_0	0	0	a_2
01	Вх	0	a_2	a_1 a_2	a_0	0	a_2
02	↔	14	a_1/a_2	a_2	a_0	0	a_2
03	+/−	01	$-a_1/a_2$	a_2	a_0	0	a_2
04	2	02	2	$-a_1/a_2$	a_2	a_0	$-a_1/a_2$
05	÷	13	α	a_2	a_0	a_0	2
06	↑	0E	α	α	a_2	a_0	2
07	x^2	22	α^2	α	a_2	a_0	2
08	→	25	α	a_2	a_0	α^2	α^2
09	→	25	a_2	a_0	α^2	α^2	α
10	÷	13	β	α^2	α	α	a_2
11	↑	0E	β	β	α^2	α	a_2
12	→	25	β	α^2	α	β	β
13	−	11	D	α	β	β	β
14	↑	21	$\sqrt{D}$	α	β	β	D
15	↔	14	α	$\sqrt{D}$	β	β	$\sqrt{D}$
16	x < 0	5C	α	$\sqrt{D}$	β	β	$\sqrt{D}$
17	21	21					
18	↔	14	$\sqrt{D}$	α	β	β	α
19	−	11	x_1	β	β	β	$\sqrt{D}$
20	0	00	0	x_1	β	β	$\sqrt{D}$
21	+	10	x_1	β	β	β	x_1
22	÷	13	x_2	β	β	β	x_1
23	Вх	0	x_1	x_2	β	β	x_1
24	С/П	50	x_1	x_2	β	β	x_1

Инструкция. a_2=РХ, a_1=РУ, a_0=РZ В/О С/П РХ=x_1, РУ=x_2.

или в необходимых случаях (например, при обнаружении несоответствия индицируемого содержимого регистра X контрольному) содержимое всех регистров операционного стека, используя оператор поворота стека →.

Пошаговая проверка программы позволяет обнаружить и катастрофическую потерю точности при возникновении «фатальной» ошиб-

ки, не диагностируемой в процессе автоматического выполнения программы. Например, при реализации вычислений по формуле $A = \ln|xe^{x^2}|$ фрагментом B ↑ x² eˣ × x² √ ln на входном языке ПМК, не содержащем оператора определения модуля числа, для значения текущей переменной $x = 11$ в автоматическом режиме вычислений получим A —106, 86061, тогда как правильное значение результата $A =$ 123,39789. Ошибка, диагностируемая в режиме пошагового выполнения программы с индикацией символа ЕГГОГ, но не диагностируемая в режиме автоматического выполнения программы (что приводит к ошибочному результату), связана с переполнением, возникающим при выполнении фрагмента x² √ для выделения модуля числа.

Еще более опасны ситуации, связанные с попаданием промежуточного конечного результата в область машинного нуля: независимо от последующих операций ПМК воспринимает этот результат как нуль. Попадание промежуточных результатов в область машинного нуля не диагностируется современными ПМК, но при разработке перспективных карманных ЭВМ учитываются рекомендации компетентных исследовательских организаций о необходимости вывода на дисплей указателей попадания промежуточных результатов в область машинных нуля и бесконечности, что позволит пользователю при отладке или использовании программы избежать ошибочных результатов.

Весьма существенным для отладки и сопровождения библиотечных программ является выбор контрольного примера, который должен отвечать следующим требованиям:

значения аргумента должны выбираться во всех областях аргумента, реализуемых различными частями программы, и обеспечивать проверку правильности выполнения всех переходов в программе;

значения исходных данных должны выбираться так, чтобы не возникали неоднозначности в процессе выполнения программы, связанные с пропусками шагов программы или совпадением результатов при ошибочной записи символов операций.

В частности, не следует выбирать в контрольных примерах в качестве исходного значения 1, так как $1 = 1 \times 1 = 1^2 = \sqrt{1}$ и при ошибке в записи символа операции она не будет обнаружена. Например, для проверки правильности вычисления степенного многочлена удобно принять значение аргумента $x = 1$, так как в этом случае результат равен сумме всех коэффициентов многочлена. Однако этот результат не изменится и при пропуске символа умножения на значение аргумента и подобная ошибка не будет обнаружена. Во многих случаях нежелателен и выбор $x = 2$, так как $2 + 2 = 2 \cdot 2 = 2^2$, а часто и выбор исходных чисел целыми (если это не предусмотрено условиями задачи). В таких случаях можно выбрать в качестве исходных многозначные числа с одинаковыми цифрами, например 1111111 или 7,7777777, или $x = \pi/4$ и т. п., что облегчит и проверку точности результатов вычислений.

При проверке и оценке точности вычисляемых результатов в процессе отладки программы следует выбирать контрольные результаты из заведомо достоверных источников, например таблиц функций с требуемым числом верных цифр, или вычислять методом, отличающимся от используемого в программе, с достоверной точностью контрольного результата. Следует добавить, что составление и оптимизацию программ с высокой точностью результатов вычислений целесообразно совместить с экспериментальной проверкой выбираемых решений.

Для примера рассмотрим составление и отладку на ПМК программы вычисления специальных функций, называемых интегралом вероятности*

$$\Phi(x) = \sqrt{2/\pi} \int_0^x e^{-t^2/2}\, dt$$

и функции

$$1 - \Phi(x) = \sqrt{2/\pi} \int_0^\infty e^{-t^2/2}\, dt$$

с точностью порядка 7 верных цифр на ПМК расширяющегося ряда.

Непосредственное определение значений заданных функций методами численного интегрирования с требуемой точностью связано со значительными затратами времени. Поэтому используем известную аппроксимацию интеграла вероятности разложением в бесконечный ряд

$$\Phi(x) = \sqrt{2/\pi}\left(x - \frac{x^3}{1!2\cdot 3} + \frac{x^5}{2!4\cdot 5} - \frac{x^7}{3!7\cdot 8} + \ldots\right).$$

Представим этот ряд в виде

$$\Phi(x) = \sqrt{2/\pi} \sum_{k=0}^{\infty} \varphi_k/(2k+1),$$

где $\varphi_0 = x$ и $\varphi_k = -\varphi_{k-1}/2k$, и выделим для формирования номера члена ряда k регистр-счетчик 4, а для накопления суммы членов ряда, значений очередного члена ряда и квадрата аргумента соответственно регистры 7, 8 и 9. Тогда подготовительные операции для вводимого в регистр X перед пуском программы значения аргумента можно представить начальным фрагментом программы

П7 П8 x^2 П9 Сх П4

Для достижения максимальной точности выберем в качестве условия выхода из цикла, тело которого содержит операции формирования очередного члена ряда и его сложение с предыдущей суммой, попадание в область машинного нуля значения очередного члена ряда. Тогда вычисление $\Phi(x)$ по условию максимальной точности можно описать фрагментом

КИП4	ИП7	ИП8	/—/	ИП9	×	ИП4	2	×	÷
П8	Вх	1	+	÷	+	П7	—	x=0	06

с адресом перехода к оператору КИП4 в случае невыполнения проверяемого условия. Число членов ряда, суммируемых при заданном значении аргумента до выполнения условия максимальной точности, можно определить после окончания вычислений по содержимому регистра 4.

*Иногда значение интеграла вероятности принимают в 2 раза меньшим.

Окончание программы должно обеспечивать умножение на коэффициент $\sqrt{2/\pi}$ и вычисление функции $1 - \Phi(x)$, что можно реализовать фрагментом

ИП7 2 π ÷ √ × 1 ↔ — Вх С/П

Составив из образованных фрагментов программу с инструкцией $x =$ РХ В/О С/П РХ $= \Phi(x)$, РУ $= 1 - \Phi(x)$, вычислим значения искомых функций для ряда значений аргумента, регистрируя и время счета:

$\Phi(0,1) = 0{,}079655672$ ($t \approx 23$ с); $\Phi(1{,}0) = 0{,}68268947$ ($t \approx 56$ с); $\Phi(3) = 0{,}99730032$ ($t \approx 2$ мин 25 с); $\Phi(5) = 1{,}0001493$ ($t \approx 4$ мин 40 с).

По содержимому регистра 4 после каждого выполнения программы находим, что для вычисления каждого из полученных значений потребовалось суммировать соответственно 3, 9, 22 и 44 члена ряда до получения условия максимальной точности.

Сравнивая результаты вычислений с табличными данными, находим, что при $x < 3$ результаты вычислений содержат не менее 7 верных цифр, но при дальнейшем увеличении аргумента точность результатов уменьшается при увеличении числа суммируемых членов ряда и соответствующих затрат времени. Кроме того, при $x > 0$ вычисление функции $1 - \Phi(x)$ вычитанием из единицы полученных результатов приводит к грубым ошибкам. Поэтому при $x \geqslant 0$ целесообразно вычислять функцию $1 - \Phi(x)$, а значение $\Phi(x)$ находить вычитанием из единицы получаемого значения.

Воспользуемся известным асимптотическим приближением

$$1 - \Phi(x) = \sqrt{\frac{2}{e^{x^2}\pi x^2}}\left(1 - \frac{1}{x^2} + \frac{3}{(x^2)^2} - \frac{3 \cdot 5}{(x^2)^3} + \dots\right).$$

Так как методическая погрешность этого приближения максимальна при наибольшем значении аргумента $x = 3$, то с помощью ПМК и сравнения с табличными данными находим, что требуемая точность достигается при 7 членах асимптотического ряда. Корректируя коэффициент 5-го члена, получаем расчетную формулу

$$1 - \Phi(x) = \left(\left(\left(\left(\frac{3{,}7}{x^2} - 1\right)\frac{15}{x^2} + 3\right)\frac{1}{x^2} - 1\right)\frac{1}{x^2} + 1\right)\sqrt{\frac{2}{e^{x^2}\pi x^2}},$$

обеспечивающую предельную погрешность вычисления $1 - \Phi(x)$ около $5 \cdot 10^{-7}$ и вычисления $\Phi(x)$ не менее 7 верных цифр [20].

Для уменьшения времени счета реализуем эту формулу без организации цикла с хранением значений $1/x^2$ в операционном стеке, что обеспечивается фрагментом программы

ИП9	1/x	↑	↑	↑	3	,	7	×	1
—	×	1	5	×	3	+	×	1	—
×	1	+	↔	↑	1/x	e^x	÷	√	×
2	π	÷	√	×	1	↔	—	Вх	С/П

Сравнение результатов вычисления $\Phi(x)$ по этому фрагменту с табличными значениями интеграла вероятности подтверждает, что $\Phi(x)$ содержит не менее 7 верных цифр.

Таким образом, требуемая точность обеспечена при разбиении интервала аргумента на участки с $x < 3$ и $x \geqslant 3$ с вычислением $\Phi(x)$ различными методами. При совмещении всех составленных фрагментов в одной программе в ее начальной части необходимо организовать переход к нужному фрагменту программы в зависимости от значений аргумента, а в обоих фрагментах выделить одинаковые последовательности операторов наибольшей длины и поместить одну из них в подпрограмму для уменьшения общей длины программы. В этом случае полу-

чим следующую программу, обеспечивающую вычисление $\Phi(x)$ с 7 верными знаками, с оптимальными затратами времени:

П7	П8	x^2	П9	9	—	x<0	34	Cx	П4
КИП4	ИП7	ИП8	/—/	ИП9	×	ИП4	2	×	÷
П8	Bx	1	+	÷	+	П7	—	x=0	10
ИП7	ПП	68	С/П	ИП9	1/x	↑	↑	↑	3
,	7	×	1	—	×	1	5	×	3
+	×	1	—	×	1	+	↔	↑	1/x
e^x	÷	√	×	ПП	68	↔	С/П	2	π
÷	√	×	1	↔	—	Bx	B/O		

Инструкцию к выполнению этой программы можно составить в такой форме: x = РХ В/О С/П РХ = $\Phi(x)$, РУ = 1 — $\Phi(x)$; $t \leqslant 140$ с для $x < 3$ и $t \approx 18$ с для $x \geqslant 3$. Для библиотечного варианта программы следует также составить контрольный пример, которым может быть: $\Phi(0,1) = 0{,}079655672$; $1 - \Phi(0,1) = 0{,}9203443$ ($t \approx 27$ с); $\Phi(5) = 0{,}9999994$; $1 - \Phi(5) = 5{,}7326836 \cdot 10^{-7}$ ($t \approx 18$ с).

Точность вычисления функции $1 - \Phi(x)$ при $x \geqslant 3$ по составленной программе определяется абсолютной погрешностью $5 \cdot 10^{-7}$, что при увеличении x и уменьшении $1 - \Phi(x)$ приводит к уменьшению относительной погрешности и числу верных цифр. Чтобы обеспечить требуемую относительную погрешность вычисления $1 - \Phi(x)$ и устранить затраты времени на вычисления с избыточной точностью, целесообразно вычислять эту функцию суммированием членов ряда

$$1 - \Phi(x) = 2/e^{x^2}\pi x^2, \text{ где } \varphi_0 = 1,\ \varphi_k = -\varphi_{k-1}(2k-1)x^2.$$

В этом случае прекращение суммирования членов ряда обеспечивается выполнением условия

$$|(S_i - S_{i-1})/S_i| \leqslant \delta,$$

где S_i — текущая сумма членов ряда; δ — заданная предельная относительная погрешность.

Аналогичные условия прекращения вычислений целесообразно ввести и для функции $\Phi(x)$ при $x < 3$, чтобы устранить избыточные затраты времени при заданной относительной погрешности, большей предельно достижимой. В этом случае длина программы увеличится, но ее можно уменьшить (за счет увеличения времени счета) при организации обращения к подпрограммам:

П7	П8	x^2	П9	Cx	П4	Bx	3	—	x<0
34	КИП4	ИП7	ИП8	/—/	ИП9	×	ИП4	2	×
÷	П8	Bx	1	+	—	ПП	75	x<0	11
ИП7	ПП	65	С/П	1	П7	П8	КИП4	ИП7	ИП8
/—/	ИП9	—	ИП4	2	×	1	—	×	П8
ПП	75	x<0	37	ИП7	ИП9	↑	e^x	×	√
÷	ПП	65	↔	С/П	2	π	÷	√	×
↔	—	Bx	B/O	+	П7	—	ИП7	÷	x^2
√	ИП5	—	B/O						

Инструкцию к этой программе необходимо дополнить указанием на ввод заданной относительной погрешности вычислений δ = Р5, x = РХ В/О С/П РХ = $\Phi(x)$, РУ = 1 — $\Phi(x)$. При $\delta = 5 \cdot 10^{-6}$ по этой программе получим $\Phi(0,1) = 0{,}079655672$; $1 - \Phi(0,1) = 0{,}9203443$ ($t \approx 26$ с); $\Phi(2{,}9999999) = 0{,}99729553$; $1 - \Phi(2{,}9999999) = 0{,}0027045$ ($t \approx 120$ с); $\Phi(5) = 0{,}9999994$;

$1 - \Phi(5) = 5{,}7331149 \cdot 10^{-7}$ ($t \approx 65$ с). Время выполнения подобной программы при заданной точности вычнсления $\Phi(x)$ и $1 - \Phi(x)$ можно уменьшить, аппроксимируя эти функции, например, степенными многочленамн с многозначными коэффициентами. Для этого можно использовать метод экономизации усеченных рядов, рассмотренных выше, с помощью многочленов Чебышева [17], готовые расчетные формулы, подобные приведенным в [6], определить коэффициенты аппроксимирующих многочленов методом неопределенных коэффициентов или наименьших квадратов [20] или другими способами. Однако ввод многозначных коэффициентов в память данных связан со значительными затратами времени и подобные аппроксимирующие выражения целесообразно использовать при наличии внешнего накопителя с постоянным хранением коэффициентов аппроксимирующих выражений и вызовом их в память данных по мере необходимости.

Рассмотренный пример подтверждает целесообразность использования ПМК для экспериментальной оптимизации программы в процессе ее отладки.

Глава 6
ПУТИ СОВЕРШЕНСТВОВАНИЯ ПРОГРАММИРУЕМЫХ МИКРОКАЛЬКУЛЯТОРОВ

6.1. СОВРЕМЕННОЕ СОСТОЯНИЕ

Начальный период разработки и производства ПМК в значительной степени определили пути развития микроэлектроники, так как ПМК были первыми массовыми изделиями, в которых широко использовались полупроводниковые элементы с большой степенью интеграции (БИС).

В отличие от управляющих и вычислительных устройств на базе типовых микропроцессорных наборов, ПМК собираются на заказных БИС, отвечающих требованиям максимальной производительности при минимальной стоимости производства и эксплуатации ПМК. Это привело к интенсивным поискам архитектуры, структуры и конструкции ПМК, в наибольшей степени отвечающих этим требованиям. Быстрое совершенствование ПМК зарубежными фирмами предопределялось и конкурентной борьбой за быстро расширяющиеся рынки сбыта.

Требование непрерывного совершенствования ПМК привело их ведущих производителей прежде всего к необходимости перехода на технологию изготовления элементной базы, в наилучшей степени отвечающую критерию производительность/стоимость. Повышение быстродействия достигается при переходе от p-канальной МОП-технологии к более дорогостоящей n-технологии, но потребление энергии и в этом случае остается достаточно высоким. Поэтому в настоящее время преимущественное положение завоевала КМОП-технология, основанная на дополнительных (комплементарных) свойствах p- и n-областей полупроводниковых кристаллов. Эта технология при относительно небольшом повышении стоимости изготовления микросхем обеспе-

чивает достаточно высокое быстродействие (единицы и даже десятки мегагерц тактовой частоты) и малое потребление энергии, что особенно существенно для носимых вычислительных средств, а также возможность работы в ПМК комплекта автономных химических источников в течение нескольких сотен (обычно 200—300) часов.

Непрерывное повышение степени интеграции и надежности элементной базы ПМК привело к разработке запоминающих устройств (ЗУ) большой емкости (например, в ПМК типа TI-74 емкость встроенного ПЗУ 1 Мбит) и однокристальных элементов, содержащих все структурные компоненты ЭВМ — арифметическо-логическое устройство (АЛУ), постоянные и оперативные ЗУ, интерфейс ввода-вывода информации. Расширение возможностей функциональных элементов ПМК стало базой и для поиска архитектурных решений, характерных не только для ПЭВМ или мини-ЭВМ, но и для стационарных ЭВМ высокой производительности.

Развитие элементной базы ПМК способствовало ее использованию и в других областях вычислительной техники, в частности при разработке микроконтроллеров для управления различными процессами, объем производства которых в настоящее время превышает выпуск ПМК. Такие микроконтроллеры отличаются низкой стоимостью производства и малыми габаритными размерами, характерными для ПМК, успешно конкурируют с микроконтроллерами, изготовляемыми на базе микропроцессорных наборов.

Характерная особенность развития ПМК — последовательная разработка и производство ряда моделей ПМК, совместимых снизу вверх по архитектуре, в частности входному языку, что позволяет использовать прикладное программное обеспечение младших моделей ряда при программировании старших моделей. Остальные модели, образующие подобные ряды, часто специализируются в соответствии со спросом потребителей. В частности, специализация ПМК для решения экономических (коммерческих или деловых) и научно-технических или инженерных задач обеспечивается соответствующим словарным запасом входного языка и встроенными функциями.

Современное состояние разработки и производства ПМК характеризуется большим разнообразием технологических, архитектурных и конструктивных решений, определяющих широкий диапазон стоимости и производительности. Однако производимые в настоящее время во всем мире ПМК можно условно разбить на четыре основные группы:

1) ПМК с традиционными компактными входными языками высокого уровня;

2) массовые ПМК с алгоритмическими входными языками, являющимися, как правило, версией Бейсика;

3) ПМК высокой производительности с большим ассортиментом внешних устройств, развитым прикладным обеспечением, а иногда несколькими входными языками, которые по вычислительным возможностям не уступают большинству массовых настольных ПЭВМ, но

отличаются от них более высокой стоимостью, что связано в основном с затратами на миниатюризацию внешних устройств;

4) ПМК, программа работы которых задается расчетными формулами, вводимыми нажатием клавиш, и операторами высокого уровня, обеспечивающими автоматическое решение достаточно сложных типовых математических задач.

Современные ПМК с компактными входными языками разнообразны как по используемой элементной базе, так и по вычислительным возможностям, что обеспечивает удовлетворение потребностей различных групп пользователей. Некоторые фирмы до настоящего времени выпускают ПМК на элементах, изготовленных по МОП-технологии, дешевые и пользующиеся спросом у пользователей. Однако большинство выпускаемых ПМК собраны на элементах, изготовленных по КМОП-технологии, причем характерно использование матричных жидкокристаллических индикаторов (ЖКИ), отличающихся малым потреблением энергии и обеспечивающих отображение не только символов, но и графиков или других изображений.

Для примера технические данные наиболее распространенных современных ПМК фирмы HP с компактными входными языками приведены в табл. 6.1. Характерной особенностью ПМК, разрабатываемых ведущими изготовителями, является широкое использование однотипных конструктивных модулей и их элементов для ПМК различного назначения, что снижает стоимость производства. В частности, первые четыре ПМК в табл. 6.1 имеют различное назначение, но изготовлены в идентичных корпусах с одинаковыми ЖКИ и клавиатурами, отличающимися лишь обозначенными на них элементами алфавитов входных языков.

Недорогой массовый микрокомпьютер HP-11C (рис. 6.1) предназначен для решения широкого круга научных и инженерно-технических задач. Словарный запас его входного языка включает арифметические операторы, операторы вычисления простейших показательных функций, степенной функции общего вида, прямых и обратных тригонометрических и гиперболических функций, простых и сложных процентов, коэффициента корреляции, факториала, числа перестановок и размещений, а также вызова числа π и квазислучайных чисел с равномерным распределением в интервале (0,1). Предусмотрена возможность представления чисел в «научной» показательной форме с мантиссой, содержащей одну значащую цифру в целой части, или в «инженерной» форме с отделенными тройками десятичных разрядов, соответствующими единицам измерений физических величин, отличающихся в 1000 раз. Любопытной особенностью, расширяющей рынки сбыта, является возможность представления чисел с точкой, отделяющей дробную часть, и запятыми между тройками разрядов, принятого в США, или с запятой перед дробной частью и точками, отделяющими тройки разрядов, принятого в Европе.

Значительно большую, чем в HP-11C, производительность и емкость оперативной памяти имеет ПМК HP-15C. Его входной язык со-

Таблица 61

Технические характеристики современных ПМК фирмы HP

Тип	HP-11C	HP-15C	HP-16C	HP-12C	HP-41CV	HP-41CX	HP-18C	HP-28C
Индикатор (ЖКИ)	Однострочный сегментный, 12 знаков						Четырехстрочный матричный, 23 знака	
Число регистров памяти данных	21	67	101	20	319	319	4	—
Емкость ОЗУ пользователя	203	448	203	99	2233/6433 байт	3100/6433 байт	1200 байт	1709 байт
Сменные модули памяти	—	—	—	—	+	+	—	—
Буквенный текст	—	—	—	—	+	+	+	+
Число флагов в программе	2	10	6	—	56	56	—	64
Число проверок условий	8	12	8	2	14	20	6	22
Число вложений подпрограмм	4	7	4	—	6	6	—	—
Число меток переходов	10	25	16	—	—	—	—	—
Габаритные размеры, мм	127×79×16				142×79×33		190×165×13	
Масса, г	150				205		230	

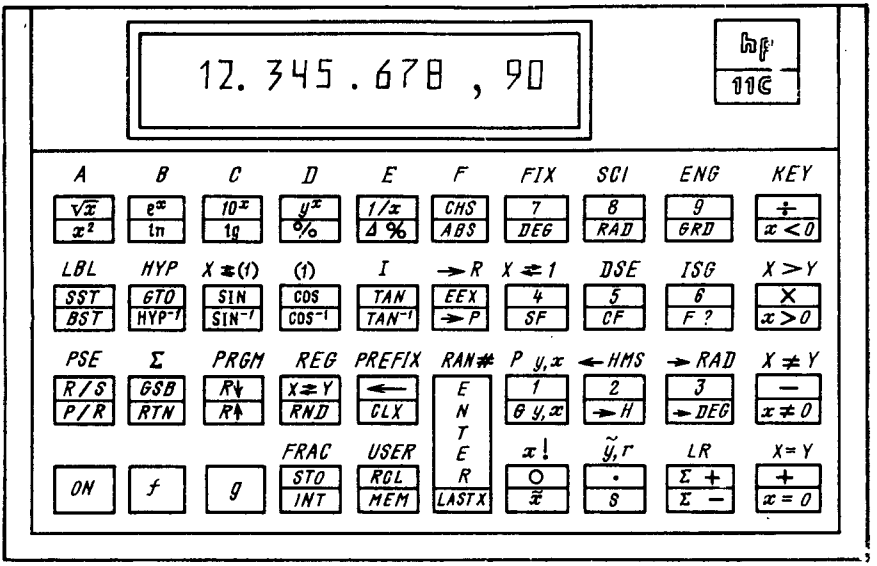


Рис. 6.1

держит также операторы высокого уровня для автоматического выполнения численного интегрирования, решения уравнений, операций над комплексными числами, векторами и матрицами.

Специализированный HP-16C, предназначенный для программистов и специалистов по вычислительной технике, обеспечивает выполнение логических и других операций над двоичными словами, содержащими до 64 разрядов, преобразование систем счисления, выполнение арифметических операций над десятичными числами, вычисление функций $\sqrt{x}$, e^x и y^x. Область памяти данных содержит 101 регистр для хранения 16-разрядных двоичных слов.

Специализированный HP-12C предназначен для решения экономических (деловых) задач. Словарный запас его входного языка содержит небольшое число математических операторов (+, —, *, /, $\sqrt{x}$, 1/x, lnx, e^x, y^x, x!), но имеет ряд операторов высокого уровня для автоматического решения типовых экономических задач, включая и статистические. Эти операторы высокого уровня могут использоваться как в программах, так и при вычислениях нажатием клавиш.

ПМК HP-41CV и HP-41CX являются современными модификациями популярного ПМК высокой производительности (подобные ПМК фирмы-производители называют карманными, ручными или персональными компьютерами) HP-41C, объем продажи которого вместе с модификациями составил за последнее десятилетие около 1 млн. экземпляров. Все они изготовлены в однотипном корпусе (рис. 6.2) с прорезью

в торцах для магнитных карточек с разъемами для подключения внешних устройств. ОЗУ пользователя этих ПМК имеет переменную границу между областями данных и программ, причем программные слова различной длины. Емкость памяти пользователя HP-41C 448 байт, или 63 регистра данных, и может расширяться до 2233 байт (что соответствует от 1500 до 2000 программных слов) с помощью внешних модулей расширения ОЗУ. Емкость оперативной памяти HP-41CV и HP-41CX значительно больше и может быть расширена до 6,4 Кбайт. Емкость ПЗУ для хранения операционной системы 24 Кбайт и может быть расширена при использовании внешних модулей ПЗУ с пакетами прикладных программ и дополнительным математическим обеспечением. Следует отметить, что возможность приобретения подготовленных профессиональными программистами пакетов прикладных программ для решения задач из различных областей знаний в значительной мере обусловила распространение рассматриваемых ПМК несмотря на их относительно высокую стоимость. Особенность HP-41C — возможность ввода в программы и вывода на индикатор или печать текстовых переменных, набираемых буквами латинского алфавита (режим «альфа»). HP-41CX может использоваться также как электронные часы с подачей звукового сигнала в заданное время, секундомер или электронный календарь.

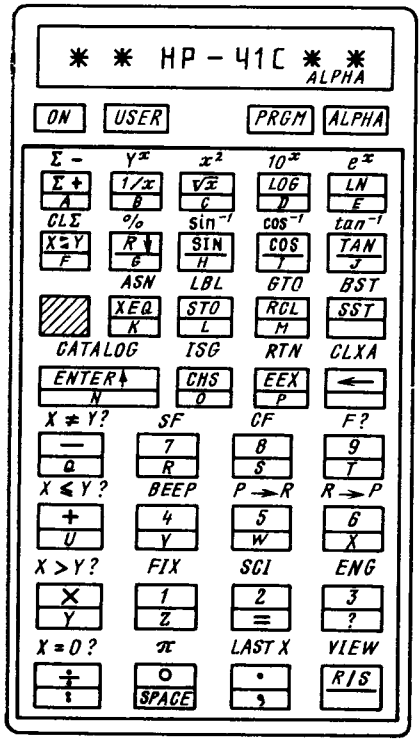


Рис. 6.2

Рассмотренные ПМК через внешний блок интерфейса могут соединяться с различными внешними устройствами, разработанными фирмой узко- и широкоформатными печатающими устройствами, видеомониторами, графопостроителями, акустическими модемами, накопителями информации на магнитных и полупроводниковых носителях.

Особенность ПМК HP-18C и HP-28C (табл. 6.1) — большой объем операционной системы, обеспечивающий высокий уровень операторов входного языка для решения типовых математических вычислительных задач. Так как расчетные выражения могут непосредственно вводиться (отображаясь на индикаторе) в обычной алгебраической форме, то эти ПМК в основном используются в режиме диалога с вводом операторов и

формул нажатием клавиш, как в непрограммируемых микрокалькуляторах.

Элементная база современных ПМК, изготовленная по КМОП-технологии, обеспечивает малое потребление энергии, что позволяет использовать в качестве автономных источников питания не только химические элементы, но и фотоэлектрические (солнечные) батареи, применяемые в массовых ПМК. В качестве примера в табл. 6.2 приведены данные ряда массовых ПМК японской фирмы Sharp.

Таблица 6.2

Технические данные массовых ПМК фирмы Sharp

Тип	Разрядность	Число функций	Число вложений скобок	Особенности
EL-506P	10(8/2)	56	15	Защита памяти
EL-509A	8(5/2)	32	3	Режим решения статистических задач
EL-512	10(8/2)	61	8	128 шагов программы
EL-514	10(8/2)	48	4	Ресурс питания 5000 ч
EL-525	10(8/2)	56	4	Гибкий корпус
EL-531	8(5/2)	34	4	Ресурс питания 10 000 ч
EL-540	8(6/2)	38	4	Солнечная батарея
EL-545	10(8/2)	56	15	» »
EL-550	8(8/2)	46	4	Встроенная термопечать
EL-5100S	24(10/2)	61	—	Расширенная память данных (двойная точность)

В табл. 6.2 указаны разрядность представления чисел в естественной форме и в скобках число разрядов мантиссы/порядка в показательной форме. Защита памяти в EL-506P, широко применяемая в более поздних ПМК этой же фирмы, — вызов содержимого энергонезависимой памяти только при вводе кода, известного пользователю.

Производительность современных ПМК повышена в связи с использованием матричных ЖКИ. Примером может служить FX-7000G (рис. 6.3) японской фирмы Casio (габаритные размеры 84 × 167 × × 14 мм, масса 156 г, потребление энергии 0,07 Вт от трех литиевых элементов). Матричный ЖКИ с 65×95 элементами обеспечивает отображение до 8 строк по 16 символов и графики. ОЗУ пользователя состоит из 26 регистров данных, обозначаемых буквами, и 422 8-разрядных ячеек программной памяти, причем память данных может быть расширена за счет программной памяти. Компактный входной язык с алгебраической записью характеризуется 82 встроенными функциями, 9 вложениями подпрограмм, что обеспечивает решение достаточно сложных задач.

Однако несмотря на большие возможности обработки информации современными ПМК с компактными входными языками, перспективы

их дальнейшего развития ограничены. В связи с умолчанием имен текущих переменных при чтении и составлении программ пользователю приходится представлять мысленно или на бумаге содержимое регистров операционного стека, что затрудняет пользование ПМК. Поэтому дальнейшие перспективы совершенствования ПМК в значительной мере связаны с использованием алгоритмических входных языков, усложняющих конструкцию ПМК, но обеспечивающих большие удобства пользователю.

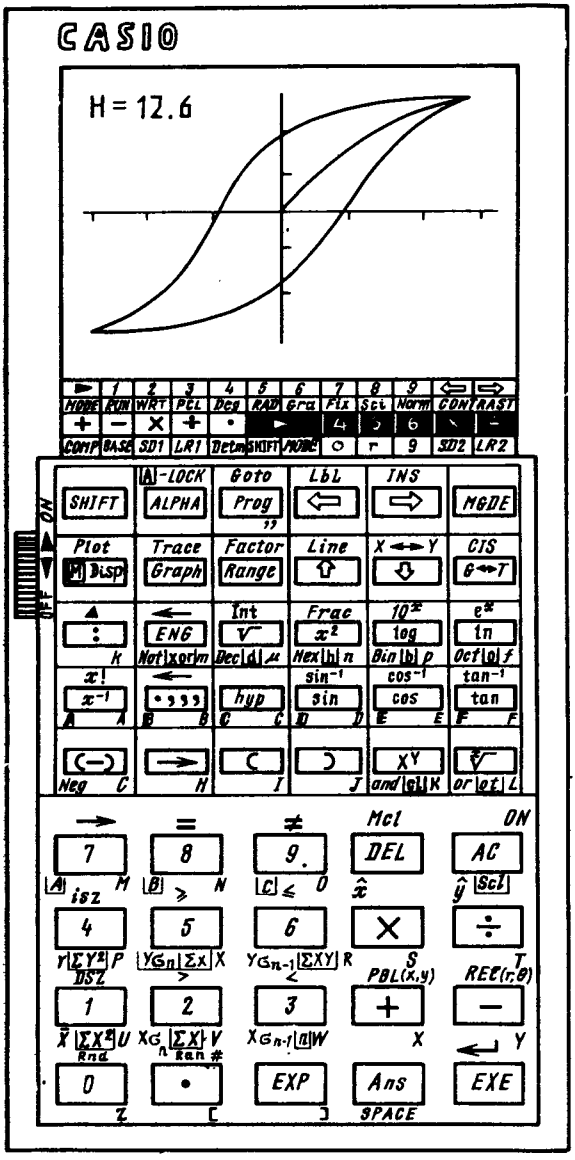


Рис. 6.3

В настоящее время наиболее широкое применение находят ПМК с различными версиями Бейсика, причем они часто настолько различаются, что по существу являются различными формальными языками, объединенными лишь несколькими общими операторами и возможностью использования диалогового режима с выполнением строк программы в порядке увеличения меток. Переход к алгоритмическим языкам характерен для ведущих производителей как массовых, так и высокопроизводительных ПМК.

Среди различных специализированных ПМК (к которым относятся и такие экзотические приложения микроэлектроники, как электронные переводчики или секретари и игровые микрокалькуляторы) следует выделить микрокомпьютеры, предназначенные для обучения школьни-

ков и студентов составлению алгоритмов и программ решения прикладных задач. К подобным ПМК относится TI-57LCD с 10-символьным ЖКИ, обеспечивающий ввод, редактирование и отладку небольших (до 48 шагов) программ при алгебраической записи выражений, содержащих до 15 вложений скобок при одном вложении подпрограмм. Входной язык этого ПМК кроме стандартного набора функций содержит также операторы взаимного преобразования десятичных и шестнадцатеричных чисел, а также полярных и прямоугольных координат.

Значительно большей производительностью отличается TI-66 с «дружественной» операционной системой, подсказывающей пользователю требуемую последовательность действий и указывающей на ошибки в составляемых программах. TI-66 снабжен 10-символьным ЖКИ, ОЗУ пользователя с программной памятью емкостью 512 байт и памятью данных на 64 адресуемых регистрах. Этот ПМК с компактным входным языком со скобочной (до 9 вложений скобок, 6 вложений подпрограмм) алгебраической записью выражений, может выполнять программы, составленные на входных языках Р-58 и Р-59 (см. табл. 1.3).

К обучающим относится также TI-LCD—Programmer, предназначенный для студентов вычислительных специальностей и обеспечивающий программирование операций над представлениями чисел в различных системах счисления, а также TI-44 для студентов экономических и административных специальностей.

Разработка ПМК для обучения основам информатики имеет особое значение для нашей страны с огромным контингентом учащихся и необходимостью применения надежных и недорогих школьных ЭВМ. Однако в этом случае возникает проблема выбора естественного языка для ключевых слов языков программирования высокого уровня.

Любая ЭВМ может быть дополнена относительно несложным устройством для замены английских слов русскими, но в общем случае использование английских слов оправдано возможностью международного обмена вычислительной техникой и ее программным обеспечением. Однако для школьных ЭВМ такой выбор не однозначен.

Основная задача школьного курса информатики — не подготовка программистов, а освоение учащимися формализации описания способов решения разнообразных задач. Так как человек мыслит на родном языке, то стандартизованные словесно-формульные описания алгоритмов, как и отображающие их программы на алгоритмических входных языках высокого уровня, ему проще составлять и понимать на родном, а не иностранном языке, даже если он знаком с последним. Заметим, что именно по этой причине Алгол в школьном курсе информатики был заменен его версией алгоритмического языка АЯ с русскими словами. По этой же причине получают распространение и входные языки ПЭВМ с русскими словами, подобные языку Рапира. Поэтому в экспериментальном школьном микрокомпьютере «Электроника МК-72» также использован входной язык высокого уровня с русскими словами.

Последовательная передача и обработка информации в ПМК первых поколений позволила минимизировать аппаратурные затраты и стоимость производства. По мере совершенствования и снижения стоимости элементной базы появилась возможность повышения быстродействия ПМК при использовании многопроводных информационных шин с параллельной передачей информации. В современных ПМК эта возможность реализована, как правило, параллельной передачей слов информации по тетрадам в течение такта, соответствующей передаче на каждом такте десятичной или шестнадцатеричной цифры. При этом оказалось возможным создание на существующей элементной базе ПМК первых поколений многофункциональных ручных компьютеров со свойствами настольных ПЭВМ.

Примером подобной микроЭВМ с последовательно-параллельной передачей информации является экспериментальный ПМК «Электроника МК-72», созданный в основном на элементной базе ПМК последовательного действия для проверки ряда архитектурных решений, которые могут оказаться полезными для ПМК следующих поколений. Непосредственной целью создания этого ПМК явилась разработка опытной многофункциональной школьной микроЭВМ с минимальной стоимостью при достаточно высокой производительности, обеспечивающей применение как отдельно, так и в локальной сети учебного класса. По архитектуре и математическому обеспечению «Электроника МК-72» может быть отнесена к бытовым ПЭВМ, а по габаритным размерам, массе (не более 400 г) и быстродействию — к ПМК при потреблении энергии от внешних источников мощностью не более 0,7 Вт.

Обобщенная структурная схема рассматриваемого ПМК (рис. 6.4) отображает соединение центрального процессора (ЦП) с контроллером интерфейса (ввода-вывода) КИ, клавиатурой и встроенными устройствами отображения информации (зуммером, светодиодами и цифровым индикатором), с блоками ОЗУ и ПЗУ, а также системного расширителя (СР), обеспечивающего включение ПМК в локальную сеть. К внешним устройствам относятся отладочный пульт с клавиатурой и индикатором, внешнее электрически стираемое репрограммируемое ЗУ (РПЗУ) и блок интерфейса (БИТ) для связи с бытовым телевизором (ТВ), экран которого используется для отображения информации в виде текстов программ и символьных или графических результатов их выполнения.

В рассматриваемом ПМК информация отображена последовательностью 16-разрядных слов, разряды которых нумеруются шестнадцатеричными цифрами от 0 до F, передаваемых за 4 такта, образующих микроцикл работы. Каждое слово определяется его адресом в памяти, который удобно записывать в шестнадцатеричной системе счислеиия. В дополнение к основной адресной шине с 16 выводами введены адреса тетрад внутри машинных слов, образованных комбинациями сигналов 00, 01 10 или 11, вырабатываемыми в каждом микроцикле.

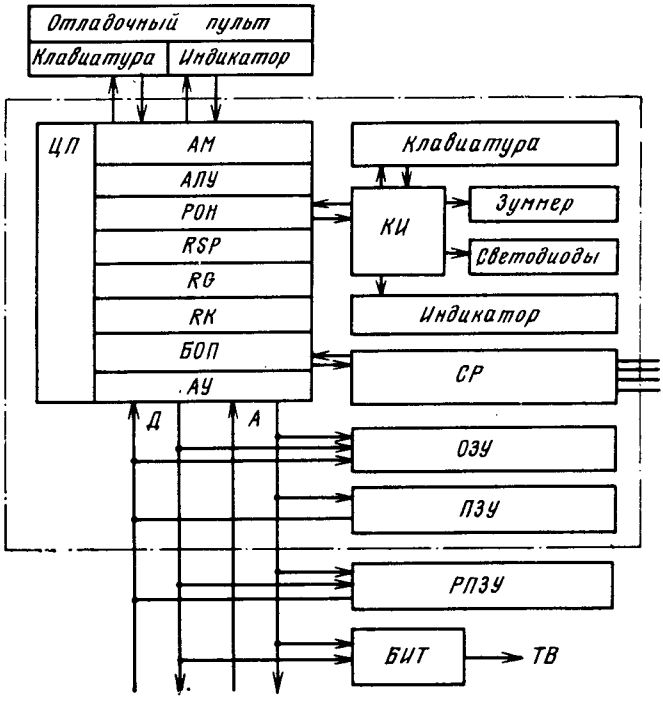


Рис. 6.4

Такой вид передачи данных позволяет использовать одну БИС памяти с 4-битовой организацией (ПЗУ и РПЗУ) или четыре БИС памяти с однобитовой организацией (ОЗУ).

Центральный процессор содержит аппаратный монитор (АМ) для настройки ПМК с отладочного пульта, 16-разрядное арифметическо-логическое устройство (АЛУ), 8 16-разрядных регистров общего назначения (РОН) Р0...Р7, блок обработки прерываний (БОП), адресное устройство (АУ) и два 4-разрядных регистра RG и состояния процессора RSP и «короткий» регистра RK. Регистр состояний процессора содержит одноразрядные поля условий (флагов), значения 0 или 1 которых зависят от результатов последней выполненной команды и отображают соответствие нулю результата операции, переполнение или перенос результата сложения. Регистр Р0 выполняет функции счетчика команд, содержащего адрес команды, вызванной из памяти, но еще не исполненной, регистр Р7 — функции указателя стека выполнения команд. ЦП через адресное устройство устанавливает адрес на выводах 16-разрядной адресной шины и пересылает обработанную информацию через 4-разрядную шину данных последовательно-параллельным кодом 4×4.

Элементную базу ПМК образуют БИС серии К745ИК18, К745ПЗУ64К для ПЗУ и К537РУ3А для ОЗУ. Адресный процессор

ЦП выполнен на БИС К745ИК1809 и интерпретатор команд на БИС К745ИК1810, контроллер интерфейса — на БИС К745ИК1814. БИС К745ИК1810 содержит входную шину данных, 4-разрядную шину для ввода информации с клавиатуры отладочного пульта, подключаемого при настройке ПМК, 7-разрядную шину кода сегментов цифрового индикатора, 4-разрядную шину управления клавиатурой и индикатором отладочного пульта, 2-разрядную шину адресов тетрад в слове, логические входы и выходы для регистра связи, выводы стробирующих сигналов записи данных во внешние устройства и выводы управления индикатором отладочного пульта.

Контроллер интерфейса содержит внутренний 16-разрядный регистр связи с встроенными устройствами отображения информации — клавиатурой, светодиодами, зуммером и цифровым индикатором с 8 знакоместами для отображения цифр и символов. Управление этими устройствами производится по команде ЦП, вызывающей обмен содержимым регистра связи контроллера и регистров процессора. При таком обмене в ЦП поступают коды нажимаемых клавиш, а в контроллер — команды управления.

Сетевой расширитель на микроЭВМ К745ИК1817 обеспечивает связь с малой локальной сетью магистрального типа, к которой могут подключаться до 16 ПМК «Электроника МК-72» с равноправным обменом информацией. Кроме выводов для связи с ЦП расширитель содержит порт асинхронной связи, а также 8-битовый параллельный порт для подключения дополнительной памяти. Эта память может использоваться как в качестве буферной, так и для хранения различных программ, включая управляющие работой системного расширителя. Набор команд системного расширителя обеспечивает указание источника и приемника, а также формат передаваемой информации. Скорость передачи данных по параллельно-последовательным каналам может достигать 1200 бод. Связь с ЦП реализуется по последовательному каналу, включающему шину данных и линию готовности приема. Системный расширитель содержит также входную и выходную параллельные 8-разрядные шины для буферной памяти и внешних устройств.

Конструктивно ПМК выполнен в виде двух модулей. Основной содержит ЦП, КИ, ПЗУ емкостью 23 Кбайт, ОЗУ емкостью 2 Кбайт и разъем общей шины. Дополнительный модуль интерфейса общей шины содержит ОЗУ емкостью 2 Кбайт, устройство интерфейса связи с телевизором. Конструкция дополнительного модуля предусматривает возможность размещения интерфейсиых блоков других внешних устройств — кассетного магнитофона, строчного термопечатающего устройства, преобразователя кодов. Примерно 0,5 Кбайт ОЗУ дополнительного модуля используется для обслуживаиия телевизионного экрана, остальная часть памяти используется как дополнительиая.

Рассматриваемый ПМК предназначен для решеиия разнообразных задач преобразования информации, хранящейся в памяти или поступающей от локальной сети и внешних устройств. Пользователь может

вводить программы решения задач на трех входных языках: языке непрограммируемого инженерного микрокалькулятора, языке ассемблера Класс (близком к языку ассемблера мини-ЭВМ PDP-11) и языке высокого уровня, близком к языку Алмир, являющемуся исходной версией языка Аналитик.

Для поддержки языков программирования в ПМК реализован язык монитора, обеспечивающего управление ходом вычислений, отладку программ, их редактирование, обмен информацией с РПЗУ, а также обучение пользователей правилам входных языков. Мониторы языков программирования выполнены таким образом, что их программные коды могут быть прочитаны пользователем на встроенном индикаторе и хранящиеся в ПЗУ мониторные программы могут служить образцами составления программ. Кроме мониторов операционной системы, в ПЗУ хранятся программы специализированных вычислителей, во внешнем ПЗУ — стандартные пакеты прикладных программ, а программы пользователя могут храниться в РПЗУ. При этом предусмотрена возможность записи в оперативную память программ, принимаемых по локальной сети.

Адресное пространство ПМК составляет 64 Кслов при 16-разрядных словах (128 Кбайт) и емкости ОЗУ в основном комплекте 1024 слов и резидентного ПЗУ 32 Кслов при максимальной емкости ПЗУ 40 Кслов с дальнейшим расширением блоками по 4 Кслов. Распределение адресов между различными ЗУ указано в табл. 6.3. Адресное пространство состоит из томов по 1000_{16} слов (например, с адресами 0000...0FFF или 1000...1FFF), каждый том разбит на страницы по 100_{16} слов в каждой (например, с адресами 1200...12FF). Следует добавить, что содержимое счетчика команд увеличивается на единицу по

Таблица 6.3

Распределение адресов в памятн

Адрес		Тип памяти	Назиачение
начальный	конечный		
0000	03FF	ОЗУ	Оперативиая иформация
0400	0FFF	—	Не используется
1000	17FF	ПЗУ 1	Моииторы ассемблера и калькулятора
1800	22FF	ПЗУ 1	Программы вычислеиия элементарных функций
2300	2FFF	ПЗУ 2	Транслятор Курс
3000	37FF	ПЗУ 3	Моиитор Курс
3800	3FFF	ПЗУ 3	Резерв
4000	4FFF	ПЗУ 4	»
5000	5FFF	—	Не используется
6000	61FF	Доп. ОЗУ	Буферы виешних устройств
6200	67FF	Доп. ОЗУ	Область текстов программ иа языке Курс

модулю $100_{16} = 256_{10}$ после того, как команда выбрана по сформированному адресу, но еще не выполнена. Так как адресация выполняется внутри страницы, то после формирования адреса 255_{10} следующие адреса начинаются с начального 0000. Если содержимое счетчика не изменилось в процессе выполнения команды (при командах переходов), то следующая команда автоматически выбирается из следующего слова. Переход на другие страницы памяти выполняется по специальным командам JUP и JSR, причем последовательность команд выполняется до считывания кода команды HALT, прерывающей работу процессора.

Программа на языке машинных команд образована последовательностями 16-разрядных двоичных слов (отображаемых 4-разрядными шестнадцатеричными кодами), каждое из которых представляет одно- или двухадресную, условную или специальную команду, операнд-адрес или операнд-число. В двухадресной команде первый операнд называют источником (обозначают src), а второй — приемником (dst).

Примерами команд, представленных на языке ассемблера, могут служить двухадресная команда MOV R1, R2, управляющая пересылкой содержимого регистра R1 в R2, одноадресная команда INC R3, добавляющая единицу к содержимому регистра R, BEQ 25 — команда перехода по условию равенства на внутристраничный адрес 25, специальная команда ожидания WAIT с кодом 0020. Специальные команды, безадресные, их коды не содержат переменных полей и предназначены для обслуживания перываний различного типа.

В системе команд имеется одноадресная команда SW <Операнд>, по которой операнд обменивается с содержимым порта, соединенного с контроллером интерфейса, причем последний интерпретирует содержимое порта как команду. Имеется также подмножество команд контроллера интерфейса (которые могут использоваться как на уровне пользователя, так и операционной системы) для управления встроенными устройствами отображения информации: F000 (запрос кода клавиши), A0XX (код XX клавиши при ответе на запрос), E3XX (индикация с начала строки), E7XX (индикация следующего символа), E200 (включение зуммера), E600 (выключение зуммера), EA0X (включение комбинации X светодиодов).

Различные основные и специальные режимы адресации, используемые в рассматриваемом ПМК, описаны в табл. 6.4. В ПМК предусмотрена возможность ввода и вывода программ на языке машинных кодов как в цифровой форме, так и на языке ассемблера Класс. Примером может служить программа вывода набора символов на экран:

```
100  MOV α, P1        3401
101  6200             6200
102  MOV α, P2        3202
103  FF               FF
104  MOV R2, —(R1)    3221
105  BNE 04           0604
106  EMT              50
```

Таблица 6.4

Режимы адресации

Тип	Обозначение	Описание
Регистровый	R_n	Операнд находится в регистре R_n
Автоинкрементный	$(R_n)+$	Содержимое R_n используется как адрес, после чего $R_n:=R_n+1$
Автодекрементный	$-(R_n)$	$R_n:=R_n-1$, после чего содержимое R_n используется как адрес
Индексный	$A\ (R_n)$	Адрес операнда определяется как сумма содержимого R_n и константы, следующей за командным словом
Регистровый косвенный	$@R_n$	Содержимое R_n используется как адрес операнда
Косвенный автоинкрементный	$@(R_n)+$	В регистре R_n адрес адреса операнда, после выборки которого $R_n:=R_n+1$
Косвенный автодекрементный	$@-(R_n)$	Содержимое $R_n:=R_n-1$ принимается как адрес адреса операнда
Косвенно-индексный	$@A\ (R_n)$	Адрес адреса операнда равен сумме содержимого R_n и константы, следующей за командным словом
Непосредственный	$(R0)+$	Операнд следует за командным словом
Абсолютный	$@(R0)+$	Операнд находится по адресу, следующему за командным словом
Относительный	$A\ (R0)$	Адрес операнда равен 16-ричной сумме PC+A, т. е. смещен на A слов
Косвенно-относительный	$@A\ (R0)$	Адрес адреса операнда равен 16-ричной сумме PC+A

В этой программе имеется обращение к области адресов 60C0...61FF экрана. Символы выводятся в порядке возрастания их стандартных (стандарт ASCII) кодов, так как в ПМК коды символов выбраны инверсными кодами ASCII и перебор кодов, задаваемых в регистре R2, производится в порядке их убывания. Первыми на экран выводятся алфавитно-цифровые символы, последними — псевдографические. Команда EMT передает управление монитору.

Для выбора языка программирования используются накладные транспаранты клавиатуры, на которых обозначены символы, вводимые нажатием клавиш при выбранном языке. Директивы монитора Класс, управляющие вводом программ и данных, организацией счета, отладкой программ и установкой режима входного языка, обозначены на клавиатуре языка ассемблера: ВВОД <адрес> (установка адреса ввода программы или данных), ЛИСТ <адрес> (вывод на экран текста программ на языке Класс, начиная с указанного адреса), АДРЕС <адрес> (установка адреса контрольной точки останова при отладке программы), СЧЕТ <адрес> (пуск программы с указанного адреса), СЧЕТ (пуск программы с точки останова), ШАГ <адрес> (исполнение одной команды по указанному адресу), ШАГ (исполнение команды,

следующей за контрольной точкой), ОТВЕТ <адрес> (вывод на экран содержимого памяти в шестнадцатеричном коде, начиная с указанного (адреса), КОПИЯ (занесение в текущую ячейку памяти при вводе содержимого предыдущей ячейки, объявление текущей следующей ячейки), КЛАСС (пуск монитора с начала с установкой флагов в исходное состояние), КУРС (перейти в монитор Курс), РЕЖИМ (переход в режим инженерного калькулятора).

В режиме инженерного микрокалькулятора при нажатии клавиш ПМК выполняет арифметические операции и вычисляет значения более 20 встроенных функций — прямых и обратных тригонометрических, степенных, логарифмических и показательных, модуля $C(x, y)$ и аргумента $B(x, y)$ комплексного числа, заданного парой действительных чисел x и y, отображающих действительную и мнимую составляющие, а также вызова квазислучайного числа с равномерным распределением в интервале (0,1).

В соответствии с назначением ПМК для применения в средней школе алгоритмы вычисления встроенных функций обеспечивают их точные значения в диапазоне представления чисел на индикаторе при 4 десятичных разрядах мантиссы и 2 порядка, что соответствует точности школьных таблиц (например, таблицы логарифмов) функций. Выбор алгоритмов определялся ограниченной емкостью ПЗУ, разрядностью машинных представлений чисел и набором базовых команд процессора. Система команд обеспечивает операции целочисленной арифметики, так что арифметические операции над действительными числами реализуются программно.

Вычисление функций основано на аппроксимации нескольких базовых функций, по которым определяются остальные. Приведение аргумента в узкий интервал, обеспечивающий повышение точности аппроксимации базовых функций, реализовано по известным соотношениям.

Базовые функции аппроксимированы степенными многочленами

$$\sin x = x - 0{,}1667x^3 + 0{,}0083x^5 - 0{,}0002x^7, \quad |x| < 0{,}7852;$$
$$\cos x = 1 - 0{,}5001x^2 + 0{,}0418x^4 - 0{,}0015x^6;$$
$$\operatorname{tg} x = x + 0{,}3333x^3 + 0{,}1333x^5 + 0{,}0540x^7 + 0{,}0306x^6;$$
$$10^x = 1 + 0{,}2303x + 0{,}0265x^2 + 0{,}0020x^3 + 0{,}0001x^4.$$

Вычисление логарифма и корня квадратного основано на потетрадных сдвигах и сложениях по методу «цифра за цифрой». По вычисленным базовым функциям определяют остальные встроенные функции

$$\arccos x = \pi/2 - \arcsin x; \quad e^x = 10^{x/\ln 10} = 10^{0{,}4343x};$$
$$\operatorname{ch} x = (e^x + e^{-x})/2, \quad \operatorname{sh} x = (e^x - e^{-x})/2, \quad |x| > 1;$$
$$\operatorname{arth} x = \ln((1+x)/(1-x))/2;$$
$$C(x, y) = \sqrt{x^2 + y^2}, \quad B(x, y) = \operatorname{sign} x \arccos x\,(y/C(xy)).$$

В режиме инженерного микрокалькулятора и в программах на языке Курс встроенные функции вызываются по кодам их имен, а в языке ассемблера Класс — по адресам.

Входной язык программирования высокого уровня Курс близок к языку Алмир (исходная версия языка Аналитик), имеет русские ключевые слова с одним типом переменных — рациональными действительными числами. Алфавит языка Курс включает все прописные буквы русского алфавита, латинские буквы D, E, F, X, Q, I, J, K, L, M, N, десятичные цифры, специальные знаки и ключевые слова. К специальным знакам относятся знаки двухместных операций +, —, ×, /, ↑ (Возведение в степень), /—/ (изменение знака числа), знаки сравнения |=|≠|<|>|≤|≥|, разделители :=|.|,|:|;|, а также круглые (алгебраические), квадратные (индексные) и двойные треугольные (операторные) скобки. Ключевыми словами являются ограничители ТО, ИН (иначе), КП (конец программы), имена операторов ЕСЛИ, ДЛЯ, ДО, ВЫЗОВ, ВЫХОД, ВЫВОД, ТЕКСТ, ГРАФИК, ПАМЯТЬ, СТОП, НА, ЧЕРЕЗ, ИМЯ, а также имена прямых SIN, COS, TG и обратных ASIN, ACOS, ATG тригонометрических, логарифмических LN и LG, экспоненциальной EXP функций, вычисления корня квадратного SQRT, целой INT, дробной FRC частей и абсолютного значения ABS числа, а также формирования квазислучайного числа RND с равномерным распределением в интервале (0,1).

В программах на языке Курс арифметические выражения вводят в обычной скобочной алгебраической форме, причем операторы присваивания записывают без ключевых слов, например X:=(A+5)×C↑2.

Оператор условного перехода имеет вид

ЕСЛИ <Условие>ТО<Оператор>ИН<Оператор>

причем после ТО или ИН может быть записан оператор безусловного перехода НА<Метка перехода> с буквенной меткой, записываемой также перед оператором, которому передается управление.

Оператор цикла начинается с заголовка ДЛЯ<Оператор присваивания>, после которого вводят тело цикла, оканчивающееся оператором ДО<Имя переменной в заголовке : условие выхода из цикла>, например

ДЛЯ K:=LN(X)/2 ... ДОК:K≤L+3;

Обращение к подпрограмме реализуется оператором ВЫЗОВ <Имя подпрограммы>, причем подпрограмма начинается ключевым словом ИМЯ, после которого записывают выбранное имя подпрограммы, и оканчивается оператором ВЫХОД (Возврат из подпрограммы).

Комментарии вводят после имени ТЕКСТ, а для вывода переменных на экран телевизора и индикатор используют оператор с именем ВЫВОД. В этом операторе указывают текст, имена переменных или определяющие их выражения и формат вывода. Например, при выполнении оператора

ВЫВОД НОМЕР, I, ДЛИНА, SQRT (L↑2 + M↑2 + N↑2), F530

для значений переменных I = 5, L = 1, M = 2, N = 3 на экран будет выведено сообщение

НОМЕР 5 ДЛИНА 3,741

В операторе ВЫВОД могут использоваться различные форматы, например 528 (перевод строки), 529 (пауза до нажатия любой клавиши), 530 (включение зуммера до нажатия одной из клавиш), 531 (включение зуммера на 1 с), 532 (очистка экрана), 533 (маркер вверх на одну строку). Применение формата позволяет запрограммировать вывод результатов в удобной для восприятия форме. При этом алфавитно-цифровая информация выводится на экран, а вспомогательная — на встроенные устройства отображения информации (зуммер, цифровой индикатор или светодиоды).

Оператор ПАМЯТЬ D[*n*] служит для обращения к одномерному массиву данных D с указанием числа *n* элементов массива.

Программа на языке Курс начинается с текстового заголовка и записывается по строкам операторов, отделяемых точкой с запятой. Оператор СТОП прекращает выполнение программы, но в конце программы записывается ограничитель КП, после которого могут записываться подпрограммы, а также описательная часть программы для ввода данных, начинающаяся словом ДАННЫЕ и присваивающая данным выбранные имена.

В языке Курс нет многомерных массивов, но их аналоги можно формировать, разбивая массив D на равные части. Например, многомерный массив A[0:N—1,0:M—1] = A[I, J] можно представить массивом D[N×M], записав D[1 + I + J × N]. Для примера приведем программу преобразования квадратной матрицы порядка N к треугольному виду методом исключения Гаусса.

Алгебраической операции $a_{ij}=a_{ij}-a_{ip}a_{pj}/a_{ii}$ над массивом данных *a* [0:N—1,0:N—1] соответствует выражение

D[1 + L + J × N]:=D[1 + L + J × N] — D[1 + L + I ×N] × × D[1 + I + J × N]/D[1 + I + I × N].

Приняв для определенности порядок матрицы $n=7$, составим текст программы:

```
ТЕКСТ ПРИВЕДЕНИЕ МАТРИЦЫ К ТРЕУГОЛЬНОМУ ВИДУ
ПАМЯТЬ D[50]
ДЛЯ I:=0:ДЛЯ J:=1;
ЕСЛИ D[1+I+J×N]≠0 ТО
⟨⟨ДЛЯ K:=1:Q:=1+K+I×N;
X:=D[Q]; D[Q]:=D[1+K+I×N];
D[1+K+I×N]:=X;
ДО K:K:=N—1; НА M⟩⟩;
ДО J:J:=N—1; ВЫВОД НУЛЬ; СТОП;
M:ДЛЯ J:=J+1:ДЛЯ K:=I+1;
```

```
Q: =I+K+J×N;
D[Q]: =D[Q]—D[I+K+I×N]×D[I+I+J×N]/D[I+I+I×N];
ДО K: K: =N—1;
D[I+I+J×N]: =0; ДО J: J =N—1;
ДЛЯ K: =0; ВЫВОД D[I+K+I×N];
ДО K: K: =N—1; ДО I: I: =N—2;
ДЛЯ K: =0; ВЫВОД D[I+K+(N—1)×N];
ДО K: K=N—1; КП
ДАННЫЕ ...
```

В качестве примера обращений к подпрограммам приведем программу пошагового решения задачи «Ханойская башня», заключающейся в переносе M дисков различного размера со стержня C на стержень HA так, чтобы ни в одном из положений больший диск не располагался над меньшим:

```
ТЕКСТ ХАНОЙСКАЯ БАШНЯ
ПАМЯТЬ D[30]
M: =10; N: =20; K: =1;
ВЫЗОВ: ПЛАН; СТОП;
ИМЯ ПЛАН: ЕСЛИ D[K]=0 ТО ⟨⟨ВЫЗОВ ЗАГРУЗЧИК; K: =K+1;
ВЫЗОВ ПЛАН; ВЫЗОВ ХОД;
ВЫЗОВ ВОЗВРАТ; ВЫЗОВ ПЛАН⟩⟩ ИН ⟨⟨ВЫЗОВ ХОД; K: =K—1⟩⟩
ВЫХОД; ИМЯ ЗАГРУЗЧИК: D[K+1]: =D[K]—1;
D[N+K+1]: =6—D[N+K]—D[M+K]; D[M+K+1]: =D[M+K];
ВЫХОД ИМЯ ВОЗВРАТ: D[K]: =D[K]—1;
D[M+K]: =6—D[N+K]—D[M+K];
ВЫХОД; ИМЯ ХОД; ВЫВОД ПЕРЕСТАВИТЬ C,D[M+K], HA,
D[N+K];
ВЫХОД; КП
ДАННЫЕ D[1]: =⟨n—1⟩; D[11]: =⟨C⟩; D[21]: =⟨HA⟩;
```

Оператор вывода может располагаться и в операторе условного перехода. Например, решение задачи перебора трехзначных чисел, сумма цифр которого равна заданному целому числу *n*, можно описать программой

```
ТЕКСТ ПЕРЕБОР ТРЕХЗНАЧНЫХ ЧИСЕЛ
ДЛЯ I: =0; ДЛЯ J: =0; K: =N—I—J;
ЕСЛИ K⩾1 ТО ≪ЕСЛИ K⩽9 ТО
ВЫВОД I+10×J+100×K≫;
ДО J: J=9; ДО I: I=9; КП
ДАННЫЕ N: =n;
```

В ЭВМ с традиционными алгоритмическими входными языками используют три этапа трансляции — лексический, синтаксический и семантический. Иероглифическое построение клавиатуры и ограниченный набор ключевых слов в языке Курс позволили исключить лексический этап трансляции. Основным в синтаксическом этапе трансляции является попарная проверка локальных и нелокальных сочетаний символов входной последовательности. Реакция на ошибочные локальные сочетания отображается звуковым сигналом, очисткой буфера входной строки и посылкой на экран и индикатор приглашения повторить ввод строки.

Анализ нелокальных сочетаний совмещен с семантическим анализом и основан на табличном алгоритме трансляции с шестью типами реакций на сочетания различных ограничителей. Исходными являются синтаксические диаграммы и таблица приоритетов, причем числовые константы, имена переменных, подпрограмм и меток обрабатываются отдельно. Выбор структуры и способа реализации транслятора обеспечил минимальные затраты ресурса памяти.

Низкое быстродействие элементной базы не позволило достичь показателей, необходимых для серийного производства ПМК «Электроника МК-72», но выбранные при его разработке архитектурные решения позволили определить перспективные пути создания ПМК новых поколений.

6.3. МАССОВЫЕ ПМК С ВХОДНЫМ ЯЗЫКОМ БЕЙСИК

Разработка относительно недорогих в серийном производстве матричных ЖКИ и БИС памяти большой емкости обусловили создание массовых ПМК с алгоритмическими входными языками, но и в этом случае первостепенным остается критерий минимальной стоимости.

В комплекте универсальных микроЭВМ относительно дорогими в изготовлении остаются внешние устройства с механическими и электромеханическими узлами (особенно графопостроители, печатающие устройства, накопители на магнитных и оптических носителях информации), а также разъемы многопроводных шин с покрытиями контактов из драгоценных металлов или их заменителей. Несмотря на значительное увеличение в последние годы объема мирового производства таких устройств, их стоимость, в особенности при необходимой для ПМК миниатюризации, снижается значительно медленнее, чем стоимость электронных компонентов. Поэтому при разработке массовых ПМК предусматривают возможность их работы без внешних устройств или с их минимальным комплектом, приобретаемым пользователем независимо от покупки ПМК. В качестве внешних устройств для массовых ПМК с алгоритмическими языками обычно используют кассетные магнитофоны, чаще всего бытовые, и печатающее устройство, часто совмещаемое с модулем интерфейса.

В качестве входного языка в таких ПМК почти исключительно используются различные версии Бейсика. Одним из первых и наиболее

известных производителей массовых ПМК с Бейсиком является японская фирма Sharp, которой принадлежит и первенство в разработке миниатюрных бытовых ПЭВМ, по габаритам и массе близких к ПМК. В табл. 6.5 приведены некоторые технические данные для нескольких распространенных массовых ПМК этой фирмы с однострочными матричными ЖКИ. В графе «Разрядность» указано число выводимых одновременно на индикатор символов программы и в скобках число десятичных разрядов мантиссы/порядка машинных представлений чисел. В графе «Число шагов» указана емкость области пользователя ОЗУ в 8-разрядных программных словах (шагах).

Таблица 6.5

Данные массовых ПМК фирмы Sharp

Тип	Разрядность	Емкость, Кбайт		Число шагов	Масса, г
		ОЗУ	ПЗУ		
PC-1246	16(10/2)	2	14	1278	115
PC-1401	16(10/2)	4,2	40	3534	150
PC-1402	16(10/2)	10,2	40	9678	150
PC-1421	16(10/2)	4,2	40	3534	115

В ПМК PC-1421 и PC-1246 предусмотрены специальные клавиши для вызова заранее составленных пользователем и хранимых в энергонезависимой памяти прикладных программ, которые могут использоваться как самостоятельно, так и в качестве подпрограмм основной программы. Подобная возможность, реализуемая различными способами, предусмотрена и в других ПМК с алгоритмическими языками.

Достаточно производительными являются ПМК PC-1401 и PC-1402, отличающиеся лишь числом БИС оперативной памяти, из которой 500 байт используется для операционной системы. В остальном эти ПМК идентичны: имеют одинаковую массу (150 г), габаритные размеры (170 × 72 × 9,5 мм) и потребляемую мощность 0,03 Вт от 6-вольтовой батареи из двух алюмокалиевых элементов. Матричный ЖКИ обеспечивает отображение до 16 символов (5 × 7 элементами каждый) при общем числе до 80 символов в строке программы.

Клавиатура этих ПМК содержит 76 клавиш со стандартным для ПЭВМ расположением (QWERTY). На печатной плате под клавиатурой размещены БИС 8-разрядного центрального процессора, контроллера индикаторного устройства, ПЗУ и одна или две (в PC-1402) БИС ОЗУ, а также элементы цепи питания и кварцевой стабилизации. ПМК имеют разъем для присоединения кассетного магнитофона или печатающего устройства с блоком интерфейса.

В качестве входного языка использована достаточно развитая версия Бейсика с операционной системой, диагностирующей 9 типов

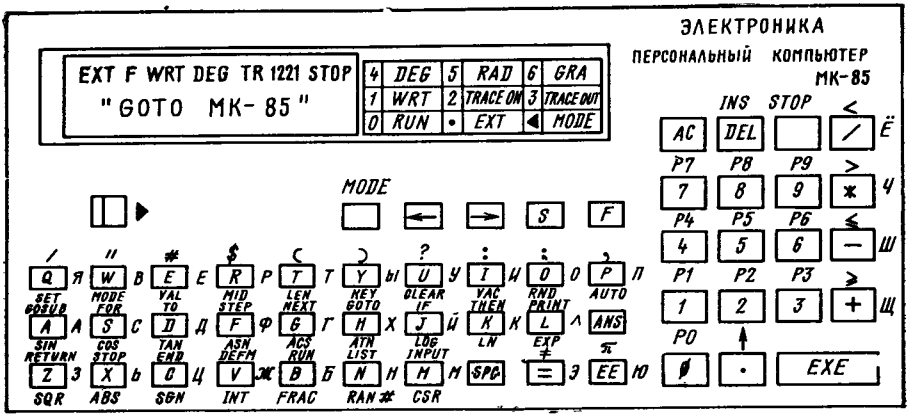


Рис. 6.5

ошибок в прикладных программах. Часть из 59 встроенных функций используется только в режиме непрограммируемого микрокалькулятора.

К рассматриваемому классу относится и отечественный ПМК «Электроника МК-85» [12] с входным языком Бейсик (габаритные размеры 163×71×13 мм, масса 150 г и потребляемая мощность 0,02 Вт от четырех элементов типа СЦ-0,18). Предусмотрена возможность подключения внешнего блока питания и использования режима повышенного быстродействия при большей потребляемой мощности.

На передней панели этого ПМК (рис. 6.5) расположен матричный ЖКИ, таблица с кодами режимов работы и клавиатура. Индикатор обеспечивает отображение до 12 символов (при максимальном числе 63 символа в строке программы) и указателей режимов работы EXT (extention — расширение) для расширения функций и ввода комментариев, RUN для установления рабочего режима, WRT (write — записывать) для ввода и редактирования программ, TRACE ON (trace — прослеживать) для перехода к отладке программ, TRACE OUT для выхода из режима отладки. Кроме того, индицируются символы нажатия префиксных клавиш S и F при вводе элементов алфавита, обозначеиных соответственно над или под клавишами, размерности углов в градусах (DEG), радианах (RAD) и десятичных градусах или градах (GRA), а также число свободных шагов программной памяти.

Клавиатура содержит 54 клавиши для ввода программ и вычислений со стандартным для ПЭВМ расположением (QWERTY). Режим работы устанавливается нажатием клавиш MODE и кода режима. Клавиша MODE используется также для ввода в комментариях русских букв, обозначенных справа от программных клавиш.

Решение прикладных задач возможно как в режиме непрограммируемого микрокалькулятора, так и в режиме автоматического выпол-

нения программ, введенных на используемой версии языка Бейсик. В обоих случаях вычисления выполняются в соответствии с правилами скобочной алгебраической записи с автоматическим учетом приоритета операций — выражений в скобках, стандартных функций, возведения в степень, умножения, сложения и вычитания. Машинное представление чисел соответствует 10 десятичным разрядам мантиссы и 4 порядка, что обеспечивает широкий диапазон представления чисел.

В режиме калькулятора (MODE Ø) вычисления выполняются после нажатия клавиши EXE (execution — исполнение), например, после нажатия последовательности клавиш

MODE Ø F FRAC S (1 5 / 4 S) + 5 × 2 EXE

на индикатор выводится результат 10.75, так как умножение выполняется перед сложением. Для округления результатов вычислений предназначен оператор RND (round — округлить).

В обоих основных режимах работы могут использоваться стандартные (встроенные) функции вычисления прямых SIN, COS, TAN и обратных ASN, ACS, ATN тригонометрических функций, натурального LN и десятичного LG логарифмов, экспоненты EXP, корня квадратного SQR, результатов арифметических операций и возведения в степень ↑ положительного числа, определения модуля ABS, целой INT и дробной FRAC частей, а также знака SGN числа, задаваемого именем переменной или выражением. Кроме того, имеются операторы RAN# для вызова квазислучайного числа с равномерным распределением в интервале (0,1) и числа π.

Аргумент тригонометрических функций определяется неравенством $|x| < 8\pi(1440°)$, для функций $\arcsin x$ и $\arccos x$ область аргумента $|x| < 1$, а для $\operatorname{arctg} x$ допускаются значения $|x| < 10^{4084}$. Область определения $\ln x$ и $\lg x$ ограничена интервалом $0 \leqslant x \leqslant 10^{4084}$, e^x — интервалом $-9429 < x < 9429$ и $\sqrt{x}$ — интервалом $0 \leqslant x \leqslant 10^{4084}$.

Режим ввода программ на Бейсике устанавливается нажатием клавиш MODE. 1. В этом режиме используют до 26 переменных, обозначаемых буквами английского алфавита, но в режиме расширения EXT с помощью оператора DEFM допустимо вводить одномерный массив, содержащий до 178 операндов при полном использовании области программной памяти. Эта часть памяти пользователя разбита на 1221 8-разрядных шагов. Метка (номер строки) занимает 2 шага, остальные символы и ключевые слова — один шаг. Например, оператор

110 FOR X = 1 TO 5

занимает 8 шагов, из которых 2 занимает метка. Исключением являются элементы одномерных массивов, каждый из которых занимает 8 шагов программной памяти.

В энергонезависимой памяти могут постоянно храниться до 10 прикладных программ, составленных пользователем, в виде файлов P0...P9, которые могут вызываться для использования в

качестве и самостоятельных программ, и подпрограмм основной программы.

Словарный запас используемой версии Бейсика содержит 21 оператор с именами INPUT, PRINT, GOTO, FOR...NEXT, IF ... THEN, GOSUB, RETURN, STOP, END, RUN, LIST, MODE, SET, VAC, CLEAR, CLEARA, DEFM, AUTO, DRAW, DRAWC и программными функциями KEY, CSP, LEN, MID, VAL.

Программа вводится по строкам, начинающимся с порядковой метки, содержащей до трех десятичных цифр, и оканчивающимся нажатиями клавиши EXE. В одной строке можно размещать более одиого оператора, если управление в программе не передается внутреннему оператору, но общее число символов строки не должно превышать 63.

Арифметические операторы присваивания вводят без ключевого слова LET, обязательного в базовой версии языка. Для ввода исходных данных предназначен также оператор ввода

INPUT <«текст», имена переменных>

для запроса числового значения переменной. При считывании кода такого оператора выполнение программы прекращается до ввода числовой переменной, имя которой отображено на индикаторе.

Основная форма оператора условного перехода

IF <Условие> THEN <Метка | Указатель файла>,

где после ключевого слова THEN может быть записана или метка перехода на нужную строку, или указатель # PN хранящейся в памяти программы (файла), используемой в качестве встроеиной функции.

Допускается использование оператора условного перехода общего вида

IF <Условие>; <Оператор, выражение>,

например IF A > B; A = B.

В операторе безусловного перехода GOTO метка перехода может быть задана арифметическим выражением. Например, при считывании в программе оператора GOTO A * B + 40 будет выполнен переход к строке с меткой, равной результату вычислений по формуле A × × B + 40.

Оператор перехода к подпрограмме
GOSUB <Метка|Указатель файла>

может содержать метку перехода к подпрограмме или указатель #PN хранящегося в памяти файла (P0...P9). Возвращение из подпрограммы, за исключением файлов, обеспечивается оператором RETURN, записываемым в конце подпрограммы.

Оператор вывода на индикатор
PRINT <«Символы», Имена переменных>

выводит на индикатор обозначенную кавычками последовательность элементов алфавита, включая комментарии на русском или английском языке, и числовые значения переменных с указанными именами. Например, после выполнения оператора PRINT «ФАКТОРИАЛ =» X на индикатор выводится ФАКТОРИАЛ = x, где x — числовое значение переменной X. Для указания числа пробелов в оператор PRINT вводится оператор CSR n, где n — число пробелов. Так, после выполнения оператора PRINT «X=»X; CSR 8; «A=» A на индикатор выводится X = x и через 8 пробелов A = a, где x и a — числовые значения переменных X и A.

Оператор цикла состоит из заголовка

FOR <Имя переменной> = n_1 TO n_2 STEP n_3,

где n_1, n_2 и n_3 — числа, соответствующие граничным значениям переменной и ее приращению на итерацию, и после тела цикла окончания NEXT <имя переменной>. Если $n_3 = 1$, то шаг не указывают, записывая заголовок FOR <имя переменной> = n_1 TO n_2.

Для вывода результатов вычислений в цикле может использоваться оператор останова, например в фрагменте

```
20 FOR X=1 TO 50 STEP 2
30 A=X↑2+15
35 STOP
40 NEXT X
```

При выполнении этого фрагмента после каждого вычисления переменной A ее значение будет индицироваться, и для продолжения вычислений достаточно нажать клавишу RUN.

В конце программы записывается оператор END, после которого по порядку меток могут следовать лишь строки с обращением к ним операторами GOTO или GOSUB.

Оператор RUN<Метка> используется для пуска программы со строки с указанной меткой. Оператор SET <Десятичная цифра> ограничивает число значащих цифр результата вычислений указанной цифрой. Для отмены указания вводят SET 10.

Оператор LIST <Метка> выводит на индикатор текст программы, начиная со строки с указанной меткой. В режиме RUN строки «перелистываются» автоматически (в режиме бегущей строки), в режиме WRT переход на новую строку выполняется после нажатия клавиши EXE.

Оператор VAC используется для очистки ячеек программной памяти. Оператор CLEAR стирает программу в файле, с которым выполняется основная программа, а оператор CLEARA — содержимое всех файлов P0...P9.

При редактировании программы используют оператор DEL для стирания одного символа и сдвига программы на один шаг влево, оператор INS для сдвига программы на один шаг вправо и ввода дополнительного символа, курсоры ← и → для сдвига индицируемого текста

программы и команду AC очистки иидикатора. Оператор AUTO <Шаг нумерации строк> обеспечивает автоматическую нумерацию строк программы метками с заданным шагом.

Ряд операторов входного языка предназначен для отображения и преобразования символьной информации. Символьные операнды могут содержать до 7 символов, обозначаемых знаком ¤. Для операций над ними используют оператор KEY (ключ), записываемый как <Символьная переменная> = KEY. Например, при выполнении оператора

```
10 A ¤=KEY:IF A ¤=«X» THEN 30
```

переход к строке с меткой 30 выполняется, если символьная переменная A совпадает с символом X.

Оператор
<Переменная> = LEN <Текстовая переменная>

определяет заданное переменной число символов текстовой переменной, выводимых на индикатор оператором PRINT.

Оператор MID (m, n), где m и n — целые числа или выражения с целочисленными результатами от 0 до 30, выводит на индикатор из строки символов, являющейся текстовой переменной, n сиволов, начиная с m-го. Например, если $ = ABCDEFI, то после выполнения фрагмента

```
50 X$=MID(3,4)
60 PRINT X$
70 STOP
```

на индикатор выводятся символы CDEF. Если число n не задано, то выводятся все символы текстовой переменной, начиная с m-го.

Оператор VAL (текстовая переменная) заменяет текстовую переменную присвоенным ей числовым значением. Например, если X = = 123, то VAL (X) формирует число 123.

Область программной памяти (1221 программных слов) обеспечивает хранение и выполнение прикладных программ, содержащих примерно до 150 строк.

Операционная система ПМК диагностирует 7 видов ошибок в прикладных программах — переполнение памяти, синтаксические, математические, связанные с попыткой выполнения стандартных функций вне областей их определения и делением на нуль, ошибки в организации циклов, задании переменных и их числовых значений, обращений к подпрограммам и файлам. При диагностировании ошибки на индикаторе указывается тип (номер) ошибки, номер строки или файла с ошибкой.

Основными структурными элементами ПМК являются центральный процессор (ЦП), контроллер клавиатуры (контроллер ввода-вывода) КК, клавиатура, ЖКИ, ОЗУ и ПЗУ, генератор напряжений (ГН), подаваемых на элементы индикатора (рис. 6.6).

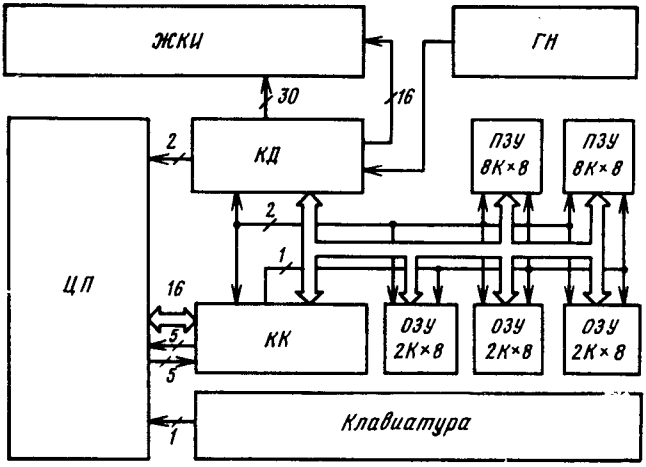


Рис. 6.6

Центральный процессор однокристальный 16-разрядный с 8 регистрами общего назначения и максимальной тактовой частотой 2 МГц. Число команд 72 (безадресных, одно- и двухадресных), причем используется прямая и косвенная регистровая, автоинкрементная и индексная адресация. Процессор соединен с контроллером клавиатуры через 16-разрядную информационную шину и линии передачи управляющих сигналов — синхронизирующих, запроса на прямой доступ к памяти и начального запуска процессора.

Контроллер клавиатуры обеспечивает обмен данными между ЦП и ЗУ, регенерацию данных для формирования изображений на ЖКИ, ввод кодов символов и операторов с клавиатуры, прерывание работы процессора. Клавиатура соединена с 11-разрядной шиной и линией прерывания (HALT) непосредственно с процессором. Данные с клавиатуры засылаются в буферный регистр для хранения информации в течение нажатия любой клавиши.

Обмен данными между КК и ЗУ реализуется по 8-разрядной шине данных, причем обмен 16-разрядными словами реализуется за два такта. Адреса пересылаются по 13-разрядной шине с синхронизацией выборки адресуемых слов сигналами CEROM и CERAM по управляющей шине. Через каждые 0,4 с работа процессора прерывается по требованию прямого доступа к ЗУ с контроллера клавиатуры сигналом логического уровня (DMP). На адресной шине контроллер автоматически формирует серию адресов, передача которых управляется сигналами CERAM (синхронизация выборки данных) и CEHC (синхронизация данных для изображения на индикаторе).

Контроллер клавиатуры через системную магистраль и шины управления связан с контроллером дисплея (КД), управляющего работой ЖКИ сигналами строк, передаваемых по 16-разрядной шине, и сигналами столбцов и указателей режимов, передаваемых по 30-разрядной

шине. Напряжения на индикатор, работающий в мультиплексном режиме 1 : 16, формируются генератором напряжения, причем на строки подаются однополярные напряжения, а на столбцы — напряжения различной полярности. При совпадении полярности напряжений строки и столбца высвечивается элемент матрицы, расположенный на пересечении соответствующих строки и столбца, а при несовпадении полярностей такой элемент не возбуждается.

Емкость адресуемой памяти ПМК 64 Кбайт, причем адресное пространство ЗУ распределено следующим образом: ПЗУ1 присвоены адреса слов от 00000 до 17777, ПЗУ2 — от 20000 до 37777, ОЗУ — от 40000 до 43777. Область ОЗУ с адресами от 40000 до 40137 предназначена для обеспечения индикатора (экранное ОЗУ), по адресам от 40140 до 41471 хранится информация, вызываемая из операционной системы (в частности, интерпретатор Бейсика), адресами от 41472 до 43777 определяется область пользователя для хранения прикладных программ. Возможно расширение ОЗУ до 7365 программных слов (предельный адрес 47777) или 15557 (до адреса 77777) при использовании соответственно одной или двух БИС ОЗУ с организацией 8К × 8. В ПМК использованы микросхемы ПЗУ типа Т242-2 с организацией 8К × 8, изготовленные по КМОП-технологии, и микросхемы ОЗУ статического типа Т244-2 с организацией 2К × 8.

Массовые ПМК с Бейсиком, подобные ПМК «Электроника МК-85», являются удобным и достаточно мощным носимым средством автоматизации решения задач обработки информации.

6.4. РАЗВИТИЕ ПРИКЛАДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Решение прикладных задач обработки информации в основном сводится к решению последовательности типовых математических задач. Производительность труда пользователя ЭВМ значительно повышается, когда он использует готовые программы решения типовых задач, не затрачивая время на их составление и отладку. Особенно важно использование готовых прикладных программ, подготовленных квалифицированными специалистами, для пользователей ПМК, не имеющих глубокой подготовки по программированию и математике. Поэтому ПМК со времени их создания снабжаются библиотеками прикладных программ в виде записей на входных языках, печатаемых в руководствах по применению МПК и справочных пособиях.

Повышение производительности ПМК и расширение емкости ОЗУ привело к повышению предельной сложности решаемых задач и соответствующему увеличению длины прикладных программ. Это ограничивает целесообразность применения печатных библиотечных программ в связи со значительными затратами времени на их ввод с клавиатуры. Поэтому повышение производительности ПМК сопровождалось внедрением и развитием встроенных и сменных накопителей информации и ПЗУ для хранения прикладного программного обеспечения, а также вспомогательных программ, расширяющих возможности

операционной системы. При этом возможны различные способы представления содержащейся в таких программах информации.

Простейший из способов — представление прикладных программ в виде последовательностей двоичных кодов операторов прикладных программ на входном языке. При вызове программ с накопителя информации они транслируются и загружаются в оперативную память, что связано с затратами времени. Кроме того, программа на входном языке высокого уровня не обеспечивает полного использования ресурсов памяти и быстродействия ПМК. Поэтому такой способ используют лишь для накопления программ, составляемых пользователями.

Более эффективный способ — составление прикладных программ, хранящихся во внешних накопителях и ПЗУ, на языке машинных команд. Такие программы, составленные профессиональными программистами, оказываются, как правило, более производительными и могут загружаться непосредственно в ОЗУ без предварительной трансляции с языка высокого уровня.

Наиболее эффективный способ подготовки и использования прикладного программного обеспечения, хранящегося во внешних накопителях или ПЗУ, связан с закреплением за программными словами адресов из адресного пространства центрального процессора. В этом случае при соответствующей организации операционной системы команды прикладных программ непосредственно вызываются для обработки в центральный процессор без их загрузки в ОЗУ, в которой хранятся лишь текущие данные. Объем области памяти пользователя в этом случае ограничивает максимальное число текущих данных, но эта область также может быть расширена при использовании внешних ЗУ (ВЗУ). Следовательно, предельная сложность прикладных задач при использовании внешнего прикладного математического обеспечения* ограничивается лишь быстродействием ПМК и стоимостью внешних устройств и математического обеспечения.

Относительно развитое прикладное программное обеспечение использовано в таких высокопроизводительных микрокомпьютерах с компактными входными языками, как HP-41C и его модификации, а также P-58 и P-59 фирмы Texas Instruments. Популярность этих ПМК среди специалистов различного профиля в значительной мере определяется большим ассортиментом прикладных программ, хранимых на покупных магнитных карточках и сменных полупроводниковых модулях ПЗУ.

Значительно бо́льшие возможности повышения производительности труда пользователей связаны с созданием высокопроизводительных ПМК с алгоритмическими входными языками, большим адресным пространством ЦП и развитым комплектом внешних устройств, представляющих миниатюризованные аналоги внешних устройств профессиональных ПЭВМ. Примером таких ПМК могут служить HP-94 и осо-

*Прикладное математическое обеспечение кроме прикладных программ может содержать таблицы функций и другую информацию.

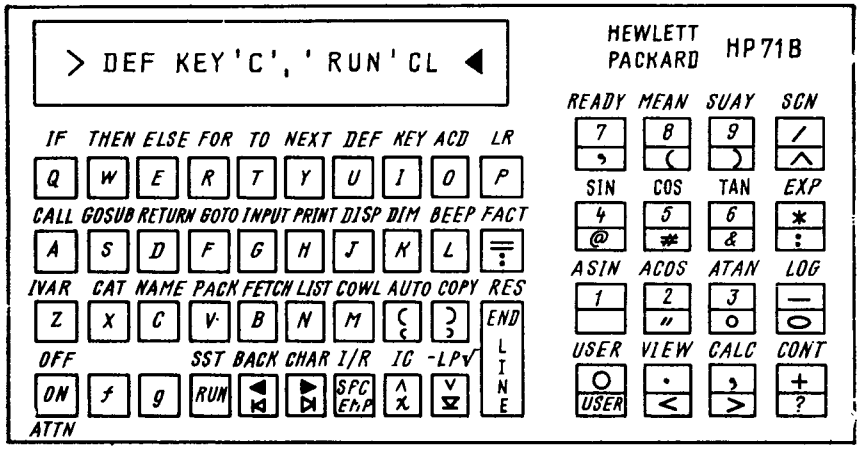


Рис. 6.7

бенно HP-71B, предназначенный для научных и инженерно-технических работников [28].

На передней панели ПМК HP-71B (рис. 6.7) (габаритные размеры 190 × 77 × 25 мм и масса 250 г) расположен матричный ЖКИ с числом элементов 132 × 9, обеспечивающий одновременное отображение до 22 символов (при максимальном числе 96 символов в строке программы) и указателей режима, а также программно формируемых неподвижных и подвижных изображений. Клавиатура содержит только 55 клавиш (стандарта QWEPTY), но операторы программы можно набирать нажатием клавиш ключевых слов, или букв английского алфавита.

Основными структурными элементами ПМК кроме клавиатуры и ЖКИ (рис. 6.8) являются центральный процессор (ЦП), три контроллера дисплея (КД), каждый с собственным ОЗУ емкостью 1 Кбайт, четыре однотипных БИС ОЗУ с областью пользователя емкостью 17,5 Кбайт, ПЗУ емкостью 64 Кбайт.

Внутренние 64-разрядные регистры центрального процессора для хранения слов с 20-разрядными адресами обеспечивают адресное пространство 512 Кбайт или 1 Мнибл (nibble — тетрада битов). Входной 16-разрядный регистр обеспечивает опрос клавишей и формирование кодов символов, а выходной 12-разрядный регистр — управление портами виешних устройств и пьезоэлектрическим зуммером с двумя акустическими уровнями. Обработка и передача информации выполняются по 4 двоичных разряда в течение рабочего такта, что соответствует одной десятичной или шестнадцатеричной цифре.

Системная магистраль управляется 4-разрядными адресами, а обмен данными при мультиплексировании — стробовыми сигналами STR и сигналами команд/данных C/D, пересылаемых по отдельным управляющим линиям.

В адресном пространстве для ПЗУ операционной системы выделены адреса от 00000 до 1FFFF, интерфейса накопителя на магнитных карточках — от 20000 до 20C01F, элементов матрицы ЖКИ — от 2E100 до 2F3FF, ОЗУ контроллеров дисплея — от 2F400 до 2FFFF, интерфейса ввода-вывода — от 2F400 до 2C000. Области пользователя выделена часть адресного пространства, начиная с адреса 30000, причем граница между областями встроенного ОЗУ и внешних расширителей ОЗУ определяется вводом указателей.

Действительные числа представлены 12 десятичными разрядами мантиссы и 3 порядка, причем в основном выполнены требования стандарта IEEE для представления чисел с плавающей запятой. В частности, стандарт предусматривает указатели Ind и NaNs для индикации попадания результатов операций соответственно в области машинных бесконечности и нуля, что упрощает отладку программ и повышает достоверность результатов их выполнения.

HP-71B работает как в режиме непрограммируемого калькулятора, так и в режиме ввода и выполнения программ пользователя, причем для ввода программ в качестве основного входного языка использована расширенная версия Бейсика, но могут использоваться также язык ассемблера и язык высокого уровня Форт, трансляторы которых хранятся в сменном модуле ПЗУ.

Основная особенность HP-71B — развитое прикладное математическое обеспечение, хранимое во внешних накопителях и сменных модулях памяти. ПМК HP-71B имеет четыре разъема для присоединения сменных модулей памяти и модуля интерфейса, через который производится обмен информацией с разнообразными фирменными внешними устройствами, включая печатающие устройства и накопители на гибких магнитных дисках, а также оригинальные накопители на магнит-

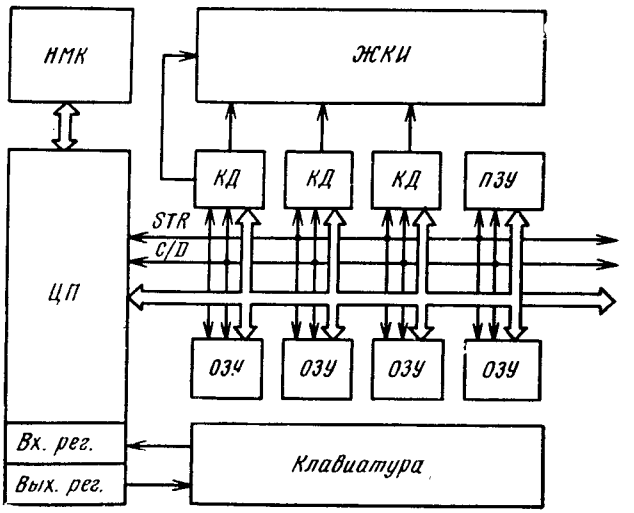


Рис. 6.8

ных карточках (НМК) емкостью 1300 байт каждая. В ранее применяемых устройствах записи-считывания информации для протяжки магнитных карточек использовался электродвигатель, в HP-71B — ручная протяжка карточек, что существенно повысило надежность работы накопителя. Автономное питание от 6-вольтовой батарейки из двух алюмокалиевых элементов.

Возможности решения сложных прикладных задач существенно расширяются при использовании прикладного математического обеспечения, в частности модулей ПЗУ с пакетами программ емкостью 64 Кбайт каждый. Одна из особенностей математического обеспечения, хранящегося в сменных модулях, — объявление комплексных переменных и внутренних постоянных способом, характерным для универсальной ЭВМ высокой производительности IBM 370. При подключении модуля в операционную систему ПМК вводятся новые типы данных COMPLEX и SHORT COMPLEX, применяемые как для переменных, так и для массивов.

Данные COMPLEX вводятся с полной точностью, а данные SHORT COMPLEX представлены 5 десятичными разрядами мантиссы и 3 порядка. Представление комплексных чисел является средним между представлением действительных чисел и массивов: каждое комплексное число представлено парой отделяемых запятой действительных чисел, отображающих действительную $Re x$ и мнимую $Im x$ части комплексного числа. Предусмотрено вычисление не только арифметических операций над комплексными числами, но и всех стандартных функций комплексного аргумента. Кроме того, при работе с пакетом прикладных программ могут использоваться и выводиться на индикатор дополнительные комплексные функции, например ARG (аргумент комплексного числа), CONJ (сопряженное комплексное число), POLAR (полярная система координат), RECT (прямоугольная система координат).

Например, вычисления по формуле

$$x = \sin((1+j2)^{(4+j5)} + \Gamma(-2{,}3)(3-j5)^*)/(e^{(2+j4)}-1)$$

на версии Бейсика HP-71B отображаются строкой

```
X = SIN((1,2)∧(4,5)) + GAMMA(—2,3) * CONJ(3,—5))/EXP((2,4) — 1)
```

с результатом вычислений, отображаемым на индикаторе парой действительных чисел (3.0491011995, .49611522894).

В пакете программ предусмотрено также выполнение всех стандартных операций над матрицами, включая арифметические операции, вычисление определителей, обращение и транспонирование (табл. 6.6).

Вычисление корней нелинейных уравнений реализовано оператором ENROOT (x_1, x_2, x_3), где x_1 и x_2 — граничные значения начального интервала корня, а x_3 — выражение для невязки уравнения с искомым приближением корня FVAR. Например, решение уравнения $(R\cos u)^2 - u^2 = 0$ при R = 0,6, X = 1, X1 = 2 обеспечивается оператором

```
FNROOT (X, X1, ABS((0,6) * COS(X1, FVAR)∧2 — (X1, FVAR∧)2)).
```

Таблица 6.6

Матричные операторы пакета программ ПМК НР-71В

Оператор	Название
FNORMA	Норма Фробениуса
PNORMA	Векторная норма $l=\infty$
CNORMA	Векторная норма $l=1$
DOT (A, B)	Внутреннее или сопряженное внутреннее произведение
UBND (A, B)	Нижняя граница массива индексов
LBND (A, B)	Верхняя граница массива индексов
DET (A)	Определитель матрицы А
DETL	Определитель матрицы до обращения
MAT A=CON	Матрица коэффициентов заданной размерности
MAT A=ZER	Нулевая матрица
MAT A=B	Присвоение имени матрицы
MAT A=—B	Изменение знака матрицы
MAT A=B+C	Сложение матриц
MAT A=B—C	Вычитание матриц
MAT A=B*C	Умножение матриц
MAT A=(*x*)	Присвоение скалярных значений матричным элементам
MAT A=INV (B)	Обращение матрицы
MAT A=(*x*)*B	Умножение матрицы на скаляр
MAT A=TRN (B)	Транспонирование или сопряженное транспонирование
MAT A=SYS (B, C)	Решение системы линейных уравнений
MAT A=TRN (B) C	Транспонирование произведения матриц

Численное интегрирование выполняется по интерполяционной схеме Ромберга с помощью оператора INTEGRAL (*a*, *b*, *c*, *y*), где *a* и *b* — границы интервала интегрирования; *c* — заданная предельная абсолютная погрешность результата; *y* — подынтегральная функция с аргументом IVAR. Например, вычисление определенного интеграла

$$\int_0^2 e^{x^2-x}\,dx$$

с погрешностью $c \leqslant 10^{-12}$ обеспечивается оператором

```
INTEGRAL (0,2,1E—12), EXP(IVAR^2 — IVAR)).
```

Для вычисления кратного интеграла

$$\int_1^5 x \int_0^x y \operatorname{ch} y\,dy\,dx$$

с той же предельной погрешностью можно использовать оператор

```
INTEGRAL (1,5,1E—12, IVAR*INTEGRAL (0,IVAR*COSH(IVAR))).
```

Если переменные в подынтегральном выражении неразделимы, то может быть использована функция пользователя, например, в фрагменте программы

```
10  DEF FNF(X) = INTEGRAL (4,5,1E—12, SQR(1+4(X↑2+IVAR↑2))
20  DISP INTEGRAL(1,2,1E—12, FNF(IVAR)).
```

Выполнение первой строки этого фрагмента обеспечнвает вычисление интеграла

$$\int_4^5 \sqrt{1+4\,(x^2+y^2)}\,dx,$$

а после выполнения второй строкн будет вычнслен интеграл

$$\int_1^2\int_4^5 \sqrt{1+4\,(x^2+y^2)}\,dydx.$$

Среди разнообразных других возможностей прикладного математического обеспечения HP-71B следует отметить вычисление всех корней алгебраических уравнений с многочленами до 286-й степени методом Лагерра.

Приведенные примеры свидетельствуют о высокой эффективности прикладного математического обеспечения, хранящегося на внешних модулях ПЗУ. Однако повышение производительности ПМК применением внешних устройств, являющихся по существу миниатюризованными аналогами внешних устройств профессиональных ПЭВМ, связано с высокой стоимостью их производства. Достаточно указать, что ПМК HP-71B, разработанный в 1984 г., в связи с его высокой стоимостью до 1989 г. изготовлялся лишь по заказам, причем не с полным комплектом внешних устройств. Это объясняется тем, что подобный ПМК, предназначенный для работы с комплектом внешних устройств, по способу его применения аналогичен настольной ПЭВМ, но стоимость последних при такой же производительности оказывается меньше в связи с большими габаритами. При использовании HP-71B в носимом варианте, являющемся основным достоинством ПМК, он не имеет существенных преимуществ перед другими карманными компьютерами.

Основной путь повышения производительности ПМК именно как носимого, а не настольного микрокомьютера в основном связан с повышением уровня операторов входного языка, обеспечивающих решение достаточно сложных типовых математических задач, и развитием «дружественных» пользователям операционных систем, освобождающих от вспомогательных операций. Среди последних с наибольшими затратами времени связаны предварительная запись на бумаге расчетных выражений и их преобразования с последующим составлением программы решения прикладной задачи на входном языке.

Примером микрокомпьютеров с простейшими «дружественными» операционными системами, обеспечивающих уменьшение необходимости в записи расчетных формул на бумаге, могут служить EL-5000 и EL-5001 формы Sharp. Эти ПМК предназначены для решения вы-

числительных задач при наборе с клавиатуры и отображении на ЖКИ последовательностей соответственно до 48 и 80 расчетных выражений. Эти выражения вводят по одному в соответствии с правилами алгебраической записи символов переменных, а исходные данные вводят по запросу ПМК, реализуемого миганием на индикаторе символа соответствующей переменной. Результаты вычислений с выбранными именами сохраняются в памяти и используются при вычислении вводимых следующими выражений.

Во многих ПМК используются встроенные функции (операторы) высокого уровня, обеспечивающие автоматическое решение типовых математических задач по достаточно сложным алгоритмам. Примером могут служить ПМК HP-12C (см. табл. 6.1) с встроенными операторами для решения типовых экономических задач и HP-11C с операторами, обеспечивающими, например, численное интегрирование, решение нелинейных уравнений и операции над комплексными числами. Подобная тенденция повышения удобства пользования ПМК наиболее полно реализована в «Деловом консультанте» HP-18C и «Калькуляторе для профессионалов» HP-28C, разработанных в 1987 г. [29]. Конструктивно эти ПМК оформлены в идентичных корпусах типа «раскрывающаяся книжка» с габаритными размерами 190 × 165 × 13 мм в раскрытом виде, массой 230 г и питанием от батареи из алюмокалиевых элементов с максимальным напряжением 6 В (минимальное рабочее напряжение 3 В) или внешнего источника питания.

Оба ПМК имеют однотипные клавиатуры с 72 клавишами, отличающиеся лишь обозначенными на них элементами алфавита, и одинаковые матричные ЖКИ, отображающие до четырех строк текста, содержащего до 23 символов, а также графические изображения.

Рассматриваемые ПМК могут дополняться единственным внешним устройством для термопечати на бумажную ленту шириной 58 мм по 23 знака в строке. Связь ПМК с печатающим устройством, имеющим независимое питание, реализуется последовательным каналом связи в инфракрасном диапазоне частот со светодиодным излучателем в ПМК, что существенно повышает надежность связи по сравнению с проводными линиями, имеющими электромеханический разъем. Печатающее устройство располагается на расстоянии до 50 см от ПМК и начинает работу при нажатии клавиши PRINT на клавиатуре микрокомпьютера.

В обоих рассматриваемых ПМК используется быстродействующий центральный процессор HP-71B, способный работать при низких напряжениях питания (до 2,7 В) и небольшом потреблении энергии. Скорость выполнения однотипных операций в HP-18C в 15 раз больше, чем в HP-12C, а в HP-28C — в 20 раз больше, чем в HP-15C.

Центральный процессор (ЦП) этих ПМК соединен 13-разрядной входной шиной через входной регистр с клавиатурой, а 8-разрядный выходной регистр обеспечивает управление зуммером (З) и через системную шину обмен информацией с остальными основными структурными элементами (рис. 6.9). Матричный ЖКИ управляется двумя

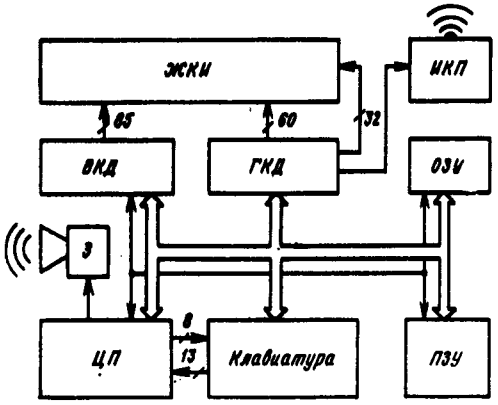


Рис. 6.9

контроллерами дисплея. Вспомогательный контроллер (ВКД) управляет формированием напряжений на столбцах, а главный контроллер (ГКД) вырабатывает напряжение строк и части столбцов (при мультиплексировании 1 : 32), а также управляет работой инфракрасного светодиодного передатчика (ИКП) информации на печатающее устройство.

Емкость ОЗУ 2 Кбайт, емкость ПЗУ для хранения операционной системы HP-18C 64 Кбайт (одна БИС ПЗУ), а HP-28C 128 Кбайт (2 БИС ПЗУ). «Дружественные» пользователю операционные системы этих ПМК обеспечивают возможность непосредственного ввода с клавиатуры и отображения на индикаторе расчетных формул с символьными или числовыми значениями переменных без их записи на бумаге, а операторы высокого уровня обеспечивают выполнение достаточно сложных типовых процедур.

При этом HP-18C используют, в частности, в режимах решения типовых экономических задач: FIN (для оценки изменения стоимости денег при финансовых операциях), BUS (для оценки текущего изменения стоимости), SUM (для анализа суммарного процесса изменения стоимости со статистическим моделированием изменения одной переменной при выборе одной из четырех возможных моделей ситуаций), TIME (изменения во времени текущих данных с вычислением оценок до шести вызывающих беспокойство ситуаций) и SOLVE для решения различных задач. В последнем режиме возможно, в частности, решение нелинейных уравнений с выводом на индикатор графика невязки уравнения в заданном масштабе с указанием курсором границ интервала нужного корня уравнения.

В отличие от HP-18C, предназначенного для решения деловых и коммерческих задач, ПМК HP-28C обеспечивает возможность решения в удобной для пользователя форме шикорого круга инженерно-технических задач.

По сравнению с любыми другими ручными компьютерами, включая высокопроизводительный HP-71B с его сменными пакетами математического обеспечения, ПМК HP-28C обеспечивает пользователя наибольшими возможностями решения математических задач. Более того, в HP-28C введены новые для ПМК операции символьных преобразований расчетных формул. Пользователь может также решать задачи в символьном или числовом виде с действительными и комплексными пе-

ременными, составлять, преобразовывать и оценивать значения параметров математических моделей физических объектов, выводить на индикатор графики функций в выбранном масштабе и решать нелинейные уравнения по более совершенным, чем в HP-18C, алгоритмам.

Основная практическая особенность ПМК HP-18C и в значительно большей степени HP-28C — возможность автоматизации решения сложных прикладных задач без предварительного составления на бумаге расчетных соотношений и программ, отображающих алгоритм решения задачи на входном языке. Вместо этого пользователь непосредственно вводит с клавиатуры (с отображением на индикаторе) расчетные формулы с символьным или числовым заданием переменных и операторы, обеспечивающие автоматическое решение типовых математических задач преобразования расчетных формул и получения результатов. Подобная методика решения задач аналогична обычной, только отпадает необходимость записей на бумаге, причем автоматизация решения задач обеспечивается встроенными 240 функциями или операторами, из которых примерно половина обеспечивает символьные преобразования, выполняемые в соответствии с правилами алгебры и математического анализа.

Вводимые с клавиатуры расчетные формулы и операторы, последовательность которых отображает выбранный способ решения задачи, отображается ПМК HP-28C программой на внутреннем языке высокого уровня RPL (от слов ROM-based Procedural Language — процедурный язык на базе ПЗУ). Особенности этого языка программирования совместно с хранимой в ПЗУ операционной системой обеспечивают возможность решения сложных задач при относительно малой емкости ОЗУ 2 Кбайт.

Таким образом, в ПМК HP-18C и особенно HP-28C реализованы новые для ПМК тенденции обеспечения максимальных удобств пользователю, принципиально отличающиеся от традиционного подхода к решению этой задачи, характерного для универсальных ЭВМ других классов, включая и персональные, а также для высокопроизводительных ручных компьютеров, подобных HP-71B. Несомненно, эти тенденции окажут заметное влияние и на особенности ПМК следующих поколений.

6.5. ПЕРСПЕКТИВЫ СОВЕРШЕНСТВОВАНИЯ ПМК

Рассмотренные особенности современных карманных компьютеров свидетельствуют о непрерывном совершенствовании их архитектуры, элементной базы и математического обеспечения. Быстрое развитие микроэлектроники создало предпосылки для непосредственного применения при разработке ПМК наиболее эффективных архитектурных решений, реализованных в универсальных ЭВМ других классов. Однако такое прямое заимствование существенно ограничено требованием минимальной стоимости и, следовательно, минимальных аппаратурных затрат. Поэтому развитие архитектуры ПМК в значительной

мере связано со снижением степени интеграции БИС при переходе к многопроводным шинам, занимающим на поверхности кристалла площадь, пропорциональную квадрату числа проводов.

В первых поколениях ПМК, как правило, использовалась последовательная передача информации по однопроводным линиям, обеспечивавшая минимальные аппаратурные затраты при соответственном снижении скорости передачи информации по сравнению с параллельным кодом. При этом архитектурные решения, принятые при разработке первого карманного компьютера HP-35, в значительной мере сохранились и для последующих поколений ПМК, разработанных как фирмой Hewlett-Packard, так и другими их ведущими изготовителями.

В ПМК HP-35 при последовательной передаче информации был выбран центральный процессор с обработкой в арифметическо-логическом устройстве за один такт содержимого 4 двоичных разрядов, соответствующего одной десятичной или шестнадцатеричной цифре, а также операционным стеком с четырьмя регистрами текущих данных, работающими по правилам обратной польской записи, что уменьшило аппаратурные затраты. Эти особенности центрального процессора сохранились и в последующих ПМК фирмы Hewlett-Packard как с компактными входными языками (например, HP-41C и его модификации), так и в высокопроизводительных ПМК с алгоритмическими входными языками, подобных HP-71B.

Однако последовательная передача данных в связи с ее малым быстродействием, принятая в ПМК первых поколений, была заменена последовательно-параллельной передачей данных с 4-разрядным параллельным кодом. В современных ПМК дальнейшее расширение разрядности параллельного кода является редким исключением, так как многопроводные информационные шины занимают на поверхности кристаллов БИС площадь, пропорциональную квадрату числа проводов, что приводит к значительным аппаратурным затратам. Относительно медленная передача информации при 4-разрядном параллельном коде компенсируется повышением тактовой частоты, развитием интерфейса для максимального уплотнения мультиплексируемой информации, а также совершенствованием архитектуры и программного обеспечения операционной системы, обеспечивающих уменьшение объема потоков информации.

В связи с разнообразием решений, принимаемых разработчиками ПМК как для поиска оптимальной архитектуры, так и для патентной защищенности решений конкурирующих фирм, остановимся лишь на некоторых примерах, иллюстрирующих наиболее перспективные тенденции в развитии архитектуры и программного обеспечения операционных систем.

Значительный интерес, в частности, представляют архитектурные решения, принятые при разработке высокопроизводительного ПМК HP-71B с большим адресным пространством (512 Кбайт или 1 Мнибл), определяемым 20-разрядными адресами. Несмотря на большие потоки

информации между центральным процессором, контроллером дисплея, встроенными БИС памяти и внешними устройствами, системная магистраль этого ПМК содержит лишь 10 проводов, из которых 4 предназначены для передачи информации параллельным 4-разрядным кодом, 2 — для передачи управляющих стробовых сигналов STR и сигналов команд-данных C/D, а 2 — для сигналов управления конфигурацией рабочего адресного пространства (оставшиеся 2 провода предназначены для «заземления» и напряжения питания).

Конфигурация рабочего адресного пространства определяется областями адресов, выделяемых для различных функциональных узлов, участвующих в обмене информацией. Формирование конфигурации рабочего адресного пространства обеспечивается структурой, образованной блоками (портами) замкнутой последовательности устройств, называемой DAISY-цепочкой (название выбрано по аналогии с лепестками цветка daisy — ромашки, маргаритки). Выбор конфигурации определяется передаваемыми по системной магистрали сигналами DAISY—OUT (DO) и DAISY—IN (DI), обеспечивающим управляемое программой формирование конфигурации.

Цепочка DAISY охватывает 6 портов (звеньев) и определяет конфигурацию (включение) или деконфигурацию (исключение) порта в текущем адресном пространстве. Конфигурированному порту соответствуют сигналы DO = DI. Команда RESET деконфигурирует все порты, а команда UNCONFG — устройство с заданным адресом. Деконфигурированное устройство (DO = 0) может реагировать на команды ID с 5-нибловым идентификатором и CONFG с 20-разрядным (5 нибл или тетрад) адресом, в котором 9 старших разрядов содержат адрес устройства, а 11 младших — внутренний фиксированный адрес слова в памяти устройства. При этом внутренние адреса слов в различных устройствах могут совпадать. Соответственно каждое устройство имеет 20-разрядные регистры идентификации и адреса, в которые засылаются идентификатор и адрес команд ID и CONFG. На эти команды реагирует лишь первое деконфигурируемое устройство в DAISY-цепочке, причем после его конфигурации устанавливается сигнал DI = 1, обеспечивающий возможность деконфигурации следующего устройства цепочки. Таким образом, линии DO и DI обеспечивают относительно простым способом конфигурацию различных устройств в текущем адресном пространстве, причем каждое устройство может занимать в этом пространстве область, соответствующую емкости памяти от 8 до 128 Кбайт. Следует добавить, что в конфигурациях адресного пространства ПМК HP-71B ряд областей зарезервирован и объем реального текущего пространства может быть расширен.

Благодаря высокому быстродействию центрального процессора ПМК HP-71B системная магистраль обеспечивает передачу данных со скоростью 1 Мбит/с, хотя реальная скорость передачи данных зависит от режима обмена данными.

Среди других особенностей HP-71B можно отметить способ согласования записи процедур по алгебраическим правилам во входном

языке и выполнения процедур в операционном стеке по правилам обратной польской записи. Для этого использован достаточно сложный итерационный алгоритм декомпиляции строк—операторов входного языка с буферным регистром индикатора, в котором выполняется грамматический разбор при сохранении машинных представлений чисел с 12 десятичными разрядами мантиссы.

Значительный интерес представляют архитектурные и программные особенности ручного компьютера HP-28C в связи с реализацией символьных преобразований и удобной для пользователя диалоговой методики решения прикладных задач. Особенности пользования HP-28C поддерживаются внутренним языком программирования RPL. Программа на этом языке состоит из последовательности близких по структуре блоков, каждый из которых образован адресом исполнительного кода (прологом), определяющим тип объекта, и тела объекта. Тело объекта может быть простым и образованным данными или сложным и состоять из последовательности объектов или их адресов. При этом может указываться адрес объекта, размещенного в этом же или других блоках, а также в ПЗУ, причем большая часть информации, включая имена переменных, хранится в ПЗУ. Такая организация программы на RPL обеспечивает малую ее длину, что позволяет ограничиться ОЗУ емкостью всего 2 Кбайт при емкости ПЗУ 128 Кбайт.

В языке RPL определено всего 17 типов объектов, разбитых на три класса — идентификаторы, данные и процедуры.

Идентификаторы подразделяются на три типа — обычные, временные и указатели ПЗУ. Обычный идентификатор представляет собой имя переменной, которое при исполнении идентификатора заменяется числовым значением переменной. Временный идентификатор при исполнении не заменяется числовым значением и рассматривается как символ переменной. Указатель ПЗУ используется как адрес объекта, расположенного в ПЗУ. При исполнении указателя ПЗУ выполняются операции над соответствующим объектом.

Данные после выполнения операций над ними относятся к исходному типу. К данным относятся действительные и комплексные числа с обычной и повышенной точностью, последовательности символов и шестнадцатеричных цифр, короткие двоичные слова, представляющие целые числа без знака, массивы данных и одномерные массивы однотипных объектов.

Процедуры содержат последовательности машинных команд, интерпретируемых центральным процессором.

Выполнение программы на языке RPL непосредственно связано с работой операционного стека. В отличие от операционных стеков других ПМК с регистрами данных, в HP-28C операционный стек является многоуровневым стеком файлов, длина которых ограничена лишь емкостью области пользователя ОЗУ. При этом содержимое различных уровней стека отображается на матричном ЖКИ в символьной (алгебраической) форме в режиме бегущей строки, содержащей до 23 знаков, а информация — на индентификаторе с апострофами для сим-

вольных выражений, простыми или двойными треугольными скобками для комплексных чисел, прямоугольными скобками для массивов, причем двумерные массивы (матрицы) ограничены двойными прямоугольными скобками (рис. 6.10).

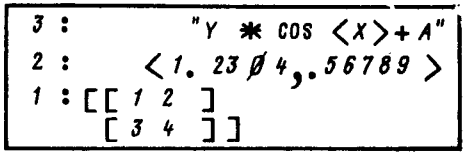


Рис. 6.10

В диалоговом режиме ввода информации с клавиатуры коды символов последовательно засылаются в командную строку, длина которой ограничена лишь емкостью памяти, а индицируемые декомпилированные последовательности кодов отображаются в командной строке указателями — компактными двоичными кодами, описывающими порядок отображения символов на индикаторе. Эти коды сохраняются до тех пор, пока индицируется соответствующая часть содержимого командной строки.

Многие символы, вводимые с клавиатуры, интерпретируются как контекстовые. Например, при вводе символа + вместе с действительными числами выполняется сложение, но при вводе этого же символа в расчетной формуле его код засылается в командную линию и интерпретируется как часть расчетной формулы. В последнем случае сложение выполняется лишь при исполнении соответствующей процедуры над числовыми значениями переменных, определяемых простыми идентификаторами.

Клавиша ENTER, при нажатии которой содержимое операционного стека обычно смещается «вверх», в ПМК HP-28C вводит операцию «выполнить грамматический разбор и ввести числовое значение в командную строку». Расчетные формулы отображаются на индикаторе в алгебраической скобочной записи, а в командной строке компилируются в соответствии с правилами обратной польской записи, обеспечивающей непосредственное выполнение операции после считывания процессором команды выполнения операции. По существу, в языке RPL два типа символьных объектов — имя операнда и имя алгебраической операции. Имя операнда можно рассматривать как выражение, содержащее только простую переменную, тогда как имя операции отображает процедуру, содержащуюся в каталоге или списке процедур ПЗУ. Внешне процедуры RPL аналогичны процедурам в программах на компактных языках с обратной польской записью, но маркируются как алгебраические объекты, подчиняющиеся синтаксическим правилам алгебраических преобразований, таким как приведение подобных членов, подстановки и ряд других.

Язык RPL имеет много общего с языками программирования высокого уровня Форт, Лисп и Аналитик. Общими для языков RPL и Форт являются синтаксические правила обратной польской записи, прохождение операндов и операторов в неограниченном операционном стеке, полное согласование выполняемых операций с работой операционного стека и непосредственное исполнение операции после считы-

вания кода оператора. Общими особенностями языков RPL и Лисп являются одинаковые описания сложных и простых объектов, операции со скобками как над числами, так и над символами, использование временных переменных при определении функций, формирование временных объектов в оперативной памяти.

Близость языка RPL к Аналитику определяется выполнением аналогичных операций над символами и «нагруженностью» программы операционной системой, содержащей списки (каталоги) последовательностей команд, обеспечивающих символьные преобразования. Формально общим для этих языков является возможность диалогового режима, но она реализована различными способами. В Аналитике, как и в Бейсике, диалог реализуется изменением или дополнительным вводом отдельных операторов в заранее составленную и выполняемую программу, тогда как RPL обеспечивает пользователю возможность решения задач по привычной методике с непрерывным изменением и редактированием вводимой информации. Следует добавить, что в языке Аналитик имеется ряд особенностей, не обеспечиваемых языком RPL, которые могут быть использованы в языках символьных преобразований ПМК следующих поколений.

Перспективы развития ПМК в обозримом будущем связаны с основными тенденциями, проявившимися в разработке таких современных ПМК, как HP-71B и HP-28C. Это, с одной стороны, повышение производительности средствами, аналогичными применяемым в универсальных ЭВМ других классов, а с другой — предоставление пользователю максимальных удобств применения ПМК как автономного, удобного и достаточно мощного вычислительного средства.

Особенности ПМК следующих поколений в значительной мере связаны и с совершенствованием как элементной базы встроенных устройств, так и развитием недорогих и достаточно эффективных внешних устройств. В частности, расширение возможностей ПМК связано с созданием достаточно дешевых матричных жидкокристаллических или светодиодных дисплеев с большим числом элементов, обеспечивающих отображение графических изображений с достаточно высокой разрешающей способностью.

ПОСЛЕСЛОВИЕ

Эта книга выйдет в свет, когда большое количество ПМК семейств расширяющегося ряда будет еще находиться в распоряжении пользователей. Это позволяет надеяться, что изложенные в книге сведения не потеряют актуальности. В частности, это относится к описанию особенностей обмена информацией в малосерийных аналогах массовых ПМК семейства «Электроника МК-52» с памятью программ/данных, предназначенного в первую очередь для радиолюбителей и разработчиков цифровой аппаратуры с недорогой и достаточно надежной элементной базой этих ПМК.

Более того, многие архитектурные решения, выбранные в свое время при разработке отечественных ПМК первых поколений, находят прямые или косвенные аналогии в архитектурных особенностях СБИС, используемых в современных ЭВМ и отечественных ПМК новых поколений.

Существование подобных аналогий часто оказывается завуалированным появлением новой терминологии в проектировании ЭВМ, создающей у молодых специалистов ощущение непонимания языка, на котором излагали свои мысли разработчики ЭВМ первых поколений и который в значительной мере сохранен в нашей книге. Однако приходится учитывать, что развитие науки и техники является итерационным процессом, в результате которого часто оказывается, что «новое — хорошо забытое старое» и что научно-технические решения, не реализованные или реализованные на доступном в свое время уровне, могут оказаться весьма плодотворными в изменившихся условиях. Хотя и не следует переоценивать значение полученных ранее результатов, но зачастую пренебрежение ими приводит к потерям средств и времени. Так как это относится в полной мере и к вычислительной технике, то попробуем взглянуть на некоторые из описанных в книге особенностей работы ПМК первых поколений с позиций современного уровня развития архитектуры ЭВМ.

В периодической литературе дискутируются преимущества традиционной и RISC-архитектуры. Появилась и книга *, в составлении которой приняли участие известные зачинатели RISC-направления — Д. Паттерсон и Дж. Хеннеси. RISC-архитектура — это организация обмена информацией в ЭВМ, основанная прежде всего на использовании ограниченного набора команд, выполняемых, как правило, в течение одного такта, из которых при необходимости формируются более сложные команды. К структурным особенностям процессоров с такой архитектурой относят, в частности, конвейер команд и кэш-память — местные ОЗУ для хранения наборов команд с определенной системой адресации.

В развитии ЭВМ высокой производительности появление идеи о RISC-архитектуре сыграло более значительную роль, чем конкретные разработки самых совершенных ЭВМ с традиционной архитектурой, так как эта идея явилась толчком к широкому обсуждению методологии проектирования современных ЭВМ. С нашей точки зрения, одним из важнейших результатов разработки RISC-компьютеров явилась возможность доступа пользователя к «гибкому» микропрограммированию.

В книге при описании синхропрограммирования не подчеркивалось, что при разработке однокристальных микроЭВМ серии К145ИК5 был выдвинут принцип допуска системного программи-

---

* Электроника СБИС. Проектирование микроструктур: Пер. с англ. / Под ред. Н. Айсирука. — М.: Мир, 1989. — 256 с.

ста к уровню микропрограммирования, поскольку допущенными оказались лишь сами разработчики этой серии. Изготовление программ в одном технологическом цикле с аппаратурной частью микрокомпьютера помешало расширить круг программистов, допускаемых к уровню синхропрограмм — программному обеспечению, размещаемому в ПЗУ непосредственно на БИС процессора.

Простота уровня микрокоманд БИС серий К145ИК5 и К145ИК13 отличается от «простоты» набора команд RISC-процессора и его аппаратурной реализации, но принцип однотактности команд, предполагающий возможность построения синхропоследовательностей, необходимых для реализации сложных команд системным программистом или пользователем, расширяет диапазон выбора системы макрокоманд. Это соответствует и аргументации сторонников RISC-архитектуры: благодаря поддержке компиляторами языков высокого уровня RISC-процессор не уступает по эффективности новейшим процессорам с традиционной архитектурой.

Интересно отметить, что по мере увеличения степени интеграции БИС и емкости ОЗУ системы команд стали расширяться — тенденция, характерная для традиционной архитектуры. Однако дальнейшее повышение степени интеграции в СБИС, обеспечившее существенное повышение емкости ОЗУ процессора, привело к упрощению системы команд вместе с появлением конвейера команд и кэш-памяти, характерных для RISC-архитектуры. Вместе с этим повысилась автономность процессорных СБИС и снизилась частота обращений к внешней памяти — для RISC-архитектуры характерны лишь две команды (LOAD и STORE), связанные с этими операциями.

Последовательная архитектура БИС серий К145ИК5 и К145ИК13 предполагала допустимость сокращения пропускной способности канала связи с внешней памятью на БИС К145ИР1 или К145ИР2 при реализации алгоритмов в однокристальных микроЭВМ. Аналогично современные сопроцессоры с плавающей точкой на СБИС при реализации длинных алгоритмов также редко обращаются к внешней памяти. Ширина канала внутренней связи таких сопроцессоров обычно в несколько раз больше ширины канала связи с внешней памятью.

Подобные примеры, число которых можно увеличить, свидетельствуют о том, что многие идеи, реализованные в отечественных ПМК первых поколений и описанные в книге, не потеряли актуальности и сегодня при разработке ПМК новых поколений.

Для использования новых ПМК широким кругом специалистов желательно также предусмотреть возможность ввода прикладных программ на языке ассемблера, что обеспечит возможность полного использования памяти и быстродействия при решении сложных задач. Наконец, для отечественных ПМК новых поколений весьма желательно предусмотреть и возможность ввода прикладных программ также на компактных входных языках ПМК первых

поколений, учитывая большой объем их прикладного обеспечения, накопленного в библиотеках пользователей и описанного в технической литературе.

Техническая реализация возможности ввода прикладных программ на различных входных языках не вызывает принципиальных затруднений, но стоимость производства ПМК при расширении набора входных языков существенно возрастает. Поэтому выбор входных языков для ПМК новых поколений будет являться компромиссом между их универсальностью и стоимостью изготовления, определяющей доступиость широкому кругу пользователей и зависящей от складывающейся экономической конъюнктуры.

СПИСОК ЛИТЕРАТУРЫ

1. **Астанин Л. Ю., Дорский Ю. Д., Костылев Л. А.** Применение программируемых микрокалькуляторов для инженерных и научных расчетов, — Л.: Энергоатомиздат, 1986. — 176 с.
2. **Барановская Г. Г., Любченко И. Н.** Микрокалькуляторы в курсе высшей математики: Практикум. — Киев: Вища школа, 1987. — 288 с.
3. **Бойко А., Чнкоруди Р.** Компьютер в кармане//Наука и жизнь. — 1987. — № 4. — С. 33 — 37.
4. **Даннлов И. Д.** Секреты программируемого микрокалькулятора. — М.: Наука, 1986. — 160 с.
5. **Дьяконов В. П.** Расчет нелинейных и импульсных устройств иа программируемых микрокалькуляторах. — М.: Радио и связь, 1984. — 176 с.
6. **Дьяконов В. П.** Справочник по расчетам на микрокалькуляторах. — М.: Наука, 1985. — 224 с.
7. **Дьяконов В. П.** Программируемые микрокалькуляторы в аппаратуре и технике эксперимента. Обзор//Приборы и техника эксперимента. — 1985. — № 6. — С. 5—18.
8. **Дьяконов В. П.** Справочник по алгоритмам и программам иа языке бейсик для персональных ЭВМ. — М.: Наука, 1987. — 240 с.
9. **Захаров В. П., Польский Ю. М., Ромашко Н. П.** и др. Одиокристальные компьютеры в системах управления. — Киев: Техиіка, 1984. — 94 с.
10. **Захаров В. П., Польский Ю. М., Ромашко Н. П.** Программируемые микрокалькуляторы на БИС серии К145(К745)//Изв. вузов СССР. Радиоэлектроника. — 1985. — № 5. — С. 99—100.
11. **Захаров В. П., Польский Ю. М., Ромашко Н. П.** и др. Программируемый микрокалькулятор «Электроиика МК-52»//Изв. вузов СССР. Радиоэлектроника. — 1986. — № 7. — С. 94—95.
12. **Лемко Л. М., Гладков В. В., Ермаков С. В.** и др. Персоиальный микрокомпьютер «Электроника МК-85»//Микропроцессориые средства и системы. — 1987. — № 4. — С. 10—12.
13. **Лопатнн В. И., Старовойтов Ю. И.** Программирование персональиого микрокомпьютера «Электроника МК-85» с использованием языка Бейсик//Микропроцессориые средства и системы. — 1987. — № 4. — С. 13—15.
14. **Мнкрокалькуляторы** в играх и задачах. — М.: Наука, 1986. — 160 с. — (Сер. Кибериетика: иеограиичеиные возможиости и возможные ограиичеиия).
15. **Романовскнй Т. Б.** Микрокалькулятор в работе, учебе, играх/На латыш. яз. — Рига: Зииатне, 1982. — 274 с.
16. **Романовскнй Т. Б.** Микрокалькуляторы в рассказах и играх. — Рига.: Зииатие, 1984. — 120 с.
17. **Славнн Г.** Программирование на программируемом микрокалькуляторе типа «Электроника Б3-34». — Таллин: Валгус, 1984. — 128 с.

18. **Трохименко Я. К.** Игры с микроЭВМ. — Киев: Техніка; 1986. — 120 с.
19. **Трохименко Я. К.** Программирование микрокалькуляторов «Электроника МК-61» и «Электроника МК-52». — Киев: Техніка, 1987. — 176 с.
20. **Трохименко Я. К., Любич Ф. Д.** Инженерные расчеты на микрокалькуляторах. — Киев: Техніка, 1980. — 328 с.
21. **Трохименко Я. К., Любич Ф. Д.** Радиотехнические расчеты на микрокалькуляторах. — М.: Радио и связь, 1988. — 302 с.
22. **Трохименко Я. К., Любич Ф. Д.** Микрокалькулятор, Ваш ход! — М.: Радио и связь, 1985. — 224 с.
23. **Трохименко Я. К., Любич Ф. Д.** Инженерные расчеты на программируемых микрокалькуляторах. — Киев: Техніка, 1985. — 328 с.
24. **Францевич Л. И.** Обработка результатов биологических экспериментов на микроЭВМ «Электроника БЗ-34». — Киев: Наукова думка, 1979. — 91 с.
25. **Цветков А. Н.** Прикладные программы для микроЭВМ «Электроника БЗ-34». — М.: Финансы и статистика, 1982. — 128 с.
26. **Цветков А. Н., Епанечников В. А.** Прикладные программы для микроЭВМ «Электроника БЗ-34», «Электроника МК-54», «Электроника МК-56». — М.: Финансы и статистика, 1984. — 176 с.
27. **Цимринг Ш. Е.** Специальные функции. Программы для микрокалькулятора «Электроника БЗ-21». — М.: Радио и связь, 1983. — 120 с.
28. **Wechsler S. L.** A New Handheld Computer for Technical Professionals// Hewlett-Packard J.— 1984. — N 7. P. 3—10.
29. **Wicks W. C.** An Evolutionary RPN Calculator for Technical Professionals// Hewlett-Packard J. — 1987. — N 8. — P. 3—14.