

**А. Л. ГУРТОВЦЕВ С. В. ГУДЫМЕНКО**

**ПРОГРАММЫ  
для  
МИКРОПРОЦЕССОРОВ**



А. Л. ГУРТОВЦЕВ С. В. ГУДЫМЕНКО

# ПРОГРАММЫ для МИКРОПРОЦЕССОРОВ

СПРАВОЧНОЕ ПОСОБИЕ

МИНСК  
«ВЫШЭЙШАЯ ШКОЛА»  
1989

Рецензенты: зав. кафедрой микропроцессорной техники и программирования Ленинградского филиала Института повышения квалификации руководящих работников и специалистов канд. техн. наук *В. М. Кисельников* и начальник сектора НИИ ВЭФ Рижского ПО ВЭФ канд. техн. наук *В. П. Калниньш*

**Гуртовцев А. Л., Гудыменко С. В.**

Г95 Программы для микропроцессоров: Справ. пособие.— Мн.: Выш. шк., 1989.— 352 с.: ил.  
ISBN 5-339-00216-0.

Содержит комплекс типичных прикладных и системных программ для микропроцессорных систем. Рассматриваются задачи арифметической обработки чисел с фиксированной и плавающей запятой, преобразования представлений данных, вычисления значений элементарных функций, обработки структур данных, обмена информацией с внешними устройствами и диалога монитормого типа. Даются алгоритмы и методы программирования этих задач.

Для инженеров, программистов и студентов вузов.

2404090000—089  
Г—————77—89  
М304(03)—89

**ББК 32.973.2я2**

Справочное издание

**Гуртовцев** Аркадий Лазаревич,  
**Гудыменко** Сергей Викторович

## **ПРОГРАММЫ ДЛЯ МИКРОПРОЦЕССОРОВ**

Зав. редакцией *А. Ф. Зиновьев*. Редактор *С. Ю. Липец*. Мл. редактор *С. А. Когадеева*. Оформление и художественное редактирование *Ю. С. Сергачева*. Технический редактор *Г. М. Романчук*. Корректор *Л. А. Еркович*.  
ИБ № 2843

Сдано в набор 18.10.88. Подписано в печать 18.08.89. АТ 10409. Формат 84 × 108/32. Бумага книжно-журнальная. Гарнитура литературная. Высокая печать. Усл. печ. л. 18,48. Усл. кр.-отт. 18,48. Уч.-изд. л. 19,75. Тираж 33 500 экз. Зак. 1926. Цена 1 р. 30 к. Издательство «Вышэйшая школа» Государственного комитета БССР по делам издательств, полиграфии и книжной торговли. 220048, Минск, проспект Машерова, 11.

Минский ордена Трудового Красного Знамени полиграфкомбинат МППО им. Я. Коласа. 220005, Минск, ул. Красная, 23.

ISBN 5-339-00216-0

© Издательство «Вышэйшая школа», 1989

## ПРЕДИСЛОВИЕ

Появление в 1971 г. первого микропроцессора открыло новую эру в развитии вычислительной техники и ее использовании для автоматизации физического и интеллектуального труда. Современный этап развития этой техники характеризуется массовым внедрением микропроцессоров и микроЭВМ в самые различные области человеческой деятельности. Расширяется круг инженеров, которым приходится изучать, осваивать и творчески применять микропроцессорные средства в прикладных разработках. За последние 10 лет в стране издано в помощь инженерам около сотни книг и брошюр по микропроцессорной тематике, однако до последнего времени существует дефицит справочных книг по программированию микропроцессорных систем. Восполнить этот пробел в определенной мере призвано данное пособие, содержащее большой комплекс тщательно документированных прикладных и системных программ реальной сложности: более 150 типичных подпрограмм общим объемом около 7000 строк на языке макроассемблера микроЭВМ, построенных на базе популярной серии КР580.

Первые четыре главы книги посвящены вопросам числовой обработки данных. В первой главе рассмотрены алгоритмы, методы и программы, реализующие арифметические операции над числами с фиксированной запятой, представленными в различных форматах в двух системах счисления: двоичной и двоично-десятичной. Во второй главе описаны аналогичные операции над числами с плавающей запятой. Материал этой главы нетрадиционен для литературы, посвященной микропроцессорной тематике. Третья глава содержит алгоритмы и программы преобразования различных форм представления данных, в том числе двоичных и десятичных чисел с плавающей запятой. В четвертой главе рассматриваются некоторые алгоритмы и программы вычисления значений элементарных функций и факториала. Эти программы наглядно

демонстрируют методологию преодоления сложности за счет иерархической организации достаточно громоздкой числовой обработки. В перечисленных главах много внимания уделяется вопросам точности вычислений и оптимизации программ по противоречивым критериям «быстродействие — объем памяти». Анализ альтернативных программных решений способствует, по мнению авторов, выработке у читателя правильного понимания стратегии проектирования программ и формированию навыка улучшения создаваемых систем. Главы построены по единому образному плану в соответствии с иерархическим принципом «сверху вниз»: от классов задач и методов их решения к конкретным алгоритмам и программам. Программы сопровождаются тестовыми наборами данных, позволяющими читателю оперативно проверить работоспособность этих программ на микроЭВМ.

Решению типовых задач нечисловой обработки данных посвящена пятая глава. Приведенные программы реализуют функции генерации, перемещения, поиска, сравнения и преобразования простых структур данных (массивов, таблиц, списков). Эти программы актуальны для микропроцессорных систем, обрабатывающих большие массивы информации. В шестой главе рассматриваются элементы системного программного обеспечения для организации физического и логического ввода-вывода и диалога монитормого типа.

В приложениях описаны архитектура и система команд микропроцессора КР580, представлены справочные таблицы, обеспечивающие разработку тестовых примеров для программ числовой обработки данных, сведения о языке макроассемблера и программировании периферийных параллельного и последовательного адаптеров. Приложения делают справочное пособие автономным от других источников и помогают читателю разобраться в текстах программ, даже если он ранее не изучал языки программирования.

Книга ориентирована на инженеров — разработчиков микропроцессорных систем, студентов инженерных специальностей и квалифицированных радиолюбителей, увлекающихся конструированием домашних микроЭВМ.

Авторы считают своим приятным долгом выразить благодарность рецензентам — зав. кафедрой микропроцессорной техники и программирования Ленинградского филиала Института повышения квалификации руково-

дящих работников и специалистов канд. техн. наук В. М. Кисельникову и начальнику сектора НИИ ВЭФ Рижского ПО ВЭФ канд. техн. наук В. П. Калниньшу за рекомендации и замечания, учет которых в процессе доработки рукописи способствовал улучшению ее стиля и содержания.

Программы, приведенные в книге, тщательно отлажены и тестированы, тем не менее авторы отдают себе отчет в том, что такой большой объем программного обеспечения вряд ли не содержит ни одной ошибки. Они будут признательны читателям за конструктивные замечания и предложения, которые просят направлять по адресу: 220048, Минск, проспект Машерова, 11, издательство «Вышэйшая школа».

*Авторы*

## УСЛОВНЫЕ СОКРАЩЕНИЯ

АЛУ	— арифметическо-логическое устройство
АОТ	— арифметика ограниченной точности
АЦПУ	— алфавитно-цифровое печатающее устройство
БИС	— большая интегральная схема
ВК	— возврат каретки
ВМ	— выбор микросхемы
ВСИН	— вид синхронизации
ВУ	— внешнее устройство
ГИ	— готовность источника
ГМД	— гибкие магнитные диски
ГП, ГПР	— готовность приемника
ГПД	— готовность передатчика
ДЛ	— делитель
ДМ	— делимое
ДПД	— данные передатчика
ДПР	— данные приемника
ДПУ	— данные периферийного устройства
ЗП	— запись
ЗПД	— запрос передатчика
ЗПР	— запрос приемника, запрос прерывания
ЗУПВ	— запоминающее устройство с произвольной выборкой
ИРПР	— интерфейс радиальный параллельный
ИРПС	— интерфейс радиальный последовательный
КОИ-7	— код обмена информацией семибитный
КПД	— конец передачи
ЛАТ	— латинский регистр
МАН	— мантисса
МЛБ	— младший байт
МЛЦ	— младшая цифра
ММ	— множимое
МН	— множитель
МП	— микропроцессор
МТК-2	— международный телеграфный код
НГМД	— накопитель на гибких магнитных дисках
ОД	— обработчик директив
ОЗУ	— оперативное запоминающее устройство
ОСТ	— остаток
ПА	— порт А
ПВ	— порт В
ПЗУ	— постоянное запоминающее устройство
ПОР	— порядок
ППА	— программируемый параллельный адаптер
ППЗУ	— перепрограммируемое постоянное запоминающее устройство

ПР	— произведение
ПС	— порт С, перевод строки
ПСМ	— полупорт С младший
ПСС	— полупорт С старший
ПУ	— периферийное устройство
РД	— регистр данных
РК	— регистр команд
РОН	— регистр общего назначения
РПД	— регистр передатчика
РПР	— регистр приемника
РР	— регистр режима
РС	— регистр состояния
РСС	— регистр синхросимвола
РУ	— регистр управления
РУС	— регистр управляющего слова, русский регистр
СБР	— сброс
СИ	— синхронизирующие импульсы
СИН	— синхронизация
СЛ	— слагаемое
СМ	— сомножитель
СОЗУ	— сверхоперативное запоминающее устройство
СПД	— синхронизация передатчика
СПР	— синхронизация приемника
СРБ	— средний байт
СРЦ	— средняя цифра
СС	— синхросимвол
СТБ	— старший байт
СТЦ	— старшая цифра
СЧП	— сумма частичных произведений
УС	— управляющее слово, устройство сопряжения
УСАПП	— универсальный синхронно-асинхронный приемопередатчик
УСК	— управляющее слово команды
УСР	— управляющее слово режима
УУ	— устройство управления
ЦИФ	— цифровой регистр
ЧП	— частичное произведение
ЧСТ	— частное
ЧТ	— чтение
ША	— шина адреса
ШД	— шина данных
ШУ	— шина управления



## МЕТОДОЛОГИЧЕСКИЕ ЗАМЕЧАНИЯ

Микропроцессоры используются для построения прикладных систем двух видов: универсальных, в том числе персональных, микроЭВМ с развитым стандартным программным обеспечением и микропроцессорных систем, специализированных по техническим и программным средствам под класс решаемых задач и особые условия эксплуатации. В обоих случаях качество и эффективность проектируемых систем во многом определяются успешным решением сложных и трудоемких задач разработки системного и прикладного программного обеспечения. Спрос на профессиональных программистов постоянно превышает предложение, и вследствие этого инженеры — разработчики систем вынуждены становиться программистами и овладевать методологией создания комплексов программ реальной сложности.

Использование в прикладных системах серийных микроЭВМ существенно облегчает задачу проектирования программного обеспечения этих систем за счет готовых стандартных программных средств микромашин. Однако при создании многих микропроцессорных систем инженер лишен такой возможности из-за отсутствия подходящих микроЭВМ и вследствие различных технико-экономических ограничений, предъявляемых к системам, например по надежности, быстродействию, энергопотреблению, стоимости и другим параметрам. В этих условиях инженер вынужден проектировать аппаратные и программные средства системы «с нуля».

Работы нулевого цикла обычно начинаются с поиска аналогов. Когда разработчик решает вопросы схмотехнического построения микропроцессорной системы, в его распоряжении имеется богатый опыт проектирования, отраженный в литературе, насчитывающей сотни наименований (справочники, каталоги, монографии, статьи, описания патентов и авторских свидетельств). Совсем иная ситуация складывается, когда он приступает к проектированию программного обеспечения. Литература по программированию микропроцессорных систем неизмеримо беднее схмотехнической литературы и дает мало готовых программных аналогов, которые разработчик мог бы использовать при создании своей системы (пожалуй, исключением является монография [44]). Общие рекомендации и фрагменты программ, приводимые в литературе, полезны на первой стадии знакомства с программированием, но не могут удовлетворить тех, кто должен разрабатывать достаточно сложные комплексы программ.

Доступ к готовому фирменному и ведомственному программному обеспечению, к программам, сданным в фонды, затруднен, требует больших затрат времени и оказывается чаще всего напрасным, поскольку качество сопровождения программ оставляет желать лучшего, а усилия, потраченные на расшифровку текстов, невосполнимы. Точную характеристику этой проблемы дал академик А. П. Ершов: «Наши программисты в целом по-прежнему находятся в плену импортного

программного продукта, т. е. либо играют с ним в режиме черного ящика, либо теряют физическое и душевное здоровье, копаясь в исходных кодах в худших традициях реверсной технологии. Разработчики же оригинальных программ приземляют свою работу, либо зарабатывая деньги родному вузу, либо находясь в искусственной изоляции в силу ведомственной разобщенности и неразвитости системы научно-технической коммуникации. В результате общезначимость программистской работы недопустимо низка» [29].

Разработчику часто приходится тратить время и усилия на создание программ, которые где-то кем-то уже созданы, но не описаны в доступной форме в литературе. Данная книга призвана частично восполнить этот пробел. В ней систематизированы алгоритмы и методы элементарной числовой и символической обработки информации для решений широкого спектра типичных задач прикладного и системного программирования микропроцессорных систем, а также представлен большой комплекс программ, практически реализующий эти решения. Большинство приведенных программ оригинальны и отражают многолетний опыт авторов по разработке и программированию различных микропроцессорных систем. Хотя представленные программы разработаны, отлажены и оформлены специально для данного пособия, они в полной мере соответствуют реальным сложным программам и методологии преодоления сложности.

Читатель может задать закономерный вопрос: «Почему пособие носит название «Программы для микропроцессоров», хотя в нем приводятся тексты программ только на языке ассемблера для микропроцессора серии КР580?» Ведь язык ассемблера является низкоуровневым, машинно-зависимым языком, т. е. программы, написанные на языке ассемблера микропроцессора одного типа, не могут быть использованы без изменений для другого типа микропроцессора.

Выбор авторами языка ассемблера объясняется тем, что он позволяет наиболее детально, почти на аппаратном уровне раскрыть механизм решения задачи и реализовать его эффективно — с минимальными затратами емкости памяти и времени микропроцессора, что актуально для многих микропроцессорных систем, особенно реального времени. Этот язык в силу его близости к структуре микропроцессора позволяет инженеру наиболее безболезненно, логично и быстро перейти от схемотехники к программированию и накоплению нового опыта. Вместе с тем трудоемкость разработки программ на языках ассемблера значительно превышает трудоемкость разработки аналогичных программ на языках высокого уровня и потому в первую очередь требует облегчения за счет использования готовых программ-аналогов. Выбор конкретного микропроцессора серии КР580 обусловлен широким распространением и доступностью этой серии большому кругу пользователей, причем вопросы схемотехнического построения систем на ее базе достаточно подробно освещены в литературе. Кроме того, данная серия используется во многих отечественных микроконтроллерах и микроЭВМ типа, например, СМ 1800. «Электроника К1-10», «Электроника К1-20», СО-4, В7, которые можно использовать в качестве программных отладочных комплексов.

Программы на языке ассемблера для микропроцессора КР580, приведенные в книге, имеют двойное назначение. Во-первых, они выступают как образцы, которые читатель может непосредственно вставлять в свой комплекс программ «с листа книги». В этом качестве программы могут, естественно, использоваться только для выбранного типа микропроцессора. Они размещены в непересекающихся областях

памяти и оформлены в виде ассемблерных листингов, содержащих адреса и объектные коды, что позволяет читателю выполнить быстрый ввод объектных кодов и тестирование программ мониторными средствами микропроцессорной системы. Во-вторых, программы можно использовать как анлоги, детально объясняющие конкретные реализации приведенных типовых алгоритмов решения задач. В этом качестве программы применимы для многих микропроцессоров: известно, что написать программу, имея алгоритм и понятную программу-аналог, гораздо проще, чем «начинать с нуля».

Пожалуй, главная проблема практического программирования — использование «чужого» программного обеспечения. Хотя программы пишутся для машин, в первую очередь они должны быть понятны людям: «чужое» может стать «своим» лишь в том случае, если оно понятно. Чужие программы, оторванные от машинной среды и представленные в виде «черного ящика», вызывают естественное недоверие, которое сдерживает их применение в проектах. Это недоверие можно устранить либо после длительного тестирования программы, либо после полного уяснения алгоритма работы и деталей ее внутренней организации. Максимальная читаемость текста программы, ее компактность и обзорность, доступность для быстрого понимания «чужим умом» являются необходимыми условиями использования и «живучести» чужих программ. Эти требования становятся еще актуальнее в том случае, когда речь идет о модернизации чужой программы. Без понимания структуры программы любые изменения в ней «на авось» чреваты ошибками.

Возможность понимания программы обеспечивается ее компактным построением и многоуровневым описанием. Самый высокий уровень описания программы — это описание в виде «черного ящика» с расшифровкой структуры его входов-выходов и выполняемой функции. Такое описание помогает восприятию программы как целостного элемента и необходимо для начального знакомства с ней. Этим описанием можно ограничиться в случае полного доверия к программе. Самый низкий и самый детальный уровень описания программы — описание на языке программирования (примитивный уровень объектных кодов не рассматриваем). Восприятие программы на этом уровне требует больших интеллектуальных усилий в связи с необходимостью оперативного запоминания и мысленной обработки большой последовательности взаимосвязанных и специфических элементов искусственного языка (операторов или команд). Такое описание необходимо для построения самой программы.

Для понимания программы решающее значение имеет третий, промежуточный уровень описания, соответствующий уровню алгоритма. Обычно этот уровень представляют в виде схемы алгоритма на отдельном документе. Более эффективный прием заключается в объединении трех уровней описания внутри текста программы. При этом промежуточный уровень реализуется за счет тщательно продуманных и лаконичных строк — предложений на родном языке, которые подразделяют весь текст программы на функционально законченные части — разделы и комментируют функции этих разделов. При таком подходе программа может читаться на любом уровне в зависимости от целей чтения. Этот прием использован во всех программах данного пособия.

Компактность и обзорность программ достигаются за счет их модульного построения в виде структуры взаимодействующих подпрограмм. При этом сложные программы формируются из менее сложных модулей-подпрограмм и в свою очередь образуют готовые модули-

подпрограммы для построения более сложных программ. Такая иерархическая, многоуровневая организация комплекса программ, хотя и ведет к дополнительным затратам машинного времени, связанным с вызовом подпрограмм, является на практике основным средством борьбы со сложностью решаемых проблем. Если перед читателем стоит задача использовать лишь отдельные программы из комплекса, приведенного в книге, и при этом желательно минимизировать время их выполнения, в выбранной программе вызовы подпрограмм он должен заменить их телами. При этом возможно упрощение результирующей программы за счет исключения механизма передачи параметров между подпрограммами.

В заключение сделаем еще одно замечание. Каждая программа имеет свое собственное имя, по которому производится обращение к ней, а в текстах программ используются имена-метки, идентифицирующие отдельные программные фрагменты и переходы к ним. Для достижения обзорности и понимания программ, особенно при их иерархическом построении, большое значение имеет смысловая нагрузка этих имен. Целесообразно для имен использовать продуманную аббревиатуру на родном языке, раскрывающую основную функцию программы или ее фрагмента. Затраты времени, связанные с переходом с латинского регистра на русский при редактировании такого смешанного русско-английского текста, с лихвой окупаются при эксплуатации программ.

# 1. ПРОГРАММЫ АРИФМЕТИКИ С ФИКСИРОВАННОЙ ЗАПЯТОЙ

## 1.1. ОБЩИЕ СВЕДЕНИЯ

В данной главе рассматриваются различные форматы представления двоичных и десятичных, целых и дробных, знаковых и беззнаковых чисел, а также методы, алгоритмы и программы выполнения арифметических операций над множеством этих чисел, обеспечивающие заданную точность и диапазон вычислений при определенных затратах памяти и времени работы микропроцессора.

Изображение чисел в любой *позиционной системе счисления* с натуральным основанием  $R (R > 1)$  базируется на представлении их в виде произведения целочисленной степени  $R^m$  основания  $R$  на *полином* от этого основания [45, 53, 54, 61]:

$$A_R = \pm R^m \sum_{i=1}^n a_i R^{-i}, \quad (1.1)$$

где  $a_i \in \{0, 1, \dots, R-1\}$  — цифры  $R$ -ичной системы счисления (« $R$ -ичные цифры»);  $n$  — количество разрядов (*разрядность*), используемых для представления числа;  $R^m$  — *характеристика числа*, причем показатель  $m \in \{\dots, -2, -1, 0, +1, +2, \dots\}$ ;  $R^m R^{-i} = R^{m-i}$  — *позиционный вес*  $i$ -го разряда числа. В *десятичной* ( $R = 10$ ) системе для представления чисел используются цифры  $a_i = (0, 1, \dots, 9)$ , в *двоичной* ( $R = 2$ ) — цифры  $a_i = (0, 1)$ , в *шестнадцатеричной* ( $R = 16$ ) — цифры  $a_i = (0, 1, \dots, 9, A, B, C, D, E, F)$ , где прописным латинским буквам  $A, \dots, F$  эквивалентны соответственно числа 10, ..., 15 в десятичной системе.

Если  $m = \text{const}$ , то формула (1.1) определяет представление числа  $A_R$  в форме с *фиксированной запятой* (наряду с этим термином в вычислительной математике и технике широко используется термин-синоним «фиксированная точка»). Позиция, в которой запятая фиксируется между разрядами числа, отделяя *целую часть* от

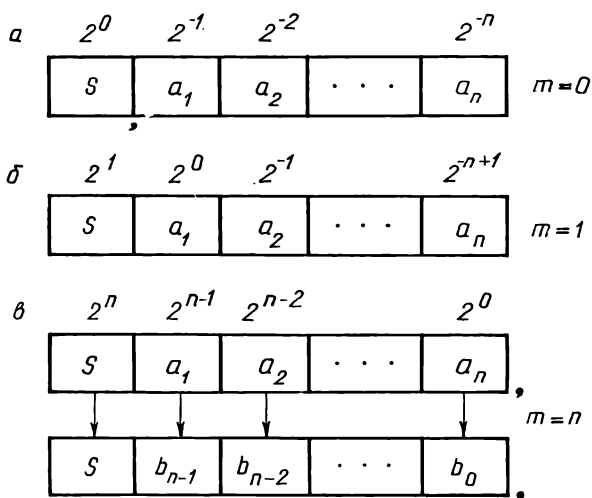


Рис. 1.1. Форматы двоичных чисел с фиксированной запятой:

*a* — дробные числа; *б* — смешанные числа; *в* — целые числа

дробной и определяя вес соответствующих разрядов, постоянна в процессе вычислений для всего множества используемых чисел и зависит от заранее установленного значения.  $m$ . Если  $m \leq 0$ , то формула (1.1) представляет *дробные числа* (правильные дроби); если  $m \geq n$ , — *целые числа*; если  $0 < m < n$ , — *смешанные числа* (неправильные дроби). Обычно для чисел с фиксированной запятой значение  $m$  ограничено  $0 \leq m \leq n$ , т. е. позиция запятой в числе выбирается в рамках  $n$  разрядов, отведенных для изображения цифровой части числа.

На рис. 1.1 приведены форматы двоичных ( $R=2$ ) чисел с фиксированной запятой для случаев  $m=0$  (*a*),  $m=1$  (*б*) и  $m=n$  (*в*). Далее ограничимся рассмотрением только целых и дробных чисел, поскольку действия над смешанными числами могут быть сведены к отдельным операциям над их целыми и дробными частями. Дробные числа в отличие от целых будем рассматривать как числа с фиксированной запятой в узком смысле этого термина и для их обозначения в формулах и программах использовать букву «ф».

В микропроцессорах основой представления чисел является двоичная система счисления. Рассмотрим пред-

ставление дробных и целых *двоичных чисел*. При  $R = 2$ ,  $m = 0$  формула (1.1) имеет вид

$$A_{2\Phi} = \pm \sum_{i=1}^n a_i \cdot 2^{-i} = \pm a_1 a_2 \dots a_n, \quad (1.2)$$

где  $2^{-i}$  — вес  $i$ -го двоичного разряда;  $a_i \in \{0, 1\}$  — двоичные цифры.

Формула (1.2) определяет *дробные двоичные числа со знаком*. Левая ее часть дает *развернутую форму записи* числа в виде полинома, а правая — *свернутую форму записи* в виде последовательности цифр (коэффициентов полинома). Для изображения *знака числа*  $S$  (от лат. Signum — знак) отводится дополнительный *знаковый разряд* (нулевой — с весом  $2^0$ ), а запятая фиксируется перед старшим (первым — с весом  $2^{-1}$ ) *цифровым разрядом* числа (см. рис. 1.1, а). Если заранее известно, что множество используемых чисел не содержит отрицательных, в формате числа разряд знака можно опустить. Такие числа называют *беззнаковыми*. В  $n$ -разрядной сетке *дробных беззнаковых чисел* может быть точно представлено  $2^n$  различных, в том числе  $2^n - 1$  ненулевых, чисел, удовлетворяющих неравенству

$$A_{2\Phi \min} = 2^{-n} \leq A_{2\Phi} \leq A_{2\Phi \max} = 1 - 2^{-n}, \quad (1.3)$$

где  $A_{2\Phi \min}$  — *минимальное* (ненулевое), а  $A_{2\Phi \max}$  — *максимальное* из представимых чисел. В свернутой форме записи эти числа изображаются в виде  $A_{2\Phi \min} = , 00 \dots 01$  и  $A_{2\Phi \max} = , 11 \dots 11$ .

В случае  $R = 2$ ,  $m = n$  формула (1.1) преобразуется к виду

$$A_2 = \pm \sum_{i=1}^n a_i \cdot 2^{n-i} = \pm \sum_{i=0}^{n-1} b_i \cdot 2^i = \pm b_{n-1} \dots b_1 b_0, \quad (1.4)$$

где  $b_i = a_{n-i}$  для всех  $i = 0, 1, \dots, n - 1$ ;  $a_i, b_i \in \{0, 1\}$  — двоичные цифры. Перекодировка индексов в формуле (1.4) дает удобочитаемое соответствие веса разряда  $2^i$  индексу его цифры. Формула (1.4) определяет *целые двоичные числа со знаком*. Под знак отводится дополнительный  $(n + 1)$ -й разряд с весом  $2^n$ , а запятая фиксируется после младшего (нулевого — с весом  $2^0$ ) цифрового разряда числа (см. рис. 1.1, в). Для *беззнаковых целых чисел*

разряд знака  $S$  опускается. В  $n$ -разрядной сетке целых беззнаковых чисел, как и для дробных, может быть представлено множество  $2^n$  целых, в том числе  $2^n - 1$  ненулевых, чисел, удовлетворяющих неравенству

$$A_{2\min} = 2^0 = 1 \leq A_2 \leq A_{2\max} = 2^n - 1. \quad (1.5)$$

В свернутой форме эти числа изображаются такими же последовательностями цифр, как и дробные числа, отличаясь лишь положением запятой:  $A_{2\min} = 00...01$ , и  $A_{2\max} = 11...11$ .

В формулах (1.2) — (1.5) двоичные числа представлены в виде полинома, коэффициенты которого выражены двоичными цифрами, а основание — десятичным числом, т. е. цифрой 2. Учитывая, что в любой позиционной системе счисления ее основание однозначно представляется с помощью цифр этой системы в виде  $10_R$  (по определению,  $R = (R - 1) + 1$ , а такая операция дает нуль в младшем разряде и единицу переноса в старший разряд), формулы (1.2) — (1.5) можно обобщить на случай любой позиционной системы:

$$A_{R\Phi} = \pm \sum_{i=1}^n a_i \cdot 10_R^{-i}, \quad 10_R^{-n} \leq |A_{R\Phi}| \leq 1 - 10_R^{-n}; \quad (1.6)$$

$$A_R = \pm \sum_{i=0}^{n-1} a_i \cdot 10_R^i, \quad 1 \leq |A_R| \leq 10_R^n - 1, \quad (1.7)$$

где обозначение модуля введено для знаковых чисел, а неравенства справедливы для всех ненулевых значений представляемых чисел. Формула (1.6) определяет свойства дробных, а (1.7) — целых чисел с основанием  $10_R$ . Эти формулы полезны, в частности, при рассмотрении чисел в десятичной и шестнадцатеричной системах счисления, широко используемых в микропроцессорах наряду с двоичной системой.

Для представления в микропроцессорах систем счисления с основанием  $R = 10$  и  $R = 16$  используются *смешанные системы счисления*, в которых каждая цифра  $R$ -ичной системы изображается цифрами другой,  $Q$ -ичной ( $Q < R$ ), в частности двоичной, системы [47, 53, 61, 75]. Такая смешанная система называется  $Q - R$ -ичной. Для записи  $R$ -ичной цифры отводится одно и то же количество  $Q$ -ичных разрядов, минимально необходимое для пред-



ставления любой  $R$ -ичной цифры. В *двоично-десятичной системе* каждая десятичная цифра кодируется тетрадой двоичных цифр, причем между десятью (из возможных  $2^4 = 16$ ) двоичными числами и десятичными цифрами устанавливается взаимно однозначное соответствие. Возможны различные варианты установления этого соответствия, порождающие различные *двоично-десятичные коды* с теми или иными свойствами [17, 53]. В программах данного пособия используется один из наиболее распространенных кодов — код с естественными весами 8421, в котором десятичные цифры кодируются их естественными двоичными эквивалентами: (0,0000), (1,0001), ..., (9,1001). Этот код преимущественно применяется для ввода-вывода данных (при преобразованиях из двоичной системы в двоично-десятичную и наоборот), но может использоваться и непосредственно при обработке данных.

В частном случае смешанных систем, когда  $R = Q^k$  ( $k$  — целое), например  $R = 16$ ,  $Q = 2$  ( $k = 4$ ), запись числа в  $R$ -ичной системе является сокращенной записью представления этого числа в  $Q$ -ичной системе: цифра шестнадцатеричной системы заменяет значение четырех цифр двоичной системы, что упрощает описание представлений двоичных чисел. Эта смешанная система широко используется в программах, приведенных в данной книге.

Известно, что арифметические операции над числами в любой позиционной системе счисления выполняются по тем же правилам, что и в десятичной арифметике, поскольку все они основываются на общих правилах выполнения операций над соответствующими полиномами [47, 54]. При этом используются те таблицы сложения (вычитания) и умножения, которые имеют место в конкретной системе счисления. В частности, для двоичной системы действуют правила:

$$\begin{array}{lll} 0 + 0 = 0 & 0 - 0 = 0 & 0 \times 0 = 0 \\ 0 + 1 = 1 & 1 - 0 = 1 & 0 \times 1 = 0 \\ 1 + 0 = 1 & 1 - 1 = 0 & 1 \times 0 = 0 \\ 1 + 1 = 10 & 10 - 1 = 1 & 1 \times 1 = 1 \end{array}$$

Эти правила не содержат операции деления двоичных цифр, поскольку ее выполнение в отличие от других операций не может быть сведено к действиям над отдельными цифрами. Заметим, что при сложении в двоичном разряде

двух единиц происходит *перенос* из данного разряда в следующий, более старший разряд, а при вычитании в двоичном разряде единицы из нуля происходит *заем* в данный разряд из более старшего разряда, в результате в данном разряде устанавливается единица.

При выполнении арифметических операций возникает проблема идентификации отрицательных чисел. Как отмечалось выше, для изображения знака двоичных знаковых чисел отводится специальный знаковый разряд  $S$ . Изображение знака «+» в этом разряде принято кодировать для двоичных чисел цифрой 0, а знака «—» — цифрой 1. При этом изображение числа со знаком содержит только двоичные цифры, но само число подразумевается состоящим из двух частей: *знаковой* и *цифровой*.

Если цифровая часть положительных и отрицательных чисел содержит всегда абсолютную величину числа, то такой способ представления знаковых чисел называют *прямым кодом*. Обработка чисел, представленных в прямом коде, требует отдельных операций над цифровой и знаковой частями, альтернативного выполнения операций сложения и вычитания, приводит к появлению двух *представлений нуля*:  $+0$  и  $-0$  (например, для дробных чисел  $+0 = 0,00...00$ ;  $-0 = 1,00...00$ ). Эти недостатки прямого кода сдерживают его использование для обработки данных, и он находит применение преимущественно в операциях ввода-вывода данных (хотя известны примеры использования его в микроЭВМ для основного представления знаковых чисел [39]). В микропроцессорах для обработки знаковых чисел используются в основном *дополнительные коды* (в качестве промежуточных — также и *обратные коды*) [17, 52, 63, 75].

Рассмотрим определения и свойства этих кодов. Пусть *знаковое число с фиксированной запятой* содержит  $n + 1$   $R$ -ичных разрядов, причем крайний левый разряд в формате числа — разряд знака, в котором «+» закодирован цифрой 0, а «—» — цифрой  $R - 1$  (1 — для двоичной, 9 — для десятичной, F — для шестнадцатеричной системы счисления). Тогда дополнительный код числа определяется выражением

$$[A_R]_д = \begin{cases} A_R, & \text{если } A_R \geq 0; \\ \Gamma + A_R, & \text{если } A_R < 0, \end{cases} \quad (1.8)$$

где  $\Gamma$  — *граница числа*, причем для дробных чисел  $\Gamma = 10_k$  ( $\Gamma = 2$  для  $R = 2$ ,  $\Gamma = 10$  для  $R = 10$  и  $\Gamma = 16$  для

$R = 16$ ), а для целых чисел  $\Gamma = 10_r^{n+1}$  ( $\Gamma = 2^{n+1}$  для  $R = 2$ ,  $\Gamma = 10_r^{n+1}$  для  $R = 10$  и  $\Gamma = 16^{n+1}$  для  $R = 16$ ). Иными словами, в дополнительном коде запись положительного числа идентична его записи в прямом коде, а запись отрицательного числа представляет собой результат операции вычитания модуля числа  $A_R$  из границы:  $\Gamma - |A_R|$ .

Поскольку граница  $\Gamma$  представляет собой фиксированную степень, легко показать, что указанная операция вычитания сводится к поразрядной операции дополнения каждой цифры числа до старшей цифры системы счисления (поиску взаимно обратной цифры  $\bar{a}_i$ , такой, что  $a_i + \bar{a}_i = R - 1$ ) и суммированию полученного обратного кода с единицей младшего разряда числа, т. е. с величиной  $10_r^{-n}$  для дробных и  $10_r^0 = 1$  — для целых чисел. Например, для десятичной дроби 0,1234 ее отрицательный эквивалент в дополнительном коде имеет вид  $9,8765 + 0,0001 = 9,8766$ . Для проверки правильности преобразования сложим эти числа:  $0,1234 + 9,8766 = 10,0000$ . Результат равен границе десятичной дроби. Поскольку формат чисел содержит 5 разрядов (знаковый и 4 цифровых), полученная при сложении чисел единица переноса выходит за рамки формата, и истинный результат равен 0,0000, что подтверждает правильность преобразования отрицательного числа в дополнительный код. Аналогичные действия используются для обратного перевода дополнительного кода отрицательного числа в прямой код, т. е. для получения модуля числа.

Смысл использования дополнительного кода заключается в том, что, во-первых, арифметические операции вычитания и сложения чисел в дополнительных кодах сводятся к операции алгебраического суммирования (вычитание заменяется сложением уменьшаемого с дополнительным кодом вычитаемого); во-вторых, обработка знаковой и цифровой частей чисел при сложении производится по одним и тем же правилам, причем правильный знак результата формируется автоматически. Заметим, что в дополнительном коде в отличие от прямого и обратного кодов существует единственное представление нуля (для дробных чисел  $0 = 0,00\dots00$ ), но вместе с тем имеется и одно особое отрицательное число вида  $1,0\dots00$ , которое является своим собственным дополнением, т. е. это число не имеет дополнительного к нему положительного числа. Поэтому в дополнительных кодах область пред-

ставления положительных и отрицательных чисел оказывается несимметричной относительно нуля.

В микропроцессорах, как и в любых других технических средствах, выполняющих автоматические вычисления, количество разрядов  $n$ , отводимых для представления чисел, ограничено и фиксированно. Этот факт наряду с использованием для представления чисел двоичной системы приводит к существенным отличиям машинной арифметики от обычной арифметики, реализуемой посредством карандаша и бумаги — неограниченных ресурсов [34, 37, 46, 47, 50, 66, 74]. Назовем числа, представляемые в фиксированной разрядной сетке, *числами с ограниченной точностью*, а машинную арифметику — *арифметикой ограниченной точности* (АОТ) [66]. Эти термины подчеркивают принципиальную особенность машинной арифметики, которую программист должен постоянно иметь в виду.

Рассмотрим основные свойства АОТ. Как следует из формул (1.6), (1.7), числа  $|A_R| > A_{R\max}$  и  $|A_R| < A_{R\min}$  нельзя представить в  $n$ -разрядном формате, поскольку представимые числа принадлежат множеству (*диапазону*)  $\{[-A_{R\max}, -A_{R\min}], 0, [+A_{R\min}, +A_{R\max}]\}$  (рис. 1.2, а). Для двоичных дробных и целых чисел эти крайние — *пограничные* — значения ненулевых представимых чисел определены формулами (1.3) и (1.5) (рис. 1.2, б, в). Если при выполнении арифметической операции появляется число  $|A_R| > A_{R\max}$ , это приводит к *переполнению  $n$ -разрядного формата числа* и, как правило, ошибочному представлению результата операции (*ошибка переполнения*). Наоборот, если при выполнении операции возникает такое ненулевое число, что  $|A_R| < A_{R\min}$ , это приводит к *антипереполнению* (потере значности) *формата* и соответственно к *ошибке антипереполнения*, при которой все ненулевые числа  $A_R \in \{-A_{R\min}, +A_{R\min}\}$  приходится представлять в виде нуля. Любое число из указанного интервала называют *машинным нулем*.

Появление ошибок переполнения и машинного нуля в АОТ нарушает ряд аксиом, истинных в арифметике точных чисел, в частности *ассоциативный*  $a + (b - c) = (a + b) - c$  и *дистрибутивный*  $a \cdot (b - c) = a \cdot b - a \cdot c$  законы, что можно показать на примерах [66]. Пусть дробные беззнаковые десятичные числа имеют в 4-разрядном формате значения  $a = 0,2000$ ,  $b = 0,9000$  и  $c = 0,8999$ . Тогда вычисление левой части выражения ас-

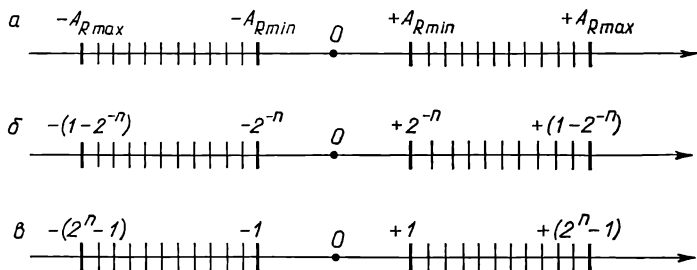


Рис. 1.2. Размещение чисел ограниченной точности на числовой оси:  
 а —  $R$ -ичные числа; б — двоичные дробные числа; в — двоичные целые числа

социативного закона дает значение  $0,2000 + (0,9000 - 0,8999) = 0,2001$ , а при вычислении правой части возникает переполнение:  $(a + b) = 0,2000 + 0,9000 > 0,9999$ . Аналогично при вычислении правой части выражения дистрибутивного закона получаем значение  $0,2000 \times 0,9000 - 0,2000 \cdot 0,8999 = 0,1800 - 0,1799 = 0,0001$ , а при вычислении левой части возникает машинный ноль:  $a \cdot (b - c) = 0,2000 \cdot (0,9000 - 0,8999) = 0,2000 \cdot 0,0001 < 0,0001$ . Поскольку значения правых и левых частей выражений не совпадают, данные законы нарушаются, и, следовательно, в АОТ *порядок операций*, позволяющий избежать ошибок переполнения и машинного нуля, имеет важное значение для получения правильного результата вычислений. Заметим, что порядок операций оказывает влияние и на точность результата. Выявление ошибок переполнения и антипереполнения является задачей, которую необходимо решать при разработке арифметических программ.

Диапазон  $[A_{Rmin}, A_{Rmax}]$  точного представления в АОТ ненулевых дробных  $R$ -ичных чисел имеет дискретный характер, т. е. в пределах этого диапазона точно представляется лишь то множество  $R^n - 1$  чисел, в  $R$ -ичной записи которых отличные от нуля цифры содержатся только в первых  $n$  после запятой разрядах (остальные дополнительные разряды, если предположить их существование, будут содержать нули) [47]. На практике приходится представлять в  $n$ -разрядном формате любые дробные числа из указанного диапазона, в том числе и те (а их большинство), разряды которых, выходящие за рамки принятого формата, содержат ненулевые  $R$ -ичные цифры. Например, десятичная дробь  $0,1$  в двоичном

формате представляется бесконечной периодической двоичной дробью  $0,0(0011)$ . При записи этой дроби в ограниченном формате приходится отбрасывать ряд ненулевых цифр и тем самым представлять ее как приближенное число с ограниченной точностью.

Числа с ограниченной точностью порождаются не только в процессе их начального представления в ограниченном формате, создающем погрешность представления, но и при выполнении арифметических операций над точными или приближенными исходными числами. Так, например, в *арифметике целых чисел*, не имеющей погрешности представления исходных чисел, операция деления порождает неправильные дроби (говорят, что множество целых чисел незамкнуто относительно операции деления), которые в формате целых чисел записываются приближенно целыми числами ограниченной точности. Аналогичные погрешности операций могут иметь место при умножении и делении дробных чисел.

• Как погрешности представления, так и погрешности операций имеют одну природу, связанную с *округлением чисел*. Поэтому рассмотрим *правила округления чисел* и величины, характеризующие точность округленных чисел. Округление *точного числа*  $A_R^*$ , содержащего  $n + k$  ( $k = 1, 2, \dots$ )  $R$ -ичных разрядов, заключается в ограничении его формата  $n$  разрядами. При выполнении этой операции желательно обеспечить наибольшую близость *округленного числа*  $A_R$  (число в формате  $n$ ) к *округляемому числу*  $A_R^*$  (число в формате  $n + k$ ). На практике обычно используют два способа округления: отбрасывание и симметричное округление [34, 50, 64]. *Способ отбрасывания* определяется для дробных и целых  $R$ -ичных чисел формулами (1.6) и (1.7). Согласно этому способу, дополнительные  $k$  разрядов округляемого числа  $A_R^*$  просто исключаются из формата без какой-либо коррекции части числа в разрядах  $n$ . *Симметричное округление* определяется следующими выражениями соответственно для дробных и целых  $R$ -ичных чисел:

$$A_{R\Phi} = \begin{cases} \pm \sum_{i=1}^n a_i \cdot 10_R^{-i}, & \text{если } a_{n+1} < R/2; \\ \pm \sum_{i=1}^n a_i \cdot 10_R^{-i} + 10_R^{-n}, & \text{если } a_{n+1} \geq R/2; \end{cases} \quad (1.9)$$

$$A_R = \begin{cases} \pm \sum_{i=0}^{n-1} a_i \cdot 10_R^i, & \text{если } a_{-1} < R/2; \\ \pm \sum_{i=0}^{n-1} a_i \cdot 10_R^i + 1, & \text{если } a_{-1} \geq R/2. \end{cases} \quad (1.10)$$

При симметричном округлении значение первой из отбрасываемых цифр округляемого числа  $A_R^*$  используется для решения вопроса о коррекции части числа в оставшихся  $n$  разрядах. На практике способ симметричного округления для случая  $a_{n+1} = R/2$  ( $A_{-1} = R/2$ ) иногда дополняют *правилом Гаусса*, в соответствии с которым коррекция оставшейся части числа производится, если цифра  $a_n$  ( $a_0$ ) — четная [15]. Оценим точностные характеристики методов округления: абсолютную и относительную ошибки (погрешности).

Определим *абсолютную ошибку округления*  $\Delta A_R$  как разность значений округляемого  $A_R^*$  и округленного  $A_R$  чисел, а *относительную ошибку округления*  $\delta A_R$  как модуль отношения абсолютной ошибки к значению округляемого числа:

$$\Delta A_R = A_R^* - A_R; \quad \delta A_R = |\Delta A_R / A_R^*|. \quad (1.11)$$

Поскольку в большинстве случаев точное значение округляемого числа  $A_R^*$  неизвестно (число, например, получено с погрешностью в процессе измерения, является иррациональным и т. п.), то неизвестны и точные величины ошибок в выражениях (1.11). Однако почти всегда имеется возможность оценить граничные (предельные) значения ошибок. Определим *граничную абсолютную*  $\Delta_\Gamma$  и *относительную*  $\delta_\Gamma$  *ошибки округления* следующим образом:

$$|\Delta A_R| \leq \Delta_\Gamma; \quad \delta_\Gamma = \Delta_\Gamma / |A_R|. \quad (1.12)$$

Замена  $|A_R^*|$  из формулы (1.11) на  $|A_R|$  в выражении (1.12) не вносит существенной погрешности в значение  $\delta_\Gamma$ , если  $\Delta_\Gamma \ll |A_R|$ . На практике используют по возможности минимальное значение граничной ошибки  $\Delta_\Gamma$ . В тех случаях, когда известна точная величина ошибки  $\Delta A_R$ , она одновременно принимается и в качестве граничной. Очевидно, что граничная абсолютная ошибка любых  $R$ -ичных чисел, представленных в  $n$ -разрядном формате, имеет значения  $\Delta_\Gamma = 10_R^{-n}$  и  $\Delta_\Gamma = 1$  соответственно для

дробных и целых чисел при способе округления с отбрасыванием и  $\Delta_r = 10_R^{-n}/2$  и  $\Delta_r = 1/2$  соответственно для дробных и целых чисел при симметричном округлении. Способ отбрасывания порождает всегда округленные числа «с недостатком», т. е. числа с положительной абсолютной ошибкой, поэтому этот способ называют еще *несимметричным округлением*. Симметричный же способ в зависимости от значения первой отбрасываемой цифры порождает как числа «с недостатком», так и числа «с избытком» (числа с отрицательной абсолютной ошибкой), что, как правило, приводит в процессе многочисленных округлений при вычислениях к *компенсации ошибок* и повышению точности результата. Процессу компенсации ошибок способствует и вышеупомянутое правило Гаусса (оно основывается на предпосылке о равновероятности четных и нечетных чисел). Способ симметричного округления получил в программах, выполняющих арифметическую обработку, преимущественное применение.

В *арифметике с фиксированной запятой* все числа представляются с одинаковой граничной абсолютной ошибкой. Величина же граничной относительной ошибки зависит от модуля округленного числа и возрастает от максимального числа к минимальному:

$$\Delta_r / |A_R|_{\max} \leq \delta_r \leq \Delta_r / |A_R|_{\min}; \quad (1.13)$$

$$2^{-n} \leq \delta_{r2} \leq 1, \quad (1.14)$$

где  $\delta_r$ ,  $\delta_{r2}$  — ошибки  $R$ -ичных и двоичных чисел, использующих правило округления с отбрасыванием. Из выражений (1.13), (1.14) следует, что малые числа, близкие к машинному нулю, могут иметь высокую относительную ошибку (до 100 % при округлении с отбрасыванием и до 50 % при симметричном округлении), т. е. обладать низкой точностью. Если такие числа участвуют в промежуточных вычислениях, точность результата будет того же порядка. Поэтому при программировании вычислений важно обеспечить требуемую точность как для исходных чисел, так и для промежуточных результатов. Это достигается правильным выбором порядка операций и значности чисел.

Понятие значности имеет важное значение для уяснения программистом путей достижения необходимой точности вычислений. *Значность* определяет не общее количество цифр в изображении числа, а только количество



значащих цифр, т. е. всех верных цифр числа, кроме нулей, стоящих слева в изображении числа. Например, числа 12,3; 0,123; 0,000123 имеют по три значащие цифры. Цифры в записи приближенного числа принято считать *верными*, если граничная абсолютная ошибка представления этого числа не превосходит половины единицы (иногда целой единицы) его младшего разряда [15]. Очевидно, что в силу этого определения все цифры округленного числа (при точном округляемом числе) являются верными, хотя фактически могут не совпадать с соответствующими цифрами точного числа. Например, при симметричном округлении точного числа 0,17997 до четырех цифр округленное число имеет значение 0,1800, не совпадающее с точным в трех разрядах, но тем не менее имеющее все верные цифры. Значность чисел определяет граничную относительную ошибку их представления: чем выше значность, тем меньше эта ошибка и, следовательно, тем выше точность числа. Точному числу на числовой оси соответствует точка, а округленному — интервал чисел. На рис. 1.3 показаны интервалы  $0,05 < 0,1 < 0,15$  и  $0,095 < 0,10 < 0,105$  округленных чисел 0,1 и 0,10 соответственно с одной и двумя значащими цифрами. Чем выше значность, тем уже интервал неопределенности и тем ближе округленное число к точному.

Заметим, что в арифметике с фиксированной запятой в отличие от арифметики с плавающей запятой (об этом будет сказано далее, в гл. 2) повышение значности приводит одновременно к существенному увеличению диапазона представимых чисел, т. е. точность и диапазон чисел оказываются тесно связанными друг с другом. Повышение значности в 8-разрядных микропроцессорах достигается, как правило, побайтным увеличением формата чисел (8, 16, 24 двоичных разрядов и т. д.), поскольку *байт* — 8-разрядное двоичное слово — является структурной единицей, требующей минимальных программных затрат для обработки (доступ же к отдельным битам байта связан с усложнением обработки). Программы сложения, вычитания, умножения и деления, рассматриваемые в этой главе, оперируют с данными различного формата, что позволяет выбрать ту или иную программу в зависимости от требуемых точности вычислений и диапазона представления чисел.

При использовании арифметики с фиксированной запятой для выполнения сложных вычислений (последова-

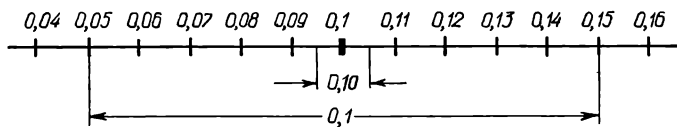


Рис. 1.3. Интервалы округленных чисел различной значности на числовой оси

тельностью арифметических операций) возникает задача расчета масштабов, обеспечивающих значения модулей исходных промежуточных и результирующих чисел в пределах заданного диапазона их представлений [46, 61, 64, 70]. В данной работе для вычислений сложных выражений (см. гл. 4) применяется арифметика с плавающей запятой, поэтому здесь задача расчета масштабов не рассматривается.

## 1.2. СЛОЖЕНИЕ И ВЫЧИТАНИЕ $N$ -БАЙТНЫХ ЧИСЕЛ

### 1.2.1. МЕТОДИКА СЛОЖЕНИЯ И ВЫЧИТАНИЯ

Операции сложения и вычитания точных чисел не вносят погрешности в результат. Свойства этих операций для чисел ограниченной точности определяются следующим правилом: *границная абсолютная ошибка суммы или разности чисел не превышает суммы граничных ошибок представления слагаемых* [34, 64, 70]:

$$\Delta_r\left(\sum_{i=1}^k \pm X_i\right) \leq \sum_{i=1}^k \Delta_r(X_i). \quad (1.15)$$

Это правило учитывает наихудший случай распределения ошибок округления слагаемых: когда ошибки независимы и одного знака. При значительном числе слагаемых, полученных методом симметричного округления, происходит взаимная компенсация ошибок разного знака, и истинная абсолютная ошибка суммы (разности) лишь в исключительных случаях близка к граничной.

*Граничная относительная ошибка суммы слагаемых одного знака находится между наименьшей и наибольшей из граничных относительных ошибок слагаемых* [с учетом формулы (1.12)]:

$$[\delta_r(X_i)]_{\min} \leq \delta_r\left(\sum_{i=1}^k X_i\right) \leq [\delta_r(X_i)]_{\max}. \quad (1.16)$$

Иными словами, точность суммы не уступает точности слагаемых (вследствие компенсации ошибок сумма оказывается обычно точнее слагаемых).

В отличие от суммы разность чисел ограниченной точности может быть менее точной, чем уменьшаемое и вычитаемое. Потеря точности особенно велика в том случае, когда уменьшаемое и вычитаемое близки по абсолютной величине. Такое вычитание приводит к резкому снижению значности результата и, следовательно, к возрастанию *границной относительной ошибки разности*. При организации вычислений необходимо либо избегать вычитания близких чисел, либо брать исходные числа с достаточно высокой точностью, обеспечивающей при вычитании необходимую точность результата.

В 8-разрядных микропроцессорах разрядность формата данных  $n = 8N$ , где  $N$  — количество байтов ( $N \geq 1$ ).  $N$ -байтное (*многобайтное*) беззнаковое двоичное число  $X$  можно рассматривать как  $N$ -разрядное число в смешанной системе счисления с основанием  $2^8 = 256$ , где каждый байт  $X_i$  ( $i = 1, 2, \dots, N$ ) кодирует цифру числа в соответствующем 256-ричном разряде согласно формуле (1.6) или (1.7). Условимся в дальнейшем в  $N$ -байтных данных ( $N > 1$ ) байт с индексом  $i = 1$  называть *младшим* (МЛБ), с индексом  $N$  — *старшим* (СТБ), а с промежуточными индексами — *средними* (СРБ) байтами.

В микропроцессоре КР580 операции сложения (вычитания) чисел для случая  $N = 1$  выполняются с помощью соответствующих команд (см. табл. П. 2). При  $N > 1$  требуется разработка программ. Пусть  $X = (X_N, \dots, X_1)$  и  $Y = (Y_N, \dots, Y_1)$  —  $N$ -байтные числа. В основе программы сложения (вычитания)  $X \pm Y = Z$  лежит *вычислительная схема*, изображенная на рис. 1.4, где  $X_N, \dots, X_2, X_1$  — байты первого слагаемого (уменьшаемого);  $Y_N, \dots, Y_2, Y_1$  — байты второго слагаемого (вычитаемого);  $Z_N, \dots, Z_2, Z_1$  — байты суммы (разности), размещаемые на месте первого слагаемого;  $CU$  — бит переноса, который можно рассматривать перед операцией сложения  $i$ -х байтов как младший разряд дополнительного слагаемого, а после выполнения этой операции — как дополнительный старший разряд суммы (разности)  $i$ -х байтов. В кружках обозначен тип выполняемой операции, стрелки показывают направления передачи информации (информационные потоки) между элементами схемы. Штриховыми линиями выделен повторяющийся процесс, который ис-

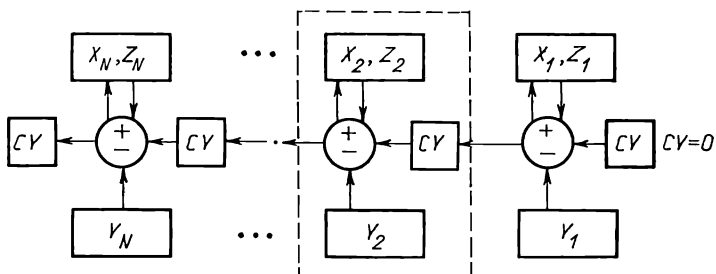


Рис. 1.4. Вычислительная схема операций сложения (вычитания)  $N$ -байтных чисел

пользуется для циклического построения программ. Возможность создания программ на основе приведенной схемы определяется наличием в наборе команд микропроцессора команд сложения (вычитания) с переносом.

Вычислительные схемы дают общее, развернутое в пространстве представление о ходе вычислительного процесса на основе взаимосвязи между структурой данных и операциями (функциями), выполняемыми над этими структурами. В этом заключается основное отличие таких схем от схем алгоритмов, описывающих лишь последовательность операций и умалчивающих о структурах данных. Вычислительные схемы полезны для уяснения и общего описания решаемой задачи.

При сложении двоичных беззнаковых чисел может возникнуть ошибка переполнения. Признаком такой ошибки является значение бита переноса, равное  $CY=1$  после сложения СТБ слагаемых. При вычитании беззнаковых чисел разность может стать отрицательным числом (говорят, что множество беззнаковых чисел незамкнуто относительно операции вычитания). Признаком этого нарушения также является значение бита переноса  $CY=1$  после вычитания СТБ вычитаемого из СТБ уменьшаемого ( $CY=1$  вследствие заема из несуществующего  $(N+1)$ -го байта уменьшаемого). Аналогичные ошибки возникают при сложении (вычитании) двоично-десятичных чисел и двоичных чисел со знаком. Приемы фиксации этих ошибок рассматриваются ниже при описании соответствующих программ сложения и вычитания.

### 1.2.2. ДВОИЧНЫЕ ЧИСЛА

Пусть  $N$ -байтные ( $N < 256$ ) беззнаковые целые двоичные числа размещены в оперативной памяти микропроцессорной системы, причем каждое  $N$ -байтное число расположено в своей локальной области памяти от МЛБ к СТБ в порядке возрастания адресов. Сложение двух таких чисел реализует программа C8N (название программы содержит аббревиатуры типа операции и формата данных):

0400		ORG	400H	
	C8N:			
	;*****			
	;ПОДПРОГРАММА СЛОЖЕНИЯ ДВУХ ЦЕЛЫХ БЕЗЗНАКОВЫХ ДВОИЧНЫХ			
	;ЧИСЕЛ ФОРМАТА В*N.			
	;ВХОДНЫЕ ПАРАМЕТРЫ: (В,С)–АДРЕС МЛБ СЛАГАЕМОГО 1, (Н, L)–			
	;–АДРЕС МЛБ СЛАГАЕМОГО 2, (Е)–КОЛИЧЕСТВО N БАЙТ ЧИСЛА.			
	;ВЫХОДНЫЕ ПАРАМЕТРЫ: (В,С)–АДРЕС МЛБ СУММЫ (СОВПАДАЕТ С			
	;АДРЕСОМ МЛБ СЛАГАЕМОГО 1, )СУ=1–ПРИЗНАК ПЕРЕПОЛНЕНИЯ			
	;СУММЫ.ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ,СОХРАНЯЮТСЯ (В,С), (Е).			
	;ГЛУБИНА СТЕКА–2.			
	;ОЦЕНКА:ДЛИНА–15 БАЙТ,ВРЕМЯ–(45+46*N) ТАКТОВ.			
	;*****			
0400 53	MOV	D,E		; (D)–СОХРАНЕНИЕ СЧЕТЧИКА
0401 C5	PUSH	B		;СОХРАНЕНИЕ АДРЕСА 1
0402 AF	XRA	A		;СУ=0
	;МНОГОВАЙТНОЕ ДВОИЧНОЕ СЛОЖЕНИЕ			
0403 0A	ЦИКЛ: LDAX	B		
0404 BE	ADC	M		;СЛОЖЕНИЕ С УЧЕТОМ ПЕРЕНОСА
0405 02	STAX	B		
0406 03	INX	B		;УВЕЛИЧЕНИЕ АДРЕСА 1
0407 23	INX	H		;УВЕЛИЧЕНИЕ АДРЕСА 2
	;ПРОВЕРКА КОНЦА ЦИКЛА			
0408 1D	DCR	E		
0409 C20304	JNZ	ЦИКЛ		;ЗАЦИКЛИВАНИЕ
	;КОНЕЦ СЛОЖЕНИЯ			
040C C1	POP	B		;ВОССТАНОВЛЕНИЕ АДРЕСА 1
040D 5A	MOV	E,D		;ВОССТАНОВЛЕНИЕ СЧЕТЧИКА
040E C9	RET			;ЕСЛИ СУ=1,ТО ПЕРЕПОЛНЕНИЕ
0000	END			

В этой программе регистровые пары (В, С) и (Н, L) используются для последовательной адресации байтов соответственно первого и второго слагаемых, а регистр (Е) — в качестве счетчика  $N$  байтов. Процедура сложения, согласно схеме рис. 1.4, реализуется в виде цикла сложения  $i$ -х байтов и проверки цикла по содержимому счетчика байтов. Непосредственное сложение двух байтов происходит путем считывания байта первого слагаемого из памяти в аккумулятор микропроцессора и сложения с переносом содержимого аккумулятора с соответствующую-

щим байтом второго слагаемого, после чего результат сложения из аккумулятора засылается в память на место использованного байта первого слагаемого. Если при сложении возникает ненулевой перенос в следующий байт, этот перенос учитывается при сложении очередных байтов слагаемых. Заметим, что начальный перенос перед сложением первых байтов слагаемых  $CY=0$ , а конечный перенос — после сложения СТБ чисел — определяет *признак переполнения*. Программа реализована (как и все последующие программы) в виде подпрограммы, что позволяет легко использовать ее в программах более высокого уровня посредством команд вызова подпрограмм типа CALL. Количественная характеристика программы содержит ее *длину* в байтах и *предельное время выполнения* в машинных тактах. Расчет времени выполнения программы сделан в соответствии с быстродействием команд микропроцессора, указанным в табл. П. 2.

В дальнейшем не будем подробно останавливаться на действиях, описанных в тексте программ, поскольку все они имеют подробный заголовок (ограничен двумя рядами звездочек), строки-комментарии, определяющие функциональное назначение очередного фрагмента программы, и поля-комментарии, поясняющие роль команд в решении задачи. (В прил. 1 дано развернутое описание набора команд микропроцессора КР580, и читатель, желающий разобраться в детальном описании программы, должен владеть этим набором.) При описании программ будут отмечаться лишь характерные их особенности и связи со структурами обрабатываемых данных.

*Вычитание двух N-байтных целых беззнаковых двоичных чисел* реализуется программой B8N:

```

0410                                ORG      410H
                                B8N:
                                ;*****
                                ;ПОДПРОГРАММА ВЫЧИТАНИЯ ДВУХ ЦЕЛЫХ БЕЗЗНАКОВЫХ ДВОИЧНЫХ
                                ;ЧИСЕЛ ФОРМАТА 8*N.
                                ;ВХОДНЫЕ ПАРАМЕТРЫ: (B,C) — АДРЕС МЛБ УМЕНЬШАЕМОГО, (H,L) —
                                ;— АДРЕС МЛБ ВЫЧИТАЕМОГО, (E) — КОЛИЧЕСТВО N БАЙТ ЧИСЛА.
                                ;ВЫХОДНЫЕ ПАРАМЕТРЫ: (B,C) — АДРЕС МЛБ РАЗНОСТИ (СОВПАДАЕТ
                                ;С АДРЕСОМ МЛБ УМЕНЬШАЕМОГО), CY=1 — ПРИЗНАК ОТРИЦАТЕЛЬНО-
                                ;ГО РЕЗУЛЬТАТА. ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ, СОХРАНЯЮТСЯ
                                ;(B,C), (E). ГЛУБИНА СТЕКА — 2.
                                ;ОЦЕНКА: ДЛИНА — 15 БАЙТ, ВРЕМЯ — (45+46*N) ТАКТОВ.
                                ;*****
0410 53      MOV     D,E      ;(D) — СОХРАНЕНИЕ СЧЕТЧИКА
0411 C5      PUSH    B        ;СОХРАНЕНИЕ АДРЕСА

```

0412 AF	XRA	A	#CY=0
	#МНОГОВАЙТНОЕ ДВОИЧНОЕ ВЫЧИТАНИЕ		
0413 0A	ЦИКЛ: LDAX	B	
0414 9E	SBB	M	#ВЫЧИТАНИЕ С УЧЕТОМ ПЕРЕНОСА
0415 02	STAX	B	
0416 03	INX	B	#УВЕЛИЧЕНИЕ АДРЕСА УМЕНЬШАЕМОГО
0417 23	INX	H	#УВЕЛИЧЕНИЕ АДРЕСА ВЫЧИТАЕМОГО
	#ПРОВЕРКА КОНЦА ЦИКЛА		
0418 1D	DCR	E	
0419 C21304	JNZ	ЦИКЛ	#ЗАЦИКЛИВАНИЕ
	#КОНЕЦ ВЫЧИТАНИЯ		
041C C1	POP	B	#ВОССТАНОВЛЕНИЕ АДРЕСА
041D 5A	MOV	E,D	#ВОССТАНОВЛЕНИЕ СЧЕТЧИКА
041E C9	RET		
0000	END		

Эта программа отличается от программы C8N тем, что вместо команды ADC M в ней задействована команда вычитания с переносом SBB M. Заметим, что обе рассмотренные программы применимы для операций и над дробными  $N$ -байтными числами. Можно программы упростить и повысить их быстродействие, если исключить команды сохранения и восстановления регистров типа MOV D, E и MOV E, D, PUSH B и POP B. Необходимость этих вспомогательных команд определяется условиями использования данных подпрограмм в рамках программ более высокого уровня.

Рассмотрим сложение знаковых  $N$ -байтных двоичных чисел, представленных в дополнительном коде. Старший бит СТБ числа используется как знаковый. При сложении знаковых чисел в дополнительных кодах, как и при сложении беззнаковых чисел, может возникнуть ошибка переполнения. Очевидно, что эта ошибка возможна лишь при сложении чисел с одинаковыми знаками, т. е. двух положительных или отрицательных чисел. При сложении таких чисел ошибка переполнения имеет место, если возникает единица переноса в знаковый разряд, т. е. значение знакового разряда суммы изменяется на противоположное. Отсюда следует простое правило выявления ошибки переполнения при сложении чисел в дополнительных двоичных кодах: ошибка имеет место, если знаки слагаемых совпадают, а сумма имеет противоположный знак.

Сложение двух целых  $N$ -байтных двоичных чисел в дополнительном коде реализуется программой CД8N:

C88N:

```

;*****
;ПОДПРОГРАММА СЛОЖЕНИЯ ДВУХ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ В ДО-
;ПОЛНИТЕЛЬНОМ КОДЕ ФОРМАТА 8*N.
;ВХОДНЫЕ ПАРАМЕТРЫ: (B,C) -АДРЕС МЛБ СЛАГАЕМОГО 1, (H,L) -
;АДРЕС МЛБ СЛАГАЕМОГО 2, (E) -КОЛИЧЕСТВО N БАЙТ ЧИСЛА.
;ВЫХОДНЫЕ ПАРАМЕТРЫ: (B,C) -АДРЕС МЛБ СУММЫ (СОВПАДАЕТ
;С АДРЕСОМ МЛБ СЛАГАЕМОГО 1), CY=1-ПРИЗНАК ПЕРЕПОЛНЕНИЯ
;СУММЫ. ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ, СОХРАНЯЮТСЯ (B,C), (E).
;ГЛУБИНА СТЕКА-2.
;ОЦЕНКА: ДЛИНА-34 БАЙТ, ВРЕМЯ НЕ БОЛЕЕ (148+56*N) ТАКТОВ.
;*****

```

```

0420 53      MOV     D,E      ;(D)-СОХРАНЕНИЕ СЧЕТЧИКА
0421 C5      PUSH    B        ;СОХРАНЕНИЕ АДРЕСА 1
0422 AF      XRA     A        ;CY=0
;МНОГОВАЙТНОЕ ДВОИЧНОЕ СЛОЖЕНИЕ
0423 0A      ЦИКЛ: LDAX     B
0424 8E      ADC     M        ;(A)-БАЙТ СУММЫ
;ПРОВЕРКА КОНЦА ЦИКЛА
0425 1D      DCR     E
0426 CA2F04  JZ      PER1     ;ЕСЛИ КОНЕЦ ЦИКЛА
;ПРОДОЛЖЕНИЕ ЦИКЛА
0429 02      STAX    B        ;ЗАПОМИНАНИЕ БАЙТА СУММЫ
042A 03      INX     B        ;УВЕЛИЧЕНИЕ АДРЕСА 1
042B 23      INX     H        ;УВЕЛИЧЕНИЕ АДРЕСА 2
042C C32304  JMP     ЦИКЛ     ;ЗАЦИКЛИВАНИЕ
;ПРОВЕРКА ЗНАКОВ СЛАГАЕМЫХ НА СОВПАДЕНИЕ
042F 5F      PER1: MOV     E,A  ;(E)-СОХРАНЕНИЕ СТБ СУММЫ
0430 0A      LDAX    B        ;(A)-СТБ СЛАГАЕМОГО 1
0431 AE      XRA     M        ;(M)-СТБ СЛАГАЕМОГО 2
0432 7B      MOV     A,E
0433 02      STAX    B        ;ЗАПОМИНАНИЕ СТБ СУММЫ
0434 37      STC
0435 FA3E04  JM      PER2     ;ЕСЛИ ЗНАКИ РАЗНЫЕ
;ПЕРЕПОЛНЕНИЕ: ЗНАКИ СУММЫ И СЛАГАЕМЫХ НЕ СОВПАДАЮТ?
0438 7E      MOV     A,M      ;(A)-СТБ СЛАГАЕМОГО 2
0439 AB      XRA     E        ;(E)-СТБ СУММЫ
043A 37      STC
043B FA3F04  JM      PER3     ;ЕСЛИ ПЕРЕПОЛНЕНИЕ
043E 3F      PER2: CMC
043F C1      PER3: POP     B    ;ВОССТАНОВЛЕНИЕ АДРЕСА 1
0440 5A      MOV     E,D      ;ВОССТАНОВЛЕНИЕ СЧЕТЧИКА
0441 C9      RET
0000      END

```

Программа строится по аналогии с программой C8N и отличается от нее тем, что проверка конца цикла осуществляется до операции записи суммы очередных байтов на место использованного байта первого слагаемого. Такое изменение хода процедуры позволяет при сложении старших байтов сохранить знак первого слагаемого с целью использования его при анализе переполнения, выполняемом в соответствии с вышерассмотренным прави-



лом. Данная программа применима также для сложения дробных чисел.

При необходимости *умножения и деления многобайтных двоичных чисел* можно воспользоваться программами, приведенными в литературе [44].

### 1.2.3. ДЕСЯТИЧНЫЕ ЧИСЛА

Десятичные числа в микропроцессорах представляются, как отмечалось выше, в смешанной двоично-десятичной системе. При этом байт данных состоит из двух десятичных цифр формата  $8 = 2 \cdot 4$ , а  $N$ -байтные данные — из  $2N$ -разрядного десятичного числа. *Сложение двоично-десятичных чисел* отличается от сложения двоичных чисел условиями формирования переноса между двоичными тетрадами [5, 16]. Обозначим десятичную и шестнадцатеричную цифры соответственно как  $a_{10}$  и  $a_{16}$ . Тогда условие *двоично-десятичного переноса* имеет вид

$$a_{10} = \begin{cases} a_{16}, & \text{если } 0 \leq a_{16} \leq 9; \\ a_{16} + 6, & \text{если } a_{16} > 9. \end{cases} \quad (1.17)$$

Микропроцессор КР580 содержит *команду десятичной коррекции ДАА*, реализующую условие (1.17), которая позволяет использовать для сложения двоично-десятичных чисел команды сложения двоичных чисел. Для получения правильной двоично-десятичной суммы необходимо выполнить над результатом двоичного сложения команду десятичной коррекции ДАА.

Сложение  $N$ -байтных целых беззнаковых двоично-десятичных чисел реализуется программой C8N10:

```

0450                                ORG      450H
                                C8N10:
; *****
; ПОДПРОГРАММА ДЕСЯТИЧНОГО СЛОЖЕНИЯ ДВУХ ЦЕЛЫХ БЕЗЗНАКО-
; Вых двоично-десятичных (код 8421) чисел формата (2*4)*N.
; ВХОДНЫЕ ПАРАМЕТРЫ: (B,C) — АДРЕС МЛБ СЛАГАЕМОГО 1, (H,L) —
; — АДРЕС МЛБ СЛАГАЕМОГО 2, (E) — КОЛИЧЕСТВО N БАЙТ ЧИСЛА.
; ВЫХОДНЫЕ ПАРАМЕТРЫ: (B,C) — АДРЕС МЛБ СУММЫ (СОВПАДАЕТ С
; АДРЕСОМ МЛБ СЛАГАЕМОГО 1), CY=1 — ПРИЗНАК ПЕРЕПОЛНЕНИЯ
; СУММЫ. ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ, СОХРАНЯЮТСЯ (B,C),
; (E). ГЛУБИНА СТЕКА — 2.
; ОЦЕНКА: ДЛИНА — 16 БАЙТ, ВРЕМЯ — (45+50*N) ТАКТОВ.
; *****
0450 53      MOV     D,E      ; (D) — СОХРАНЕНИЕ СЧЕТЧИКА
0451 C5      PUSH    B        ; СОХРАНЕНИЕ АДРЕСА 1
0452 AF      XRA     A        ; CY=0

```

;МНОГОБАЙТНОЕ ДВОИЧНО-ДЕСЯТИЧНОЕ СЛОЖЕНИЕ			
0453 0A	ЦИКЛ:	LDAH	B
0454 8E		ADC	M
0455 27		DAA	
0456 02		STAX	B
0457 03		INX	B
0458 23		INX	H
	;ПРОВЕРКА КОНЦА ЦИКЛА		
0459 1D		DCR	E
045A C25304		JNZ	ЦИКЛ
	;ЗАЦИКЛИВАНИЕ		
	;КОНЕЦ СЛОЖЕНИЯ		
045D C1		POP	B
045E 5A		MOV	E,D
045F C9		RET	
0000		END	

Программа аналогична программе C8N, за исключением одной дополнительной команды DAA, вставленной в цикл сложения двух байтов. Ошибка переполнения при десятичном сложении фиксируется, если после сложения СТБ слагаемых и выполнения команды DAA имеет место ненулевой перенос  $CY = 1$ . Программа также применима к дробным беззнаковым числам.

Реализация десятичного *вычитания*  $N$ -байтных *двоично-десятичных чисел* сложнее их суммирования, поскольку микропроцессор КР580 не содержит команды десятичной коррекции при двоичном вычитании двоично-десятичных чисел. Поэтому вычитание необходимо выполнять путем суммирования уменьшаемого с дополнительным десятичным кодом вычитаемого, что позволяет применить команду десятичной коррекции [5, 52]:

$$X - Y = X + (10^{2N} - |Y|),$$

где  $10^{2N}$  — граница числа [согласно формуле (1.8)];  $(10^{2N} - |Y|)$  — *дополнительный десятичный код* вычитаемого. Этот код целесообразно формировать в два этапа:

$$[Y]_d = 10^{2N} - |Y| = (10^{2N} - 1) - |Y| + 1,$$

т. е. вначале получить  $2N$ -разрядное десятичное число вида 99...99, вычесть из него  $|Y|$  (такое двоично-десятичное вычитание полностью эквивалентно обычному двоичному вычитанию, так как не приводит к переносам между тетрадами), а затем к результату добавить единицу. Десятичное вычитание реализует программа B8N10:

B8N10:

```

;*****
;ПОДПРОГРАММА ДЕСЯТИЧНОГО ВЫЧИТАНИЯ ДВУХ ЦЕЛЫХ БЕЗЗНА-
;КОВЫХ ДВОИЧНО-ДЕСЯТИЧНЫХ (КОД 8421) ЧИСЕЛ ФОРМАТА (2*4)*N
;ВХОДНЫЕ ПАРАМЕТРЫ: (В,С) -АДРЕС МЛБ УМЕНЬШАЕМОГО, (Н, L) -
;-АДРЕС МЛБ ВЫЧИТАЕМОГО, (Е) -КОЛИЧЕСТВО N БАЙТ ЧИСЛА.
;ВЫХОДНЫЕ ПАРАМЕТРЫ: (В,С) -АДРЕС МЛБ РАЗНОСТИ (СОВПАДАЕТ
;С АДРЕСОМ МЛБ УМЕНЬШАЕМОГО), CY=0 -ПРИЗНАК ОТРИЦАТЕЛЬНОГО
;РЕЗУЛЬТАТА. ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ, СОХРАНЯЮТСЯ (В,С),
; (Е). ГЛУБИНА СТЕКА -4.
;ОЦЕНКА: ДЛИНА -26 БАЙТ, ВРЕМЯ - (78+69*N) ТАКТОВ.
;*****
;ПОДГОТОВКА РЕГИСТРОВ

```

```

0470 D5      PUSH    D      ;СОХРАНЕНИЕ СЧЕТЧИКА
0471 C5      PUSH    B      ;СОХРАНЕНИЕ АДРЕСА РАЗНОСТИ
0472 50      MOV     D,B
0473 43      MOV     B,E     ; (В) -СЧЕТЧИК БАЙТ
0474 59      MOV     E,C     ; (D,E) -АДРЕС УМЕНЬШАЕМОГО
0475 0E00    MVI     C,0     ; ПСЕВДОСЛАГАЕМОЕ=0
0477 37      STC          ;CY=1

;ПОБАЙТНОЕ ПРЕОБРАЗОВАНИЕ ВЫЧИТАЕМОГО В ДОПОЛНИТЕЛЬНЫЙ
;КОД И ДВОИЧНО-ДЕСЯТИЧНОЕ СЛОЖЕНИЕ С УМЕНЬШАЕМЫМ
ЦИКЛ:
0478 3E99    MVI     A,99H
047A 89      ADC     C      ;СЛОЖЕНИЕ С УЧЕТОМ ПЕРЕНОСА
047B 96      SUB     M      ;ДОПОЛНЕНИЕ ВЫЧИТАЕМОГО
047C EB      XCHG
047D 86      ADD     M      ;СЛОЖЕНИЕ С УМЕНЬШАЕМЫМ
047E EB      XCHG
047F 27      DAA          ;ДВОИЧНО-ДЕСЯТИЧНАЯ КОРРЕКЦИЯ

0480 12      STAX    D
0481 13      INX     D      ;УВЕЛИЧЕНИЕ АДРЕСА УМЕНЬШАЕМОГО
0482 23      INX     H      ;УВЕЛИЧЕНИЕ АДРЕСА ВЫЧИТАЕМОГО

;ПРОВЕРКА КОНЦА ЦИКЛА
0483 05      DCR     B
0484 C27804  JNZ     ЦИКЛ   ;ЗАЦИКЛИВАНИЕ

;КОНЕЦ ВЫЧИСЛЕНИЯМ
0487 C1      POP     B      ;ВОССТАНОВЛЕНИЕ АДРЕСА РАЗНОСТИ
0488 D1      POP     D      ;ВОССТАНОВЛЕНИЕ СЧЕТЧИКА
0489 C9      RET
0000      END

```

Программа имеет три интересные особенности: 1) преобразование вычитаемого и дополнительный код производится побайтно и совмещается в цикле с операцией десятичного сложения с соответствующим байтом уменьшаемого; 2) для адресации уменьшаемого в процессе работы программы применяется регистровая пара (D, E), а не (B, C), что позволяет, используя команду обмена регистровых пар XCHG, быстро адресовать в арифметических командах сложения, обращающихся к памяти, как уменьшаемое, так и вычитаемое; 3) применяется псевдослагаемое (C)=0, которое позволяет использовать в цикле

более быстродействующую команду ADC C (4 такта) вместо команды ACI 0 (7 тактов).

### 1.3. УМНОЖЕНИЕ ДВОИЧНЫХ ЧИСЕЛ

#### 1.3.1. МЕТОДИКА УМНОЖЕНИЯ

Умножение точных  $n$ -разрядных чисел дает *точное*  $2n$ -разрядное *произведение*. Погрешность этой операции для чисел ограниченной точности определяется следующим правилом: *границная относительная ошибка произведения* приблизительно равна сумме граничных относительных ошибок представления сомножителей [15, 70].

$$\delta_r \left( \prod_{i=1}^k X_i \right) \approx \sum_{i=1}^k \delta_r(X_i), \quad (1.18)$$

где  $X_i$  — сомножители. Из формулы (1.18) следует, что результат умножения чисел ограниченной точности (будь это целые или дробные числа) не может иметь больше значащих цифр, чем их имеет наименее точный сомножитель. Очевидно, что при умножении  $n$ -разрядных чисел ограниченной точности произведение является также  $n$ -разрядным числом, причем, если значность каждого из двух сомножителей равна  $n$ , то в произведении предпоследняя цифра —  $(n - 1)$ -я для дробного и 2-я для целого числа — безусловно верна, а последняя не вполне точна (сомнительна). При увеличении количества сомножителей пропорционально растет граничная относительная ошибка произведения и уменьшается его значность.

Микропроцессор КР580 не содержит команд умножения чисел, и поэтому умножение требует разработки программ. Любой программный метод выполнения умножения (за исключением громоздкого табличного метода) основан на многократном выполнении операций сложения. Очевидно, что простейший способ умножения целых беззнаковых чисел  $X \cdot Y = Z$  есть суммирование множителя  $X$  с накоплением его столько раз, сколько единиц содержит значение множителя  $Y$ . Этот способ требует при больших значениях  $Y$  значительных затрат времени на выполнение многократных операций сложения, и поэтому применение его ограничено.

Сокращение количества операций сложения при реализации умножения основано на методах совместного анализа цифр множителя [32, 33, 46, 61]. Эти методы дробят

процесс получения произведения на ряд шагов, связанных с формированием *частичных произведений* (ЧП) — произведений множимого на отдельные разряды или группы разрядов множителя — и их суммированием, т. е. формированием *суммы частичных произведений* (СЧП). Ускоренные методы умножения используют анализ одновременно двух и более цифр множителя для получения ЧП, а обычные (неускоренные) методы — анализ поочередно каждой отдельной цифры множителя. Ускоренные методы требуют специальных схмотехнических решений, которые, как правило, не используются в микропроцессорах, и поэтому здесь применяются обычные методы неускоренного умножения.

Методы *умножения двоичных беззнаковых чисел* основаны на представлении произведения в виде полинома [с учетом формулы (1.4)]:

$$\begin{aligned} Z = X \cdot Y &= X \left( \sum_{i=0}^{n-1} y_i \cdot 2^i \right) = \sum_{i=0}^{n-1} (X \cdot y_i) \cdot 2^i = \\ &= \sum_{i=0}^{n-1} \text{ЧП}_i \cdot 2^i, \end{aligned} \quad (1.19)$$

где  $y_i \in \{0,1\}$  — двоичные цифры множителя;  $\text{ЧП}_i = X \times y_i$  — частичные произведения ( $\text{ЧП}_i = X$ , если  $y_i = 1$ , и  $\text{ЧП}_i = 0$ , если  $y_i = 0$ ). Заметим, что умножение  $\text{ЧП}_i \neq 0$  на степень  $2^i$  эквивалентно сдвигу множимого на  $i$  разрядов влево. Формирование произведения, согласно формуле (1.9), представляется как последовательность таких действий: 1) обнуление СЧП; 2) анализ младшего разряда множителя: если  $y_0 = 1$ , выполняется сложение  $\text{ЧП}_0 = X$  с СЧП и переход к п. 3; если  $y_0 = 0$ , сразу переход к п. 3; 3) сдвиг множимого влево; 4) анализ очередного разряда множителя: если  $y_1 = 1$ , выполняется сложение  $\text{ЧП}_1 \cdot 2^1$  с СЧП и переход к п. 5; если  $y_1 = 0$ , непосредственно переход к п. 5; 5) сдвиг множимого влево; 6) анализ очередного  $y_2$  разряда множителя и т. д. После анализа старшего  $y_{n-1}$  разряда множителя осуществляется последнее сложение  $\text{ЧП}_{n-1} \cdot 2^{n-1}$  с СЧП (если  $y_{n-1} = 1$ ), и процесс прекращается. Результирующая СЧП является искомой. Очевидно, что неподвижную СЧП и сдвигаемое влево на  $n - 1$  разряд множимое необходимо размещать в  $2n$ -разрядных форматах, причем операция сложения должна обрабатывать  $2n$ -разрядные данные.

Форматы множимого и операции сложения можно ограничить  $n$  разрядами, если изменить процедуру обработки, определяемую формулой (1.19), следующим образом:

$$\begin{aligned} Z &= \sum_{i=0}^{n-1} X \cdot 2^i \cdot y_i = \sum_{i=0}^{n-1} (X \cdot 2^n) \cdot 2^{i-n} \cdot y_i = \\ &= \sum_{i=0}^{n-1} V \cdot 2^{i-n} \cdot y_i, \end{aligned}$$

где  $V = X \cdot 2^n$  — множимое, сдвинутое влево на  $n$  разрядов.

Преобразуем полученное выражение по *схеме Горнера*:

$$Z = (...((0 + V \cdot y_0) \cdot 2^{-1} + V \cdot y_1) \cdot 2^{-1} + ... + V \cdot y_{n-1}) \cdot 2^{-1}. \quad (1.20)$$

Выражения в скобках в формуле (1.20) представляют собой последовательные значения СЧП<sub>*i*</sub>, определяемые рекуррентной формулой

$$\text{СЧП}_i = \text{СЧП}_{i-1} \cdot 2^{-1} + V \cdot y_{i-1}, \quad \text{СЧП}_0 = 0, \quad i \in \{1, \dots, n\}, \quad (1.21)$$

где  $V \cdot y_{i-1} = \text{ЧП}_{i-1}$ . Заметим, что умножение СЧП на  $2^{-1}$  эквивалентно ее сдвигу вправо на один разряд.

В соответствии с формулами (1.20) и (1.21) процесс формирования произведения выглядит следующим образом: 1) обнуление СЧП; 2) анализ младшего разряда множителя: если  $y_0 = 1$ , сложение  $\text{ЧП}_0 = X$  с СЧП и переход к п. 3; если  $y_0 = 0$ , непосредственно переход к п. 3; 3) сдвиг СЧП вправо; 4) анализ очередного разряда  $y_1$  множителя и т. д. После анализа старшего разряда  $y_{n-1}$  множителя осуществляются последнее сложение  $\text{ЧП}_{n-1}$  с СЧП (если  $y_{n-1} = 1$ ) и последний сдвиг СЧП вправо, после чего процесс прекращается. Таким образом, данный процесс умножения требует для представления неподвижного множимого  $V = X \cdot 2^n$   $n$ -разрядного формата (поскольку младшие  $n$  разрядов множимого равны нулю) и такого же формата для операции сложения (сложение выполняется со старшими  $n$  разрядами СЧП). Формулы (1.19) и (1.20) определяют алгоритмы *умножения с младших разрядов множителя* соответственно при неподвижной и подвижной СЧП.

Аналогичные формулы имеют место при *умножении со старших разрядов множителя*:

$$Z = (...(((0) \cdot 2^1 + X \cdot y_{n-1}) \cdot 2^1 + X \cdot y_{n-2}) \cdot 2^1 + \dots + X \times X y_1) \cdot 2^1 + X \cdot y_0; \quad (1.22)$$

$$\text{СЧП}_i = \text{СЧП}_{i-1} \cdot 2^1 + X \cdot y_{n-i}; \text{СЧП}_0 = 0, i \in \{1, \dots, n\}, \quad (1.23)$$

$$Z = \sum_{i=0}^{n-1} V \cdot y_{n-i-1} \cdot 2^{-(i+1)}, \quad V = X \cdot 2^n. \quad (1.24)$$

Формулы (1.22) и (1.24) получены из полинома (1.19) соответственно по схеме Горнера и перенумерацией индексов. Выражения (1.22) и (1.23) определяют процесс умножения с неподвижным  $n$ -разрядным множимым и подвижной, сдвигаемой влево  $2n$ -разрядной СЧП. Этот процесс реализуется такой последовательностью действий: 1) обнуление СЧП; 2) сдвиг СЧП влево; 3) анализ старшего разряда множителя: если  $y_{n-1} = 1$ , выполняется сложение  $\text{ЧП}_{n-1} = X \cdot y_{n-1}$  с СЧП и переход к п. 4; если  $y_{n-1} = 0$ , прямой переход к п. 4; 4) сдвиг СЧП влево; 5) анализ очередного  $y_{n-2}$  разряда множителя и т. д. После анализа младшего  $y_0$  разряда множителя выполняется последнее сложение  $\text{ЧП}_0 = X \cdot y_0$  с СЧП (если  $y_0 = 1$ ), и процесс прекращается.

Выражения (1.24) определяют процесс умножения с неподвижной СЧП и подвижным, сдвигаемым вправо  $2n$ -разрядным множимым. Этот процесс имеет такую последовательность действий: 1) обнуление СЧП; 2) сдвиг множимого вправо; 3) анализ старшего разряда множителя: если  $y_{n-1} = 1$ , выполняется сложение  $\text{ЧП}_{n-1} \cdot 2^{-1}$  с СЧП и переход к п. 4; если  $y_{n-1} = 0$ , прямой переход к п. 4; 4) сдвиг множимого вправо; 5) анализ очередного разряда  $y_{n-2}$  множителя и т. д. После анализа младшего разряда  $y_0$  множителя осуществляется последнее сложение  $\text{ЧП}_0 \cdot 2^{-1}$  с СЧП (если  $y_0 = 1$ ), и процесс прекращается.

Рассмотренным четырем алгоритмам умножения соответствуют четыре *вычислительные схемы*, используемые при разработке программ умножения (рис. 1.5): схема 1 (рис. 1.5, а) соответствует выражениям (1.20), (1.21); схема 2 (рис. 1.5, б) — формуле (1.19); схема 3 (рис. 1.5, в) — соотношениям (1.22), (1.23); схема 4 (рис. 1.5, г) — формулам (1.24). На схемах обозначены: МН — множители, ММ — множимые (с указанием в

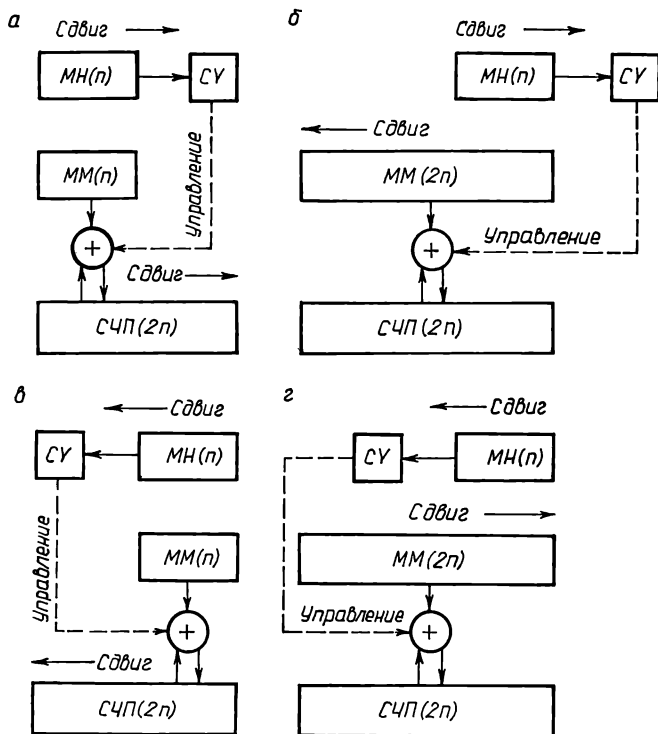


Рис. 1.5. Вычислительные схемы операции умножения

скобках разрядности данных). Анализ разрядов множителя осуществляется путем сдвига его разрядов вправо (умножение с младших разрядов в схемах 1,2) или влево (умножение со старших разрядов в схемах 3, 4) с модификацией признака переноса  $СУ$ , используемого для управления процессом накопления  $СЧП$ .

Последовательность выполнения в вычислительных схемах операций сдвига и сложения поясняется в табл. 1.1—1.4 на примере умножения двух беззнаковых целых четырехразрядных чисел:  $ММ(4) = 1111_2 = 15_{10}$  и  $МН(4) = 0101_2 = 5_{10}$ . В таблицах процесс умножения раскрыт как последовательность пяти шагов: подготовительного (обнуления  $СЧП$ ) и четырех шагов сдвига множителя с анализом признака переноса и выполнения сдвигов  $ММ$ ,  $СЧП$  и операции сложения. Тип выполняе-



мой операции условно изображен в последнем столбце каждой таблицы: «→» означает сдвиг вправо, «←» — сдвиг влево, «+» — сложение, «=» — присвоение значения. Справа от знака операций указаны аббревиатуры соответствующих данных, над которыми выполняется указанная операция (в операции сложения вторым слагаемым всегда является СЧП). Результат умножения — 8-разрядное число  $01001011_2 = 75_{10}$ .

**Табл. 1.1. Умножение по схеме 1**

Номер шага	Сдвиг МН		Операции с СЧП и ММ		
	МН (4)	СЧ	СЧ	СЧП (8)	тип
0	0101	—	—	00000000	=СЧП
1	—010	1	0	1111 11110000 01111000	+ММ =СЧП →СЧП
2	— — 01	0	0	00111100	→СЧП
3	— — — 0	1	1 0	1111 00101100 10010110	+ММ =СЧП →СЧП
4	— — — —	0	0	01001011	→СЧП

**Табл. 1.2. Умножение по схеме 2**

Номер шага	Сдвиг МН		Операции с СЧП и ММ	
	МН (4)	СЧ	СЧП (8)	тип
0	0 1 0 1	—	00000000	=СЧП
1	— 0 1 0	1	1111 00001111 00011110	+ММ =СЧП ←ММ
2	— — 0 1	0	00001111 00111100	=СЧП ←ММ
3	— — — 0	1	00111100 01001011 01111000	+ММ =СЧП ←ММ
4	— — — —	0	01001011	=СЧП

Табл. 1.3. Умножение по схеме 3

Номер шага	Сдвиг МН		Операции с СЧП и ММ	
	СЧ	МН (4)	СЧП (8)	тип
0	—	0101	00000000	=СЧП
1	0	101—	00000000	←СЧП
2	1	01— —	00000000 1111 00001111	←СЧП +ММ =СЧП
3	0	1— — —	00011110	←СЧП
4	1	— — — —	00111100 1111 01001011	←СЧП +ММ =СЧП

Табл. 1.4. Умножение по схеме 4

Номер шага	Сдвиг МН		Операции с СЧП и ММ	
	СЧ	МН (4)	СЧП (8)	тип
0	—	0101	00000000	=СЧП
1	0	101—	01111000 00000000	→ММ =СЧП
2	1	01— —	00111100 00111100 00111100	→ММ +ММ =СЧП
3	0	1— — —	00011110 00111100	→ММ =СЧП
4	1	— — — —	00001111 00001111 01001011	→ММ +ММ =СЧП

Достоинства и недостатки приведенных вычислительных схем описаны далее, при рассмотрении конкретных программ умножения.

Методы *умножения беззнаковых чисел* применимы и для вычисления произведений *двоичных знаковых чисел* в дополнительных кодах. Пусть целые двоичные сомножители  $X$  и  $Y$  представлены в дополнительном коде в  $n$ -разрядном формате, старший разряд которого исполь-

зуется для представления знака. При умножении этих чисел возможны четыре ситуации [20, 46, 61]:

1)  $X > 0, Y > 0$ . Так как  $[X > 0]_d = X$  и  $[Y > 0]_d = Y$ , то  $[X \cdot Y > 0]_d = X \cdot Y$ , т. е. результат умножения совпадает с произведением беззнаковых чисел;

2)  $X < 0, Y > 0$ . Согласно формуле (1.8),  $[X < 0]_d = 2^n + X$ , и произведение, полученное методом умножения беззнаковых чисел, — *псевдопроизведение* — равно  $[X]_d \cdot [Y]_d = Y \cdot 2^n + X \cdot Y$ . Правильный же результат равен  $[X \cdot Y < 0]_d = 2^{2n} + X \cdot Y$  (с учетом того, что формат произведения  $2n$ ). Очевидно, что для получения правильного результата из псевдопроизведения необходимо к последнему прибавить  $2^{2n}$  (эту операцию практически не надо выполнять, так как данное число — граница произведения — выходит за рамки формата произведения) и вычесть  $Y \cdot 2^n$ , т. е. вычесть множитель, сдвинутый влево на  $n$  разрядов;

3)  $X > 0, Y < 0$ . Эта ситуация аналогична предыдущей. Для получения правильного произведения необходимо вычесть из псевдопроизведения число  $X \cdot 2^n$ , т. е. вычесть множимое, сдвинутое влево на  $n$  разрядов;

4)  $X < 0, Y < 0$ . Так как  $[X < 0]_d = 2^n + X$  и  $[Y < 0]_d = 2^n + Y$ , то псевдопроизведение равно  $[X]_d \cdot [Y]_d = 2^{2n} + 2^n \cdot X + 2^n \cdot Y + X \cdot Y$ , а правильное произведение  $[X \cdot Y > 0]_d = X \cdot Y$ . Следовательно, для получения правильного произведения из псевдопроизведения необходимо вычесть числа  $2^n \cdot X$  и  $2^n \cdot Y$ , т. е. вычесть и множитель, и множимое, сдвинутые предварительно на  $n$  разрядов влево.

Помимо прямой коррекции произведения чисел в дополнительном коде путем вычитания поправок из псевдопроизведения, существуют методы, обеспечивающие автоматическое введение поправок при любых знаках сомножителей. В частности, известен *алгоритм Бута* [5, 61]. Его программная реализация для микропроцессора КР580 подробно рассматривается далее (см. п. 1.3.3).

Методы умножения двоичных чисел рассматривались выше на примере целых чисел. Все они справедливы и по отношению к дробным числам. Необходимо помнить, что дробные числа являются, как правило, числами ограниченной точности, и поэтому формат их произведения обычно не превышает формата сомножителей (в отличие от удвоенного формата произведения точных целых

дробных, знаковых и беззнаковых чисел различных форматов, обеспечивающих необходимую точность и диапазон значений как сомножителей, так и их произведений.

### 1.3.2. ЦЕЛЫЕ БЕЗЗНАКОВЫЕ ЧИСЛА

#### 1.3.2.1. Формат 8·8=16

Программа У8 реализует простейший способ *умножения путем накопления* суммы множимого:

```

0500                                ORG      500H
                                УВ:
;*****
;ПОДПРОГРАММА УМНОЖЕНИЯ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ БЕЗ ЗНАКА
;ФОРМАТА 8×N=16,ГДЕ N<4.
;МЕТОД УМНОЖЕНИЯ:НАКОПЛЕНИЕ СУММЫ МНОЖИМОГО В СООТВЕТ-
;СТВИИ С ЧИСЛОВЫМ ЭКВИВАЛЕНТОМ МНОЖИТЕЛЯ.
;ВХОДНЫЕ ПАРАМЕТРЫ: (С) -МНОЖИТЕЛЬ, (Е) -МНОЖИМОЕ. ВЫХОДНОЙ
;ПАРАМЕТР: (Н, L) -ПРОИЗВЕДЕНИЕ. ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ,
;КРОМЕ (В), СОХРАНЯЮТСЯ (С), (Е).
;ОЦЕНКА: ДЛИНА-12 БАЙТ, ВРЕМЯ-НЕ БОЛЕЕ 213 ТАКТОВ.
;*****
;ОБНУЛЕНИЕ НАКОПИТЕЛЯ СУММЫ
0500 AF          XRA      A
0501 67          MOV      H,A
0502 6F          MOV      L,A
0503 57          MOV      D,A
;ПРОВЕРКА МНОЖИТЕЛЯ НА 0
0504 81          ADD      C      ; (А) -МНОЖИТЕЛЬ
0505 C8          RZ          ;ЕСЛИ МНОЖИТЕЛЬ=0
;НАКОПЛЕНИЕ СУММЫ
0506 19          ЦИКЛ: DAD      D
0507 3D          DCR      A
0508 C20605      JNZ      ЦИКЛ      ;ЗАЦИКЛИВАНИЕ
050B C9          RET
0000            END

```

Программа имеет наименьшую среди программ умножения длину (12 байт) и наиболее быстрый цикл накопления суммы (25 тактов), определяемый быстродействием трех команд цикла, но вследствие выполнения при умножении большого количества таких циклов (до 255 при максимальном значении множителя) в целом работает медленно:  $T \leq 38 + 25 |MN|$  тактов. При  $|MN| < 8$  время не превышает 213 тактов, и в этой области программа У8 превосходит по быстродействию все аналогичные программы умножения. Заметим, что сложение однобайтного множимого в регистре (Е) с двухбайтной суммой в регистровой паре (Н, L) осуществляется в программе командой *двухбайтного сложения* DAD D (СБ

в (D) = 0), что минимизирует длительность цикла накопления.

Программа У88А реализует *умножение по вычислительной схеме 1* (см. рис. 1.5, а):

```

0510                                ORG      510H
                                У88А:
;*****
;ПОДПРОГРАММА УМНОЖЕНИЯ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ БЕЗ ЗНАКА
;ФОРМАТА 8*8=16.ВАРИАНТ А.
;МЕТОД УМНОЖЕНИЯ:8 ЦИКЛОВ СДВИГОВ МНОЖИТЕЛЯ И ТЕКУЩЕЙ
;СУММЫ ЧАСТИЧНЫХ ПРОИЗВЕДЕНИЙ ВПРАВО.
;ВХОДНЫЕ ПАРАМЕТРЫ:(С)-МНОЖИТЕЛЬ,(Е)-МНОЖИМОЕ.ВЫХОДНОЙ
;ПАРАМЕТР:(Н,L)-ПРОИЗВЕДЕНИЕ.ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ,
;СОХРАНЯЕТСЯ (С).ДОПОЛНИТЕЛЬНАЯ ТОЧКА ВХОДА В ПОДПРО-
;ГРАММУ-У88АА ((D)-МНОЖИМОЕ,(Е)=0,(С)-МНОЖИТЕЛЬ).
;ОЦЕНКА:ДЛИНА-30 БАЙТ,ВРЕМЯ-НЕ БОЛЕЕ 679 ТАКТА.
;*****
;ОБНУЛЕНИЕ ТЕКУЩЕЙ СУММЫ ЧП
0510 AF      XRA      A
0511 67      MOV      H,A
0512 6F      MOV      L,A
;ПЕРЕМЕЩЕНИЕ МНОЖИМОГО
0513 53      MOV      D,E      ;(D)-МНОЖИМОЕ
0514 5F      MOV      E,A      ;(E)=0
;ПРОВЕРКА СОМНОЖИТЕЛЕЙ НА 0
0515 81      ADD      C
0516 C8      RZ          ;ЕСЛИ МНОЖИТЕЛЬ=0
0517 AF      XRA      A
0518 82      ADD      D
0519 C8      RZ          ;ЕСЛИ МНОЖИМОЕ=0
051A 0608    У88АА: MVI      B,8      ;СЧЕТЧИК ЦИКЛОВ
;СДВИГ МНОЖИТЕЛЯ ВПРАВО
051C 79      ЦИКЛ: MOV      A,C
051D 0F      RRC
051E 4F      MOV      C,A
051F D22305  JNC      ПЕР      ;ЕСЛИ РАЗРЯД МНОЖИТЕЛЯ =0
;СЛОЖЕНИЕ МНОЖИМОГО С ТЕКУЩЕЙ СУММОЙ ЧП
0522 19      DAD      D
;СДВИГ ТЕКУЩЕЙ СУММЫ ЧП ВПРАВО
0523 7C      ПЕР:  MOV      A,H
0524 1F      RAR
0525 67      MOV      H,A
0526 7D      MOV      A,L
0527 1F      RAR
0528 6F      MOV      L,A
;ПРОВЕРКА КОНЦА ЦИКЛА
0529 05      DCR      B
052A C21C05  JNZ      ЦИКЛ      ;ЗАЦИКЛИВАНИЕ
052D C9      RET
0000                                END

```

В программе максимальная длительность цикла, вычисленная в предположении, что каждый разряд множителя равен 1, составляет 77 тактов, а время выполнения программы  $T \leq 63 + 77 \cdot 8 = 679$  тактов. Длину про-

граммы и ее время можно уменьшить (на 5 байт и 22 такта), если исключить операции проверки сомножителей на нуль (эти операции сокращают среднее время выполнения программы, когда в потоке обрабатываемых данных часто встречаются нулевые сомножители). Заметим, что сложение множимого (D) с двухбайтной суммой в регистровой паре (H, L), как и в предыдущей программе, осуществляется командой DAD D, но при этом в отличие от предыдущей ситуации может возникать переполнение СЧП. Это переполнение автоматически устраняется при сдвиге СЧП вправо. Микропроцессор КР580 не имеет команд сдвига вправо двухбайтных данных, поэтому операция такого сдвига выполняется побайтно с использованием аккумулятора.

Программа У88А представляет собой вариант «бесхитростного» программирования схемы 1, при котором МН и СЧП размещаются, согласно схеме, в структурно отдельных элементах: МН — в регистре (С), а СЧП — в регистровой паре (H, L). Более детальный анализ схемы 1 показывает, что операции сдвига вправо МН и СЧП можно совместить, если размещать выдвигаемые младшие разряды СЧП в освобождающиеся старшие разряды МН [33]. Эту «хитрость» реализует программа У88А1 (аналогичная программа приведена в работе [20]):

```

0530                                ORG      530H
                                Y88A1:
                                ;*****
                                ;ПОДПРОГРАММА УМНОЖЕНИЯ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ БЕЗ ЗНАКА
                                ;ФОРМАТА 8*8=16.ВАРИАНТ А1 (УЛУЧШЕНИЕ ВАРИАНТА А).
                                ;ВХОДНЫЕ ПАРАМЕТРЫ: (С) —МНОЖИТЕЛЬ, (Е) —МНОЖИМОЕ.ВЫХОДНОЙ
                                ;ПАРАМЕТР: (В,С) —ПРОИЗВЕДЕНИЕ.ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ,
                                ;КРОМЕ (H,L), СОХРАНЯЕТСЯ (Е).
                                ;ОЦЕНКА:ДЛИНА—19 БАЙТ,ВРЕМЯ—НЕ БОЛЕЕ 541 ТАКТА
                                ;*****
0530 AF      XRA      A
0531 47      MOV      B,A      ;ОБНУЛЕНИЕ СТБ СУММЫ ЧП
0532 1609    MVI      D,9      ;СЧЕТЧИК ЦИКЛОВ
                                ;СДВИГ МНОЖИТЕЛЯ И МЛБ СУММЫ ЧП ВПРАВО
0534 79      ЦИКЛ:   MOV      A,C
0535 1F      RAR
0536 4F      MOV      C,A
                                ;ПРОВЕРКА КОНЦА ЦИКЛА
0537 15      DCR      D
0538 C8      RZ          ;КОНЕЦ,ЕСЛИ (D)=0
0539 78      MOV      A,B      ;(A)—СТБ СУММЫ ЧП
053A D23E05 JNC      PER      ;ЕСЛИ РАЗРЯД МНОЖИТЕЛЯ=0
                                ;СЛОЖЕНИЕ МНОЖИМОГО С ТЕКУЩЕЙ СУММОЙ ЧП
053D 83      ADD      E

```

		;СДВИГ ТЕКУЩЕЙ СУММЫ ЧП ВПРАВО		
053E 1F	PER:	RAR		
053F 47		MOV	B,A	; (B) — СТВ СУММЫ ЧП
0540 C33405		JMP	ЦИКЛ	;ЗАЦИКЛИВАНИЕ
0000		END		

В программе У88А1 множитель размещен в регистре (С), СТВ СЧП — в регистре (В), а МЛБ СЧП в процессе работы программы размещается в освобождающихся разрядах регистра (С). Поскольку после 8 сдвигов МН вправо необходимо в соответствии с алгоритмом (1.20) произвести последний, 9-й сдвиг вправо СЧП, счетчик цикла перед началом программы устанавливается в регистре (D) = 9. Последний, 9-й неполный цикл используется только для сдвига вправо МЛБ СЧП в регистр (С) (9-й сдвиг СТВ СЧП происходит в конце полного 8-го цикла). Максимальная длительность полного цикла равна 62 тактам, а время программы  $T \leq 45 + 62 \cdot 8 = 541$  такту. Программа У88А1 экономичнее программы У88А по длине на 24 %, а по затратам времени на 17 % (при условии, что в программе У88А исключены операции проверки сомножителей на нуль, отсутствующие в программе У88А1).

Программа У88Б в отличие от программ У88А и У88А1 выполняет *умножение по вычислительной схеме 3* (см. рис. 1.5, в):

0550		ORG	550H
------	--	-----	------

У88Б:

```

;*****
;ПОДПРОГРАММА УМНОЖЕНИЯ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ БЕЗ ЗНАКА
;ФОРМАТА 8*8=16.ВАРИАНТ Б.
;МЕТОД УМНОЖЕНИЯ:8 ЦИКЛОВ СДВИГОВ МНОЖИТЕЛЯ И ТЕКУЩЕЙ
;СУММЫ ЧАСТИЧНЫХ ПРОИЗВЕДЕНИЙ ВЛЕВО.
;ВХОДНЫЕ ПАРАМЕТРЫ:(С) —МНОЖИТЕЛЬ,(Е) —МНОЖИМОЕ.ВЫХОДНОЙ
;ПАРАМЕТР:(Н,L) —ПРОИЗВЕДЕНИЕ.ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ,
;СОХРАНЯЮТСЯ (С),(Е).
;ОЦЕНКА:ДЛИНА-22 БАЙТА,ВРЕМЯ-НЕ БОЛЕЕ 423 ТАКТОВ.
;*****
;ОБНУЛЕНИЕ ТЕКУЩЕЙ СУММЫ ЧП

```

0550 AF	XRA	A	
0551 67	MOV	H,A	
0552 6F	MOV	L,A	
0553 57	MOV	D,A	; (D)=0

;ПРОВЕРКА СОМНОЖИТЕЛЕЙ НА 0

0554 81	ADD	C	
0555 C8	RZ		;ЕСЛИ МНОЖИТЕЛЬ=0
0556 AF	XRA	A	
0557 83	ADD	E	
0558 C8	RZ		;ЕСЛИ МНОЖИМОЕ=0
0559 61	MOV	H,C	; (H) —МНОЖИТЕЛЬ

```

055A 0608          MVI      B,8      ;СЧЕТЧИК ЦИКЛОВ
                  ;СДВИГ МНОЖИТЕЛЯ И ТЕКУЩЕЙ СУММЫ ЧП ВЛЕВО
055C 29           ЦИКЛ: DAD      H
055D D26105        JNC      ПЕР      ;ЕСЛИ РАЗРЯД МНОЖИТЕЛЯ=0
                  ;СЛОЖЕНИЕ МНОЖИМОГО С ТЕКУЩЕЙ СУММОЙ ЧП
0560 19           DAD      D
                  ;ПРОВЕРКА КОНЦА ЦИКЛА
0561 05           ПЕР:  DCR      B
0562 C25C05        JNZ      ЦИКЛ     ;ЗАЦИКЛИВАНИЕ
0565 C9           RET
0000              END

```

Программа У88Б использует тот же прием совмещения сдвигов МН и СЧП, который применен в программе У88А1. При этом МН размещается в регистре (Н), а СЧП накапливается в регистровой паре (Н, L). Максимальная длительность цикла в программе равна 45 тактам, а ее время  $T \leq 63 + 45 \cdot 8 = 423$  тактам. В программе использованы сдвиг влево МН и СЧП посредством команды двухбайтного сложения DAD H и сложение однобайтного множимого регистра (Е) с двухбайтной СЧП в регистровой паре (Н, L) посредством аналогичной команды DAD D. В последующих программах более высокого уровня широко используется упрощенный вариант программы У88Б (без проверки сомножителей на ноль) — программа У88Б1:

```

0570              ORG      570H
                  У88Б1:
;*****
;ПОДПРОГРАММА УМНОЖЕНИЯ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ БЕЗ ЗНАКА
;ФОРМАТА 8*8=16.ВАРИАНТ Б1 (БЕЗ ПРОВЕРКИ СОМНОЖИТЕЛЕЙ
;НА НОЛЬ).
;ВХОДНЫЕ ПАРАМЕТРЫ: (А) —МНОЖИТЕЛЬ, (Е) —МНОЖИМОЕ. ВЫХОДНОЙ
;ПАРАМЕТР: (Н,L) —ПРОИЗВЕДЕНИЕ. ИСПОЛЗУЮТСЯ ВСЕ РЕГИСТРЫ,
;КРОМЕ (С), СОХРАНЯЮТСЯ (А), (Е).
;ОЦЕНКА: ДЛИНА —17 БАЙТ, ВРЕМЯ — НЕ БОЛЕЕ 396 ТАКТОВ
;*****
0570 67           MOV      H,A      ; (Н) —МНОЖИТЕЛЬ
                  ;ОБНУЛЕНИЕ РЕГИСТРОВ (D), (L)
0571 1600          MVI      D,0
0573 2E00          MVI      L,0
0575 0608          MVI      B,8      ;СЧЕТЧИК ЦИКЛОВ
                  ;СДВИГ МНОЖИТЕЛЯ И ТЕКУЩЕЙ СУММЫ ЧП ВЛЕВО
0577 29           ЦИКЛ: DAD      H
0578 D27C05        JNC      ПЕР      ;ЕСЛИ РАЗРЯД МНОЖИТЕЛЯ=0
                  ;СЛОЖЕНИЕ МНОЖИМОГО С ТЕКУЩЕЙ СУММОЙ ЧП
057B 19           DAD      D
                  ;ПРОВЕРКА КОНЦА ЦИКЛА
057C 05           ПЕР:  DCR      B
057D C27705        JNZ      ЦИКЛ     ;ЗАЦИКЛИВАНИЕ
0580 C9           RET
0000              END

```



Программа У88Б1 экономичнее программы У88А1 по длине на 10 % и по затратам времени на 26 %, но использует для размещения данных лишний регистр. Быстродействие программы можно увеличить (с 396 до 257 тактов), если раскрыть цикл, т. е. исключить команды проверки конца цикла, а сам цикл повторить 8 раз путем 8-кратного размещения команд цикла в теле программы [18]. Сравнение вычислительных схем 1 и 3 по их программным реализациям показывает предпочтительность использования схемы 3. Ее преимущество объясняется наличием у микропроцессора КР580 команд сложения типа DAD, обеспечивающих быстрое сложение двухбайтных данных и их сдвиг влево. Из-за отсутствия команд сдвига вправо двухбайтных данных малоперспективна схема 4.

Программа У88В реализует *умножение по вычислительной схеме 2* (см. рис. 1.5, б) (аналогичная программа приведена в работе [16]):

```

0590                                ORG      590H
                                У88В:
;*****
;ПОДПРОГРАММА УМНОЖЕНИЯ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ БЕЗ ЗНАКА
;ФОРМАТА 8*8=16.ВАРИАНТ В.
;МЕТОД УМНОЖЕНИЯ:8 (ИЛИ МЕНЕЕ) ЦИКЛОВ СДВИГОВ МНОЖИТЕЛЯ
;ВПРАВО (С КОНТРОЛЕМ ОСТАТКА НА 0),А МНОЖИМОГО ВЛЕВО.
;ВХОДНЫЕ ПАРАМЕТРЫ:(С) -МНОЖИТЕЛЬ,(Е) -МНОЖИМОЕ.ВЫХОДНОЙ
;ПАРАМЕТР:(Н,Л) -ПРОИЗВЕДЕНИЕ.ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ,
;КРОМЕ В;СОХРАНЯЕТСЯ (С).
;ОЦЕНКА:ДЛИНА-22 БАЙТА,ВРЕМЯ-НЕ БОЛЕЕ 499 ТАКТОВ.
;*****
;ОБНУЛЕНИЕ ТЕКУЩЕЙ СУММЫ ЧП
0590 AF      XRA      A
0591 67      MOV      H,A
0592 6F      MOV      L,A
0593 57      MOV      D,A      ; (D)=0
;ПРОВЕРКА СОМНОЖИТЕЛЕЙ НА 0
0594 83      ADD      E
0595 C8      RZ          ;ЕСЛИ МНОЖИМОЕ=0
0596 AF      XRA      A
0597 81      ADD      C      ; (A) -МНОЖИТЕЛЬ
0598 C8      RZ          ;ЕСЛИ МНОЖИТЕЛЬ=0
;СДВИГ МНОЖИТЕЛЯ ВПРАВО
0599 1F      ЦИКЛ: RAR
059A D29E05  JNC      PER      ;ЕСЛИ РАЗРЯД МНОЖИТЕЛЯ=0
;СЛОЖЕНИЕ МНОЖИМОГО С ТЕКУЩЕЙ СУММОЙ ЧП
059D 19      DAD      D
;СДВИГ МНОЖИМОГО ВЛЕВО
059E EB      PER: XCHG
059F 29      DAD      H
05A0 EB      XCHG

```

05A1 B7	ORA	A	
05A2 C29905	JNZ	ЦИКЛ	;ЗАЦИКЛИВАНИЕ
05A5 C9	RET		
0000	END		

Особенность этой программы заключается в том, что МН постоянно размещен (и сдвигается) в аккумуляторе, причем конец цикла контролируется не по счетчику (счетчик цикла отсутствует), как в предыдущих программах, а по остатку МН в аккумуляторе. Выполнение же операций сложения и сдвига осуществляется, как и в программах У88Б и У88Б1, посредством команд типа DAD, минуя аккумулятор, причем для сдвига МН влево временно используется та же регистровая пара (H,L), в которой размещается СЧП. Программа У88В по быстродействию ( $T \leq 51 + 56 \cdot n$  тактов, где  $n \leq 8$ ) превосходит программы У88А и У88А1, а в ряде случаев и программы У88Б, У88Б1, поскольку они всегда выполняют 8 циклов умножения, независимо от значения ненулевого МН, а программа У88В при нулевых старших разрядах МН — на соответствующее количество циклов меньше.

Проверить работоспособность программ полезно на тестовых наборах данных (табл. 1.5).

Табл. 1.5. Тесты умножения формата 8·8=16  
(целые двоичные беззнаковые числа)

Представление чисел	
шестнадцатеричное	десятичное
FF·FF=FE01	255·255=65025
FF·01=00FF	255·1=255
F0·0F=0E10	240·15=3600
55·AA=3872	85·170=14450
00·00=0000	0·0=0
10·10=100	16·16=256
20·20=400	32·32=1024
40·40=1000	64·64=4096
80·80=4000	28·128=16384
7F·7F=3FO1	127·127=16129

### 1.3.2.2. Форматы 8·16=24, 8·16=16

Программа У24 реализует умножение по вычислительной схеме 3 аналогично программе У88Б:

05B0

ORG 5B0H

Y24:

```

;*****
;ПОДПРОГРАММА УМНОЖЕНИЯ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ БЕЗ ЗНАКА
;ФОРМАТА 16*8=24.
;МЕТОД УМНОЖЕНИЯ:8 ЦИКЛОВ СДВИГОВ МНОЖИТЕЛЯ И ТЕКУЩЕЙ
;СУММЫ ЧАСТИЧНЫХ ПРОИЗВЕДЕНИЙ ВЛЕВО.
;ВХОДНЫЕ ПАРАМЕТРЫ:(С)-МНОЖИТЕЛЬ,(D,E)-МНОЖИМОЕ,ВЫХОД-
;НЫЕ ПАРАМЕТРЫ:(A,H,L)-ПРОИЗВЕДЕНИЕ.ИСПОЛЬЗУЮТСЯ ВСЕ
;РЕГИСТРЫ,СОХРАНЯЮТСЯ (C),(D,E).
;ОЦЕНКА:ДЛИНА-25 БАЙТ,ВРЕМЯ-НЕ БОЛЕЕ 510 ТАКТОВ.
;*****
;ОБНУЛЕНИЕ ТЕКУЩЕЙ СУММЫ ЧП

```

```

05B0 AF      XRA      A
05B1 67      MOV      H,A
05B2 6F      MOV      L,A

;ПРОВЕРКА СМНОЖИТЕЛЕЙ НА 0
05B3 81      ADD      C
05B4 C8      RZ        ;ЕСЛИ МНОЖИТЕЛЬ=0
05B5 AF      XRA      A
05B6 B2      ORA      D
05B7 B3      ORA      E
05B8 C8      RZ        ;ЕСЛИ МНОЖИМОЕ=0
05B9 79      MOV      A,C ;(A)-МНОЖИТЕЛЬ
05BA 060B    MVI      B,8 ;СЧЕТЧИК ЦИКЛОВ

;СДВИГ ТЕКУЩЕЙ СУММЫ ЧП И МНОЖИТЕЛЯ ВЛЕВО
05BC 29      DAD      H   ЦИКЛ:
05BD 17      RAL
05BE D2C405  JNC      PER ;ЕСЛИ РАЗРЯД МНОЖИТЕЛЯ=0

;СЛОЖЕНИЕ МНОЖИМОГО С ТЕКУЩЕЙ СУММОЙ ЧП
05C1 19      DAD      D
05C2 CE00    ACI      0

;ПРОВЕРКА КОНЦА ЦИКЛА
05C4 05      PER:     DCR      B
05C5 C2BC05  JNZ      CYCL ;ЗАЦИКЛИВАНИЕ
05C8 C9      RET
0000      END

```

Максимальная длительность цикла умножения составляет 56 тактов, а время выполнения программы  $T \leq 62 + 56 \cdot 8 = 510$  тактам. В последующих программах более высокого уровня широко используется ее упрощенный вариант (без проверки сомножителей на ноль) — программа Y24A (аналогичная программа приведена в работе [21]):

05D0

ORG 5D0H

Y24A:

```

;*****
;ПОДПРОГРАММА УМНОЖЕНИЯ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ БЕЗ ЗНАКА
;ФОРМАТА 16*8=24 (БЕЗ ПРОВЕРКИ СМНОЖИТЕЛЕЙ НА НОЛЬ).
;ВХОДНЫЕ ПАРАМЕТРЫ:(A)-МНОЖИТЕЛЬ,(D,E)-МНОЖИМОЕ,ВЫХОД-
;НЫЕ ПАРАМЕТРЫ:(A,H,L)-ПРОИЗВЕДЕНИЕ.ИСПОЛЬЗУЮТСЯ ВСЕ
;РЕГИСТРЫ,СОХРАНЯЕТСЯ (D,E).
;ОЦЕНКА:ДЛИНА-18 БАЙТ,ВРЕМЯ-НЕ БОЛЕЕ 454 ТАКТА.
;*****

```

05D0 210000	LXI	H,0	;ОБНУЛЕНИЕ СУММЫ ЧП
05D3 010800	LXI	B,08	; (B)=0, (C) - СЧЕТЧИК ЦИКЛОВ
	;СДВИГ ТЕКУЩЕЙ СУММЫ ЧП И МНОЖИТЕЛЯ ВЛЕВО		
05D6 29	DAD	H	
05D7 8F	ADC	A	
05D8 D2DD05	JNC	PER	;ЕСЛИ РАЗРЯД МНОЖИТЕЛЯ=0
	;СЛОЖЕНИЕ МНОЖИМОГО С ТЕКУЩЕЙ СУММОЙ ЧП		
05DB 19	DAD	D	
05DC 88	ADC	B	
	;ПРОВЕРКА КОНЦА ЦИКЛА		
05DD 0D	PER:	DCR	C
05DE C2D605	JNZ	ЦИКЛ	;ЗАЦИКЛИВАНИЕ
05E1 C9	RET		
0000	END		

Максимальная длительность цикла умножения равна 53 тактам, а время выполнения программы  $T \leq 30 + 53 \cdot 8 = 454$  тактам. Сокращение времени цикла по сравнению с программой У24 достигнуто за счет использования более быстродействующей команды ADC В вместо ACI 0.

Иногда при умножении целых чисел возникает необходимость получения *произведения в сокращенном формате*, не превышающем формат одного из сомножителей. Обычно такое произведение формируют из полноформатного путем его округления. Программа У168 реализует сокращенный формат умножения путем симметричного округления произведения формата  $16 \cdot 8 = 24$ , вычисляемого предварительно подпрограммой У24:

05F0		ORG	5F0H
05B0	У24	SET	5B0H

У168:

```

;*****
;ПОДПРОГРАММА УМНОЖЕНИЯ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ БЕЗ ЗНАКА
;ФОРМАТА 16*8=16 С ОКРУГЛЕНИЕМ.
;ВХОДНЫЕ ПАРАМЕТРЫ: (C) - МНОЖИТЕЛЬ. (D,E) - МНОЖИМОЕ. ВЫХОД-
;НОЙ ПАРАМЕТР: (H,L) - ПРОИЗВЕДЕНИЕ. ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИ-
;СТРЫ, СОХРАНЯЮТСЯ (C), (D,E). ИСПОЛЬЗУЕТСЯ ПОДПРОГРАММА
;*У24*.
;ОЦЕНКА: ДЛИНА - 11 (+25*У24*) БАЙТ, ВРЕМЯ - НЕ БОЛЕЕ 571
;ТАКОВ (С УЧЕТОМ *У24*).
;*****
;ВЫПОЛНЕНИЕ УМНОЖЕНИЯ 16*8=24
05F0 CDB005      CALL    У24      ; (A,H,L) - ПРОИЗВЕДЕНИЕ
;ОКРУГЛЕНИЕ РЕЗУЛЬТАТА
05F3 45          MOV     B,L      ; (B) - МЛБ
05F4 6C          MOV     L,H      ; (L) - СРБ
05F5 67          MOV     H,A      ; (H) - СТБ
05F6 78          MOV     A,B      ; (A) - МЛБ
05F7 17          RAL             ;
05F8 D0          RNC             ;ЕСЛИ СТР БАЙТА=0

```

05F9 23	INX	H	; (СТБ, СРБ) +1
05FA C9	RET		
0000	END		

В результате округления точного произведения формируется произведение ограниченной точности, погрешность которого определяется формулами (1.10) и (1.12). Заметим, что модуль полученного произведения содержит неявный множитель 256, так как сокращенный формат представляет только СТБ и СРБ полного произведения (МЛБ отбрасывается). Тестовые наборы данных для программ У24 и У24А приведены в табл. 1.6.

**Табл. 1.6. Тесты умножения формата 8·16=24  
(целые двоичные беззнаковые числа)**

Представление чисел	
шестнадцатеричное	десятичное
FF·FFFF=FEFF01	1 255·65535 = 16711425
01·FFFF = 00FFFF	01·65535 = 65535
0F·F000 = 0E1000	15·64440 = 966600
FF·FE00 = FE0100	255·65280 = 16646400
AA·5555=38AA72	170·21845 = 31713650
7F·07FF=03F781	127·2047 = 259969

### 1.3.2.3. Формат 16·16=32

Программа У32А, подобно программам У88Б, У24, реализует умножение по вычислительной схеме 3 (аналогичная программа приведена в работе [71]):

```

0600                                ORG      600H
                                У32А:
;*****
;ПОДПРОГРАММА УМНОЖЕНИЯ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ БЕЗ ЗНАКА
;ФОРМАТА 16*16=32.ВАРИАНТ А.
;МЕТОД УМНОЖЕНИЯ:16 ЦИКЛОВ СДВИГОВ МНОЖИТЕЛЯ И СУММЫ
;ЧАСТИЧНЫХ ПРОИЗВЕДЕНИЙ ВЛЕВО.
;ВХОДНЫЕ ПАРАМЕТРЫ: (В,С) —МНОЖИМОЕ, (D,E) —МНОЖИТЕЛЬ. ВЫ-
;ХОДНЫЕ ПАРАМЕТРЫ: (D,E,H,L) —ПРОИЗВЕДЕНИЕ. ИСПОЛЬЗУЮТСЯ
;ВСЕ РЕГИСТРЫ, СОХРАНЯЕТСЯ (В,С).
;ОЦЕНКА: ДЛИНА-40 БАЙТ, ВРЕМЯ-НЕ БОЛЕЕ 1711 ТАКТОВ.
;*****
;ОБНУЛЕНИЕ ТЕКУЩЕЙ СУММЫ ЧП
0600 AF      XRA      A
0601 67      MOV     H,A
0602 6F      MOV     L,A
;ПРОВЕРКА СОМНОЖИТЕЛЕЙ НА 0
0603 B2      ORA     D

```

0604 B3	ORA	E	
0605 C8	RZ		;ЕСЛИ МНОЖИТЕЛЬ=0
0606 AF	XRA	A	
0607 B0	ORA	B	
0608 B1	ORA	C	
0609 C20F06	JNZ	ПЕР	;ЕСЛИ МНОЖИМОЕ НЕ 0
060C 57	MOV	D, A	;ОБНУЛЕНИЕ МНОЖИТЕЛЯ
060D 5F	MOV	E, A	
060E C9	RET		
060F AF	ПЕР:	XRA	A ;СЧЕТЧИК ЦИКЛОВ=0
	;СДВИГ	МНОЖИТЕЛЯ	ВЛЕВО
0610 EB	ЦИКЛ:	XCHG	
0611 29	DAD	H	
0612 EB	XCHG		
0613 1F	RAR		;СОХРАНЕНИЕ ПЕРЕНОСА В (A)
	;СДВИГ	СУММЫ	ЧП ВЛЕВО
0614 29	DAD	H	
0615 D21906	JNC	ПЕР1	
0618 13	INX	D	;УЧЕТ ПЕРЕНОСА ОТ СДВИГА
	;АНАЛИЗ	СДВИНУТОГО	РАЗРЯДА МНОЖИТЕЛЯ
0619 17	ПЕР1:	RAL	;ВОССТАНОВЛЕНИЕ ПЕРЕНОСА
061A D22206	JNC	ПЕР2	;ЕСЛИ РАЗРЯД МНОЖИТЕЛЯ=0
	;СЛОЖЕНИЕ	МНОЖИМОГО	С ТЕКУЩЕЙ СУММОЙ ЧП
061D 09	DAD	B	
061E D22206	JNC	ПЕР2	
0621 13	INX	D	;УЧЕТ ПЕРЕНОСА ОТ СЛОЖЕНИЯ
	;ПРОВЕРКА	КОНЦА	ЦИКЛА ПО ПЕРЕПОЛНЕНИЮ (A)
0622 C610	ПЕР2:	ADI	16
0624 D21006	JNC	ЦИКЛ	;ЗАЦИКЛИВАНИЕ
0627 C9	RET		
0000	END		

Увеличение разрядности сомножителей и произведения ведет к росту количества используемых регистров и усложнению типа передач данных между ними, что в результате увеличивает длину программы, длительность ее цикла и общее время ее выполнения. Максимальная длительность цикла умножения в программе У32А равна 103 тактам, а время ее выполнения  $T \leq 63 + 103 \cdot 16 = 1711$  тактам. Программа имеет три особенности: 1) для сдвига МН временно используется та же регистровая пара (Н, L), что и для накопления СЧП; 2) для сохранения переноса от сдвига множителя (с целью последующего его анализа) используется старший бит аккумулятора; 3) счетчик конца цикла построен не по принципу уменьшения переменной цикла до нуля, как в предыдущих программах, а, наоборот, по принципу увеличения переменной цикла от нуля до переполнения счетчика (аккумулятора).

Программа У32А выполняет процесс умножения как 16-кратное повторение цикла умножения множимого на очередной разряд множителя. Благодаря этому обеспе-

чиваются регулярная структура программы и минимальные затраты памяти, но время выполнения программы велико. Его можно сократить, если организовать процесс вычисления произведения формата  $16 \cdot 16 = 32$  как последовательность двух подпроцессов умножения форматов  $16 \cdot 8 = 24$  для МЛБ МН и  $16 \cdot 8 = 24$  для СТБ МН с последующим суммированием полученных произведений с учетом их сдвига на один байт относительно друг друга. Этот алгоритм реализует программа УЗ2Б (аналогичная программа приведена в работе [21]):

0630		ORG	630H
05D0	Y24A	SET	5D0H

УЗ2Б:

```

;*****
;ПОДПРОГРАММА УМНОЖЕНИЯ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ БЕЗ ЗНАКА
;ФОРМАТА 16*16=32.ВАРИАНТ Б.
;МЕТОД УМНОЖЕНИЯ:РЕЗУЛЬТАТ 16*16=32 ОПРЕДЕЛЯЕТСЯ КАК
;СУММА ДВУХ СДВИНУТЫХ ОТНОСИТЕЛЬНО ДРУГ ДРУГА ПРОИЗВЕ-
;ДЕНИЙ 16*8=24,ПОЛУЧЕННЫХ УМНОЖЕНИЕМ МНОЖИМОГО СООТВЕТ-
;СТВЕННО НА МЛБ И СТБ МНОЖИТЕЛЯ.
;ВХОДНЫЕ ПАРАМЕТРЫ:(В,С)-МНОЖИТЕЛЬ,(D,E)-МНОЖИМОЕ.ВЫ-
;ХОДНЫЕ ПАРАМЕТРЫ:(В,С,H,L)-ПРОИЗВЕДЕНИЕ.ИСПОЛЬЗУЮТСЯ
;ВСЕ РЕГИСТРЫ.ГЛУБИНА СТЕКА-4.ИСПОЛЬЗУЕТСЯ ПОДПРОГРАММА
;*Y24A*.
;ОЦЕНКА:ДЛИНА 42 (+18 *Y24A*) БАЙТ,ВРЕМЯ- НЕ БОЛЕЕ 1127
;ТАКОВ (С УЧЕТОМ *Y24A*).
;*****
;ОБНУЛЕНИЕ ТЕКУЩЕЙ СУММЫ ЧП
0630 AF      XRA      A
0631 67      MOV      H,A
0632 6F      MOV      L,A
;ПРОВЕРКА МНОЖИМОГО НА 0
0633 B2      ORA      D
0634 B3      ORA      E
0635 C23B06  JNZ      PER1      ;ЕСЛИ МНОЖИМОЕ НЕ 0
0638 47      MOV      B,A
0639 4F      MOV      C,A
063A C9      RET
;ПРОВЕРКА МНОЖИТЕЛЯ НА 0
063B AF      PER1: XRA      A
063C B0      ORA      B
063D B1      ORA      C
063E C8      RZ              ;ЕСЛИ МНОЖИТЕЛЬ=0
;УМНОЖЕНИЕ МНОЖИМОГО НА МЛБ МНОЖИТЕЛЯ
063F 79      MOV      A,C      ;(A)-МЛБ МНОЖИТЕЛЯ
0640 C5      PUSH     B      ;СОХРАНЕНИЕ МНОЖИТЕЛЯ
0641 CDD005  CALL     Y24A      ;(A,H,L)-ПРОИЗВЕДЕНИЕ 1 (P1)
;СОХРАНЕНИЕ ПРОИЗВЕДЕНИЯ 1
0644 E3      XTHL          ;СОХРАНЕНИЕ СРБ,МЛБ P1
; (H,L)-МНОЖИТЕЛЬ
0645 F5      PUSH     FSW      ;СОХРАНЕНИЕ СТБ P1
;УМНОЖЕНИЕ МНОЖИМОГО НА СТБ МНОЖИТЕЛЯ
0646 7C      MOV      A,H      ;(A)-СТБ МНОЖИТЕЛЯ

```

```

0647 CDD005      CALL    Y24A.      ; (A,H,L) - ПРОИЗВЕДЕНИЕ 2 (ПР2)
                  ; СЛОЖЕНИЕ СДВИНУТЫХ НА 8 РАЗРЯДОВ ПР1 И ПР2
064A 47          MOV     B,A         ; (B) - СТВ ПР2
064B F1          POP     PSW         ; (A) - СТВ ПР1
064C 84          ADD     H
064D 4F          MOV     C,A         ; (C) - СУММА СТВ ПР1 И СРБ ПР2
064E D25206      JNC     ПЕР2
0651 04          INR     B           ; УЧЕТ ПЕРЕНОСА В СТВ
0652 65          ПЕР2: MOV     H,L     ; (H) - МЛБ ПР2
0653 2E00          MVI     L,0
0655 D1          POP     D           ; (D,E) - СРБ, МЛБ ПР1
0656 19          DAD     D
0657 D0          RNC
0658 03          INX     B           ; УЧЕТ ПЕРЕНОСА В СТВ
0659 C9          RET
0000            END

```

В программе для получения *промежуточных произведений* формата  $16 \cdot 8 = 24$  используется быстродействующая подпрограмма Y24A. В результате программа Y32B на 34 % снижает время умножения по сравнению с программой Y32A, но на 50 % увеличивает затраты памяти (с учетом длины подпрограммы Y24A). В программе Y32B для временного хранения первого промежуточного произведения (из-за нехватки свободных регистров) использована область стека, причем для записи СРБ ПР1 и МЛБ ПР1 из регистровой пары (H, L) в стек использована команда XTHL обмена вершины стека с регистровой парой (H, L). Действие этой команды поясняется на рис. 1.6. Указатель стека (SP) перед выполнением

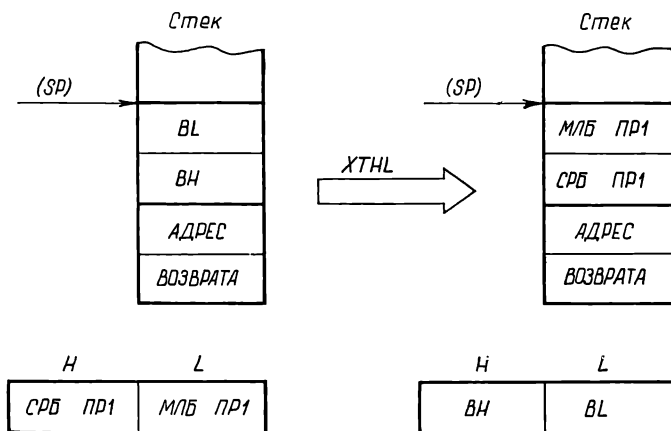


Рис. 1.6. Действие команды XTHL



команды XTHL указывает на ячейку стека, в которую предыдущей командой PUSH B записан младший байт BL регистровой пары (B, C). При выполнении команды XTHL содержимое вершины стека по адресам (SP) и (SP) + 1 переписывается в регистровую пару (H, L), а содержимое регистровой пары (H, L) — на место BH, BL (BH — старший байт регистровой пары (B, C)). Глубина стека в программе равна 4 байтам. Тестовые данные для программ У32А, У32Б приведены в табл. 1.7.

**Табл. 1.7. Тесты умножения формата 16·16=32  
(целые двоичные беззнаковые числа)**

Представление чисел	
шестнадцатеричное	десятичное
FFFF·FFFF=FFFE0001	65535 · 65535 = 4294836225
FFFF·0001=0000FFFF	65535 · 1 = 65535
F000·000F=000E1000	61440 · 15 = 921600
FF00·00FF=00FE010	65280 · 255 = 16646400
5555·AAAA=38E31C72	211845 · 43690 = 954408050
7FFF·7FFF=3FFF0001	32767 · 32767 = 1073676289

### 1.3.3. ЦЕЛЫЕ ЧИСЛА СО ЗНАКОМ

#### 1.3.3.1. Формат 8·8=16

Программа УД88 выполняет умножение, обращаясь к подпрограмме У88Б:

```

0660                                ORG      660H
0550                                SET      550H

УД88:
; *****
; ПОДПРОГРАММА УМНОЖЕНИЯ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ В ДОПОЛНИ-
; ТЕЛЬНОМ КОДЕ ФОРМАТА 8*8=16.
; ВХОДНЫЕ ПАРАМЕТРЫ: (C) — МНОЖИТЕЛЬ, (E) — МНОЖИМОЕ. ВЫХОДНОЕ
; ПАРАМЕТР: (H,L) — ПРОИЗВЕДЕНИЕ. ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ,
; СОХРАНЯЮТСЯ (C), (E). ИСПОЛЬЗУЕТСЯ ПОДПРОГРАММА *У88Б*.
; ОЦЕНКА: ДЛИНА — 22 (+22 *У88Б*) БАЙТ, ВРЕМЯ — НЕ БОЛЕЕ 528
; ТАКТОВ (С УЧЕТОМ *У88Б*).
; *****
; ВЫПОЛНЕНИЕ БЕЗЗНАКОВОГО УМНОЖЕНИЯ 8*8=16
0660 CD5005      CALL      У88Б      ; (H,L) — ПРОИЗВЕДЕНИЕ
; ПРОВЕРКА ПРОИЗВЕДЕНИЯ НА 0
0663 AF         XRA      A
0664 B4         ORA      H
0665 B5         ORA      L
0666 C8         RZ          ; ЕСЛИ ПРОИЗВЕДЕНИЕ=0

```

```

;ПРОВЕРКА ЗНАКА МНОЖИМОГО
0667 7B      MOV     A,E
0668 17      RAL
0669 D26F06  JNC     ПЕР1      ;ЕСЛИ ЗНАК "+"
;ВЫЧИТАНИЕ МНОЖИТЕЛЯ ИЗ СТЬ ПРОИЗВЕДЕНИЯ
066C 7C      MOV     A,H
066D 91      SUB     C
066E 67      MOV     H,A
;ПРОВЕРКА ЗНАКА МНОЖИТЕЛЯ
066F 79      ПЕР1: MOV     A,C
0670 17      RAL
0671 D0      RNC             ;ЕСЛИ ЗНАК "+"
;ВЫЧИТАНИЕ МНОЖИМОГО ИЗ СТЬ ПРОИЗВЕДЕНИЯ
0672 7C      MOV     A,H
0673 93      SUB     E
0674 67      MOV     H,A
0675 C9      RET
0000      END

```

Вместо подпрограммы У88Б можно использовать подпрограмму У88В1, если предварительно передать МН из регистра (С) в аккумулятор и не ставить задачу сохранения МН после выполнения УД88. Глубина стека в программе равна 2 байтам. Тестовые данные для программы УД88 приведены в табл. 1.8.

**Табл. 1.8. Тесты умножения формата 8·8=16  
(целые двоичные числа в дополнительном коде)**

Представление чисел	
шестнадцатеричное	десятичное
FF·FF=0001	$(-1) \cdot (-1) = +1$
FF·01=FFFF	$(-1) \cdot (+1) = -1$
F0·0F=FF10	$(-16) \cdot (+15) = -240$
55·AA=E372	$(+85) \cdot (-86) = -7310$
80·80=4000	$(-128) \cdot (-128) = +16384$
7F·7F=3F01	$(+127) \cdot (+127) = +16129$

### 1.3.3.2. Формат 8 · 16 = 24

Программа УД24 выполняет умножение, обращаясь к подпрограмме У24:

```

0680      ORG     680H
0580      Y24     SET     5B0H

УД24:
;*****
;ПОДПРОГРАММА УМНОЖЕНИЯ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ В ДОПОЛ-
;НИТЕЛЬНОМ КОДЕ ФОРМАТА 16*8=24.
;ВХОДНЫЕ ПАРАМЕТРЫ: (С) -МНОЖИТЕЛЬ, (D,E) -МНОЖИМОЕ. ВЫХОД-
;НЫЕ ПАРАМЕТРЫ: (B,H,L) -ПРОИЗВЕДЕНИЕ. ИСПОЛЬЗУЮТСЯ ВСЕ

```

```

;РЕГИСТРЫ, СОХРАНЯЮТСЯ (C), (D, E). ИСПОЛЬЗУЕТСЯ ПОДПРОГ-
;РАММА *У24*.
;ОЦЕНКА: ДЛИНА-25 (+25 *У24*) БАЙТ, ВРЕМЯ-НЕ БОЛЕЕ 630
;ТАКТОВ (С УЧЕТОМ *У24*).
;*****
;ВЫПОЛНЕНИЕ БЕЗЗНАКОВОГО УМНОЖЕНИЯ 16*8=24
0680 CDB005      CALL    У24      ; (А,Н,Л) -ПРОИЗВЕДЕНИЕ
0683 47          MOV     В,А      ; (В) -СТБ
;ПРОВЕРКА ПРОИЗВЕДЕНИЯ НА 0
0684 B4          ORA     Н
0685 B5          ORA     Л
0686 C8          RZ              ;ЕСЛИ ПРОИЗВЕДЕНИЕ=0
;ПРОВЕРКА ЗНАКА МНОЖИТЕЛЯ
0687 79          MOV     А,С
0688 17          RAL
0689 D29206      JNC     ПЕР1      ;ЕСЛИ ЗНАК "+"
;ВЫЧИТАНИЕ МНОЖИМОГО ИЗ СТАРШИХ БАЙТОВ ПРОИЗВЕДЕНИЯ
068C 7C          MOV     А,Н
068D 93          SUB     Е
068E 67          MOV     Н,А
068F 78          MOV     А,В
0690 9A          SBB     Д
0691 47          MOV     В,А
;ПРОВЕРКА ЗНАКА МНОЖИМОГО
0692 7A          ПЕР1: MOV   А,Д
0693 17          RAL
0694 D0          RNC              ;ЕСЛИ ЗНАК "+"
;ВЫЧИТАНИЕ МНОЖИТЕЛЯ ИЗ СТБ ПРОИЗВЕДЕНИЯ
0695 78          MOV     А,В
0696 91          SUB     С
0697 47          MOV     В,А
0698 C9          RET
0000            END

```

Глубина стека — 2. Тестовые данные для программы УД24 приведены в табл. 1.9.

Табл. 1.9. Тесты умножения формата 8·16=24  
(целые двоичные числа в дополнительном коде)

Представление чисел	
шестнадцатеричное	десятичное
FF·FFFF=000 001	(-1)·(-1)=+1
01·FFFF=FFFFFF	(+1)·(-1)=-1
0F·F000=FF1000	(+15)·(-4096)=-61440
FF·FF00=000100	(-1)·(-256)=+256
AA·5555=E35 572	(-86)·(+21845)=-1878670
7F·07FF=03F781	(+127)·(+2047)=+259969

### 1.3.3.3. Формат 16 · 16 = 32

Программа УД32 выполняет умножение, обращаясь к подпрограмме У32Б:

06D0		ORG	6D0H
0630	У32Б	SET	630H

УД32:

```

;*****
;ПОДПРОГРАММА УМНОЖЕНИЯ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ В ДОПОЛ-
;НИТЕЛЬНОМ КОДЕ ФОРМАТА 16*16=32.
;ВХОДНЫЕ ПАРАМЕТРЫ: (В,С)–МНОЖИТЕЛЬ, (D,E)–МНОЖИМОЕ. ВЫ-
;ХОДНЫЕ ПАРАМЕТРЫ: (В,С,Н,L)–ПРОИЗВЕДЕНИЕ. ИСПОЛЬЗУЮТСЯ
;ВСЕ РЕГИСТРЫ, СОХРАНЯЕТСЯ (D,E). ГЛУБИНА СТЕКА–10. ИС-
;ПОЛЬЗУЕТСЯ ПОДПРОГРАММА *У32Б*.
;ОЦЕНКА: ДЛИНА–41 ((+42 *У32Б* )+18 *У24А* ), ВРЕМЯ–НЕ
;БОЛЕЕ 1340 ТАКТОВ (С УЧЕТОМ *У32Б* ).
;*****
;ВЫПОЛНЕНИЕ БЕЗЗНАКОВОГО УМНОЖЕНИЯ 16*16=32
06D0 C5      PUSH    B      ;СОХРАНЕНИЕ МНОЖИТЕЛЯ
06D1 D5      PUSH    D      ;СОХРАНЕНИЕ МНОЖИМОГО
06D2 CD3006  CALL    Y32Б   ;(В,С,Н,L)–ПРОИЗВЕДЕНИЕ
06D5 D1      POP     D      ;ВОССТАНОВЛЕНИЕ МНОЖИМОГО
;ПРОВЕРКА ПРОИЗВЕДЕНИЯ НА 0
06D6 AF      XRA     A
06D7 B0      ORA     B
06D8 B1      ORA     C
06D9 B4      ORA     H
06DA B5      ORA     L
06DB C2E006  JNZ     PER     ;ЕСЛИ ПРОИЗВЕДЕНИЕ НЕ 0
06DE F1      POP     PSW    ;БАЛАНС СТЕКА
06DF C9      RET
;СОХРАНЕНИЕ МЛАДШИХ БАЙТОВ ПРОИЗВЕДЕНИЯ
;ВОССТАНОВЛЕНИЕ МНОЖИТЕЛЯ
06E0 E3      PER:  XTHL
;ПРОВЕРКА ЗНАКА МНОЖИМОГО
06E1 7A      MOV     A,D
06E2 17      RAL
06E3 D2EC06  JNC     PER1    ;ЕСЛИ ЗНАК "+"
;ВЫЧИТАНИЕ МНОЖИТЕЛЯ ИЗ СТАРШИХ БАЙТОВ ПРОИЗВЕДЕНИЯ
06E6 79      MOV     A,C
06E7 95      SUB     L
06E8 4F      MOV     C,A
06E9 78      MOV     A,B
06EA 9C      SBB     H
06EB 47      MOV     B,A
;ПРОВЕРКА ЗНАКА МНОЖИТЕЛЯ
06EC 7C      PER1: MOV     A,H
06ED 17      RAL
06EE D2F706  JNC     PER2    ;ЕСЛИ ЗНАК "+"
;ВЫЧИТАНИЕ МНОЖИМОГО ИЗ СТАРШИХ БАЙТОВ ПРОИЗВЕДЕНИЯ
06F1 79      MOV     A,C
06F2 93      SUB     E
06F3 4F      MOV     C,A
06F4 78      MOV     A,B
06F5 9A      SBB     D

```

06F6 47	MOV	B,A	
06F7 E1	PER2: POP	H	;ВОССТАНОВЛЕНИЕ МЛАДШИХ
06F8 C9	RET		;БАЙТОВ ПРОИЗВЕДЕНИЯ
0000	END		

Тестовые данные для программы УД32 приведены в табл. 1.10.

**Табл. 1.10. Тесты умножения формата  $16 \cdot 16 = 32$  (целые двоичные числа в дополнительном коде)**

Представление чисел	
шестнадцатеричное	десятичное
FFFF·FFFF=00000001	$(-1) \cdot (-1) = +1$
FFFF·0001=FFFFFFFF	$(-1) \cdot (+1) = -1$
F000·000F=FFFF1000	$(-4096) \cdot (+15) = -61440$
FF00·00FF=FFFF0100	$(-256) \cdot (+255) = -65280$
5555·AAAA=E38E1C72	$(+21845) \cdot (-21846) = -477225870$
7FFF·7FFF=3FFF0001	$(+32767) \cdot (+32767) = +1073676289$

#### 1.3.3.4. Формат $8 \cdot 24 = 32$

Программа УД248 реализует умножение чисел формата  $8 \cdot 24 = 32$  как сумму произведений МН соответственно на МЛБ и СРБ ММ формата  $8 \cdot 16 = 24$  и СТБ ММ формата  $8 \cdot 8 = 16$ , используя быстродействующие подпрограммы У24А и У88Б1 и операцию вычитания поправок из псевдопроизведения:

0700		ORG	700H
05D0	Y24A	SET	5D0H
0570	Y88B1	SET	570H

**УД248:**

```

;*****
;ПОДПРОГРАММА УМНОЖЕНИЯ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ В ДОПОЛ-
;НИТЕЛЬНОМ КОДЕ ФОРМАТА 24*8=32.
;МЕТОД УМНОЖЕНИЯ:РЕЗУЛЬТАТ 24*8=32 ОПРЕДЕЛЯЕТСЯ КАК
;СУММА ДВУХ СДВИНУТЫХ ОТНОСИТЕЛЬНО ДРУГ ДРУГА ПРОИЗ-
;ВЕДЕНИИ ФОРМАТА 16*8=24 И 8*8=16,ПОЛУЧЕННЫХ УМНОЖЕ-
;НИЕМ МНОЖИТЕЛЯ СООТВЕТСТВЕННО НА МЛАДШИЕ И СТАРШИЙ
;БАЙТЫ МНОЖИМОГО.КАЖДОЕ ПРОИЗВЕДЕНИЕ ФОРМИРУЕТСЯ ЗА
;8 ЦИКЛОВ СДВИГА МНОЖИТЕЛЯ И СУММЫ ЧАСТИЧНЫХ ПРОИЗ-
;ВЕДЕНИИ ВЛЕВО.
;ВХОДНЫЕ ПАРАМЕТРЫ:(С,D,E)-МНОЖИМОЕ,(A)-МНОЖИТЕЛЬ.
;ВЫХОДНЫЕ ПАРАМЕТРЫ:(B,C,H,L)-ПРОИЗВЕДЕНИЕ.ИСПОЛЬ-
;ЗУЮТСЯ ВСЕ РЕГИСТРЫ.ГЛУБИНА СТЕКА 12.ИСПОЛЬЗУЮТСЯ
;ПОДПРОГРАММЫ *У24А*.*У88Б1*.
;ОЦЕНКА:ДЛИНА- 70 (+18 *У24А* + 17 *У88Б1*) БАЙТА,
;ВРЕМЯ-НЕ БОЛЕЕ 1264 ТАКТА (С УЧЕТОМ ПОДПРОГРАММ).
;*****

```

```

0700 F5      PUSH    PSW      ;СОХРАНЕНИЕ МНОЖИТЕЛЯ
              ;ОБНУЛЕНИЕ ТЕКУЩЕЙ СУММЫ ЧП
0701 AF      XRA      A
0702 67      MOV     H,A
0703 6F      MOV     L,A
              ;ПРОВЕРКА МНОЖИМОГО НА 0
0704 B1      ORA      C
0705 B2      ORA      D
0706 B3      ORA      E
0707 C20E07  JNZ     ПЕР1     ;ЕСЛИ МНОЖИМОЕ НЕ 0
070A F1      POP     PSW      ;БАЛАНС СТЕКА
070B C31307  JMP     ПЕР2     ;МНОЖИМОЕ=0
              ;ПРОВЕРКА МНОЖИТЕЛЯ НА 0
070E F1      ПЕР1: POP     PSW      ;ВОССТАНОВЛЕНИЕ МНОЖИТЕЛЯ
070F B7      ORA      A
0710 C21607  JNZ     ПЕР3     ;ЕСЛИ МНОЖИТЕЛЬ НЕ 0
              ;ОБНУЛЕНИЕ ПРОИЗВЕДЕНИЯ
0713 47      ПЕР2: MOV     B,A
0714 4F      MOV     C,A
0715 C9      RET
              ;ВЫПОЛНЕНИЕ УМНОЖЕНИЯ СРБ,МЛБ МНОЖИМОГО НА МНОЖИТЕЛЬ
0716 F5      ПЕР3: PUSH    PSW      ;СОХРАНЕНИЕ МНОЖИТЕЛЯ
0717 C5      PUSH    B          ;СОХРАНЕНИЕ СТБ МНОЖИМОГО
0718 D5      PUSH    D          ;СОХРАНЕНИЕ СРБ,МЛБ МНОЖИМОГО
0719 F5      PUSH    PSW      ;СОХРАНЕНИЕ СТБ МНОЖИТЕЛЯ
071A C5      PUSH    B          ;СОХРАНЕНИЕ СТБ МНОЖИМОГО
071B CDD005  CALL    Y24A      ; (A,H,L)-ПРОИЗВЕДЕНИЕ 1 (ПР1)
              ;ВЫПОЛНЕНИЕ УМНОЖЕНИЯ СТБ МНОЖИМОГО НА МНОЖИТЕЛЬ
071E D1      POP     D          ; (E)-СТБ МНОЖИМОГО
071F 4F      MOV     C,A        ; (C)-СТБ ПР1
0720 F1      POP     PSW      ; (A)-МНОЖИТЕЛЬ
0721 E5      PUSH    H          ;СОХРАНЕНИЕ СРБ,МЛБ ПР1
0722 C5      PUSH    B          ;СОХРАНЕНИЕ СТБ ПР1
0723 CD7005  CALL    Y8BE1     ; (H,L)-ПРОИЗВЕДЕНИЕ ПР2
              ;СЛОЖЕНИЕ ПР2 СО СДВИГОМ НА 2 БАЙТА ВПРАВО С ПР1
0726 C1      POP     B          ; (C)-СТБ ПР1
0727 0600    MVI     B,0
0729 09      DAD     B
072A 44      MOV     B,H
072B 4D      MOV     C,L        ; (B,C)-СТБ,СРБ2 РЕЗУЛЬТАТА
072C E1      POP     H          ; (H,L)-СРБ1,МЛБ РЕЗУЛЬТАТА
072D D1      POP     D          ; (D,E)-СРБ,МЛБ МНОЖИМОГО
              ;ПРОВЕРКА ЗНАКА МНОЖИТЕЛЯ: (A)-МНОЖИТЕЛЬ
072E 17      RAL
072F D23D07  JNC     ПЕР4     ;ЕСЛИ ЗНАК "+"
              ;ВЫЧИТАНИЕ МНОЖИМОГО ИЗ СТАРШИХ БАЙТОВ ПРОИЗВЕДЕНИЯ
0732 7C      MOV     A,H
0733 93      SUB     E          ;ВЫЧИТАНИЕ МЛБ
0734 67      MOV     H,A
0735 79      MOV     A,C
0736 9A      SBB     D          ;ВЫЧИТАНИЕ СРБ
0737 4F      MOV     C,A
0738 E3      XTHL
0739 78      MOV     A,B
073A 9D      SBB     L          ;ВЫЧИТАНИЕ СТБ

```

```

073B 47      MOV      B,A
073C E3      XTHL
              ;ПРОВЕРКА ЗНАКА МНОЖИМОГО
073D D1      PER4:  POP      D      ; (E) - СТВ МНОЖИМОГО
073E 7B      MOV      A,E
073F 17      RAL
0740 D1      POP      D      ; (D) - МНОЖИТЕЛЬ
0741 D0      RNC      ; ЕСЛИ ЗНАК "+"
              ; ВЫЧИТАНИЕ МНОЖИТЕЛЯ ИЗ СТВ ПРОИЗВЕДЕНИЯ
0742 78      MOV      A,B
0743 92      SUB      D
0744 47      MOV      B,A
0745 C9      RET
0000        END

```

Для временного хранения промежуточных результатов в программе задействована область стека. Тестовые данные для программы УД248 приведены в табл. 1.11.

**Табл. 1.11. Тесты умножения формата  $8 \cdot 24 = 32$  (целые двоичные числа в дополнительном коде)**

Представление чисел	
шестнадцатеричное	десятичное
FF·FFFFFF=00000001	$(-1) \cdot (-1) = +1$
01·FFFFFF=FFFFFFFF	$(+1) \cdot (-1) = -1$
0F·F00000=FF100000	$(+15) \cdot (-1048576) = -15728640$
AA·555555=E3555572	$(-86) \cdot (+5592405) = -480946830$
80·800000=40000000	$(-128) \cdot (-8388608) = +1073741824$
7F·7FFFFFFF=3F7FFF81	$(+127) \cdot (+8388607) = +1065353089$

### 1.3.4. ДРОБНЫЕ ЧИСЛА СО ЗНАКОМ

#### 1.3.4.1. Умножение дробных чисел

Программы, приведенные ниже, реализуют умножение дробных двоичных чисел в дополнительном коде форматов  $16 \cdot 16 = 16$ ,  $17 \cdot 17 = 17$ ,  $24 \cdot 24 = 24$  на основе обращения к соответствующим подпрограммам целочисленной арифметики, рассмотренным выше. Кроме того, приводится программа умножения формата  $16 \cdot 16 = 32$ , основанная на алгоритме Бута. Формат произведения здесь не ограничен форматом сомножителей, как в других программах умножения чисел ограниченной точности, что позволяет использовать программу и для умножения целых точных чисел.

Выше отмечалось, что программы умножения целых беззнаковых чисел применимы и к умножению дробных беззнаковых чисел. Правильность этого утверждения можно показать на примерах умножения двоичных чисел, имеющих одинаковое начертание, но различающихся положением запятой (для простоты рассматриваются двухразрядные числа):

$$\begin{array}{ll} (00,) \cdot (00,) = 0000, = 0_{10} & (,00) \cdot (,00) = ,0000 = 0_{10} \\ (01,) \cdot (01,) = 0001, = 1_{10} & (,01) \cdot (,01) = ,0001 = 0,0625_{10} \\ (10,) \cdot (10,) = 0100, = 4_{10} & (,10) \cdot (,10) = ,0100 = 0,25_{10} \\ (11,) \cdot (11,) = 1001, = 9_{10} & (,11) \cdot (,11) = ,1001 = 0,5625_{10} \end{array}$$

Очевидно, что начертания произведений дробных и целых чисел совпадают и отличаются лишь начальным положением запятой (неявным масштабным множителем). Следовательно, программы умножения целых беззнаковых чисел полностью применимы для умножения дробных беззнаковых чисел.

Аналогично можно использовать умножение целых беззнаковых двоичных чисел для получения модуля произведения дробных чисел со знаком:

$$\begin{array}{ll} (000,) \cdot (000,) = 000000, & (0,00) \cdot (0,00) = 0,0000 \\ (001,) \cdot (001,) = 000001, & (0,01) \cdot (0,01) = 0,0001 \\ (010,) \cdot (010,) = 000100, & (0,10) \cdot (0,10) = 0,0100 \\ (011,) \cdot (011,) = 001001, & (0,11) \cdot (0,11) = 0,1001 \end{array}$$

Сравнивая начертания произведений целых и дробных чисел со знаком, легко заметить, что последние образуются путем сдвига изображения соответствующего целочисленного произведения на один разряд влево.

Методика *умножения дробных знаковых чисел* с использованием программ умножения целых чисел имеет следующий вид: 1) получение модулей сомножителей; 2) формирование их произведения с помощью программ целочисленной арифметики; 3) сдвиг произведения влево на один разряд; 4) преобразование модуля произведения в дополнительный код, если знаки сомножителей различны.

#### 1.3.4.2. Формат 16 · 16 = 16

Программа УДФ16 реализует знаковое умножение, обращаясь к подпрограмме полноформатного беззнаково-



го умножения УЗ2Б и трем вспомогательным подпрограммам ДОПВ, ДОПД, ДОПН:

0750		ORG	750H
0630	УЗ2Б	SET	630H
0050	ДОПВ	SET	50H
0058	ДОПД	SET	58H
0060	ДОПН	SET	60H

УД016:

```

;*****
;ПОДПРОГРАММА УМНОЖЕНИЯ ДВОИЧНЫХ ЧИСЕЛ В ДОПОЛНИТЕЛЬНОМ
;КОДЕ С ФИКСИРОВАННОЙ ПОСЛЕ ЗНАКОВОГО РАЗРЯДА ЗАПЯТОЙ
;ФОРМАТА 16*16=16.
;ВХОДНЫЕ ПАРАМЕТРЫ: (В,С)–МНОЖИТЕЛЬ, (D,E)–МНОЖИМОЕ. ВЫ-
;ХОДНОЙ ПАРАМЕТР: (H,L)–СТАРШИЕ 2 БАЙТА ПРОИЗВЕДЕНИЯ.
;ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ, ГЛУБИНА СТЕКА–12. ИСПОЛЬЗУ-
;ЮТСЯ ПОДПРОГРАММЫ: *УЗ2Б*, *ДОПВ*, *ДОПД*, *ДОПН*, *У24А*.
;ОЦЕНКА: ДЛИНА–45 (+84 БАЙТА ПОДПРОГРАММ) БАЙТ, ВРЕМЯ–
;НЕ БОЛЕЕ 1422 ТАКТОВ (С УЧЕТОМ ПОДПРОГРАММ).
;*****
;ОБНУЖЕНИЕ СУММЫ ЧП, ПРОВЕРКА СОМНОЖИТЕЛЕЙ НА 0

```

```

0750 AF      XRA      A
0751 67      MOV      H,A
0752 6F      MOV      L,A
0753 B2      ORA      D
0754 B3      ORA      E
0755 C8      RZ              ;ЕСЛИ МНОЖИМОЕ=0
0756 78      MOV      A,B
0757 B1      ORA      C
0758 C8      RZ              ;ЕСЛИ МНОЖИТЕЛЬ=0
;АНАЛИЗ ЗНАКОВ СОМНОЖИТЕЛЕЙ НА СООПАДЕНИЕ
0759 7A      MOV      A,D
075A A8      XRA      B      ;S=0, ЕСЛИ ЗНАКИ ОДИНАКОВЫ
075B F5      PUSH     PSW     ;СОХРАНЕНИЕ S В СТЕКЕ
;ПРОВЕРКА ЗНАКА МНОЖИМОГО
075C 7A      MOV      A,D
075D 17      RAL
075E D26407 JNC      PER1     ;ЕСЛИ ЗНАК "+"
0761 CD5800 CALL     DOПД      ;(D,E)–ДОПОЛНИТЕЛЬНЫЙ КОД

```

;ПРОВЕРКА ЗНАКА МНОЖИТЕЛЯ

```

0764 78      PER1: MOV     A,B
0765 17      RAL
0766 D26C07 JNC      PER2     ;ЕСЛИ ЗНАК "+"
0769 CD5000 CALL     ДОПВ      ;(В,С)–ДОПОЛНИТЕЛЬНЫЙ КОД
;БЕЗЗНАКОВОЕ УМНОЖЕНИЕ ДВОИЧНЫХ ЧИСЕЛ 16*16=32
076C CD3006 PER2: CALL    УЗ2Б      ;(В,С,H,L)–ПРОИЗВЕДЕНИЕ
;СДВИГ СТАРШИХ БАЙТОВ ПРОИЗВЕДЕНИЯ ВЛЕВО НА 1 РАЗРЯД
076F 7C      MOV      A,H
0770 17      RAL
0771 79      MOV      A,C
0772 17      RAL
0773 6F      MOV      L,A
0774 78      MOV      A,B
0775 17      RAL
0776 67      MOV      H,A      ;(H,L)–ПРОИЗВЕДЕНИЕ

```

;КОРРЕКЦИЯ ПРОИЗВЕДЕНИЯ С УЧЕТОМ ЗНАКОВ СОМНОЖИТЕЛЕЙ

0777 F1	POP	PSW	;ВОССТАНОВЛЕНИЕ ПРИЗНАКА S
0778 F0	RP		;ЕСЛИ ЗНАКИ ОДИНАКОВЫ
0779 CD6000	CALL	ДОПН	; (H,L) -ДОПОЛНИТЕЛЬНЫЙ КОД
077C C9	RET		
0000	END		

В программе использовано округление произведения путем отбрасывания младших 16 разрядов полного произведения. Вспомогательные программы ДОПВ, ДОПД, ДОПН выполняют преобразование чисел, размещенных соответственно в регистровых парах (В, С), (D, E) и (H, L), в дополнительный код в соответствии с формулой (1.8):

```

0050                                ORG      50H

ДОПВ:
;*****
;ПОДПРОГРАММА ПРЕОБРАЗОВАНИЯ КОДА ЧИСЛА В РЕГИСТР-
;ВОЙ ПАРЕ (В,С) В ДОПОЛНИТЕЛЬНЫЙ КОД.
;ВХОДНОЙ ПАРАМЕТР: (В,С)-ДВОИЧНОЕ ЧИСЛО.ВЫХОДНОЙ ПАРА-
;МЕТР: (В,С)-ДОПОЛНИТЕЛЬНЫЙ КОД ЧИСЛА.ИСПОЛЬЗУЕТСЯ
;РЕГИСТР А.
;ОЦЕНКА:ДЛИНА-8 БАЙТ,ВРЕМЯ-43 ТАКТА.
;*****
0050 78          MOV      A,B
0051 2F          CMA
0052 47          MOV      B,A
0053 79          MOV      A,C
0054 2F          CMA
0055 4F          MOV      C,A
0056 03          INX      B
0057 C9          RET

ДОПД:
;*****
;ПОДПРОГРАММА ПРЕОБРАЗОВАНИЯ КОДА ЧИСЛА В РЕГИСТР-
;ВОЙ ПАРЕ (D,E) В ДОПОЛНИТЕЛЬНЫЙ КОД.
;ВХОДНОЙ ПАРАМЕТР: (D,E)-ДВОИЧНЫЙ КОД ЧИСЛА.ВЫХОДНОЙ
;ПАРАМЕТР: (D,E)-ДОПОЛНИТЕЛЬНЫЙ КОД.ИСПОЛЬЗУЕТСЯ РЕ-
;ГИСТР А.
;ОЦЕНКА:ДЛИНА-8 БАЙТ,ВРЕМЯ-43 ТАКТА.
;*****
0058 7A          MOV      A,D
0059 2F          CMA
005A 57          MOV      D,A
005B 7B          MOV      A,E
005C 2F          CMA
005D 5F          MOV      E,A
005E 13          INX      D
005F C9          RET

ДОПН:
;*****
;ПОДПРОГРАММА ПРЕОБРАЗОВАНИЯ КОДА ЧИСЛА В РЕГИСТР-
;ВОЙ ПАРЕ (H,L) В ДОПОЛНИТЕЛЬНЫЙ КОД.
;ВХОДНОЙ ПАРАМЕТР: (H,L)-ДВОИЧНЫЙ КОД ЧИСЛА.ВЫХОДНОЙ

```

ПАРАМЕТР: (H,L) — ДОПОЛНИТЕЛЬНЫЙ КОД. ИСПОЛЬЗУЕТСЯ РЕ-  
ГИСТР A.  
ОЦЕНКА: ДЛИНА—8 БАЙТ, ВРЕМЯ—43 ТАКТА.

\*\*\*\*\*

```

0060 7C      MOV     A,H
0061 2F      CMA
0062 67      MOV     H,A
0063 7D      MOV     A,L
0064 2F      CMA
0065 6F      MOV     L,A
0066 23      INC     H
0067 C9      RET
0000        END

```

Эти вспомогательные программы широко используются далее при разработке арифметических программ.

Тестовые данные для программы УДФ16 приведены в табл. 1.12 (запятая в данных подразумевается после первого бита старшей шестнадцатеричной цифры).

Табл. 1.12. Тесты умножения формата  $16 \cdot 16 = 16$   
(дробные двоичные числа в дополнительном коде)

Представление чисел	
шестнадцатеричное	десятичное
$4000 \cdot 4000 = 2000$	$0,5 \cdot 0,5 = 0,25$
$7FFF \cdot 7FFF = 7FFE$	$0,99996 \cdot 0,99996 = 0,99992$
$5555 \cdot 5555 = 38E3$	$0,66666 \cdot 0,66666 = 0,44443$
$AAAA \cdot AAAA = 38E4$	$(-0,66666) \cdot (-0,66666) = 0,44443$
$5555 \cdot AAAA = C71D$	$0,66666 \cdot (-0,66666) = (-0,44443)$
$FFFF \cdot FFFF = 0000$	$(-0,00003) \cdot (-0,00003) = 0,00000$

### 1.3.4.3. Формат $17 \cdot 17 = 17$

Программа УДФ17 в отличие от всех предыдущих программ умножения оперирует с сомножителями, размещенными не на регистрах, а в памяти системы, причем знак каждого сомножителя расположен в старшем разряде знакового байта, а цифровая часть сомножителя — в двух соседних байтах:

```

0790      ORG     790H
0050      ДОПВ   SET     50H
0058      ДОПД   SET     58H
0630      УЗДВ   SET     630H

```

УДФ17:

\*\*\*\*\*  
ПОДПРОГРАММА УМНОЖЕНИЯ ДВОИЧНЫХ ЧИСЕЛ В ДОПОЛНИТЕЛЬНОМ  
КОДЕ С ФИКСИРОВАННОЙ ПОСЛЕ ЗНАКОВОГО РАЗРЯДА ЗАПЯТОЙ

```

;ФОРМАТА (1,16)*(1,16)=(1,16),ГДЕ (1,16)=(ЗНАК,СТБ,МЛБ)
;ВХОДНЫЕ ПАРАМЕТРЫ:(D,E)-АДРЕС МНОЖИМОГО МН В ПАМЯТИ,
;(H,L)-АДРЕС МНОЖИТЕЛЯ МН В ПАМЯТИ.ВЫХОДНОЙ ПАРАМЕТР:
;(B,C)-ДВА СТАРШИХ БАЙТА ПРОИЗВЕДЕНИЯ.ИСПОЛЬЗУЮТСЯ ВСЕ
;РЕГИСТРЫ,СОХРАНЯЮТСЯ (D,E),(H,L).ГЛУБИНА СТЕКА-8.ИС-
;ПОЛЬЗУЮТСЯ ПОДПРОГРАММЫ:*ДОПВ*,*ДОПД*,*УЗ2Б*,*У24А*.
;ОЦЕНКА:ДЛИНА-41 БАЙТ (+76 БАЙТ ПОДПРОГРАММ),ВРЕМЯ-НЕ
;БОЛЕЕ 1481 ТАКТОВ (С УЧЕТОМ ПОДПРОГРАММ).
;*****

```

```

0790 D5      PUSH    D
0791 E5      PUSH    H
;ПРОВЕРКА ЗНАКОВ СОМНОЖИТЕЛЕЙ НА СОВПАДЕНИЕ
0792 1A      LDAX    D
0793 AE      XRA     M      ;(A7)=1,(S=1),ЕСЛИ ЗНАКИ РАЗНЫЕ
0794 F5      PUSH    PSW     ;СОХРАНЕНИЕ ПРИЗНАКОВ
;ПЕРЕСЫЛКА МНОЖИМОГО ИЗ ПАМЯТИ В (B,C)
0795 EB      XCHG
0796 7E      MOV     A,M
0797 B7      ORA     A      ;ПРОЯВЛЕНИЕ ПРИЗНАКА ЗНАКА
0798 23      INX     H
0799 46      MOV     B,M
079A 23      INX     H
079B 4E      MOV     C,M     ;(B,C)-МНОЖИМОЕ
079C EB      XCHG
079D FC5000 CM      ДОПВ    ;ДОПОЛНЕНИЕ МН,ЕСЛИ МН <0
;ПЕРЕСЫЛКА МНОЖИТЕЛЯ ИЗ ПАМЯТИ В (D,E)
07A0 7E      MOV     A,M
07A1 B7      ORA     A      ;ПРОЯВЛЕНИЕ ПРИЗНАКА ЗНАКА
07A2 23      INX     H
07A3 56      MOV     D,M
07A4 23      INX     H
07A5 5E      MOV     E,M     ;(D,E)-МНОЖИТЕЛЬ
07A6 FC5800 CM      ДОПД    ;ДОПОЛНЕНИЕ МН,ЕСЛИ МН <0
;БЕЗЗНАКОВОЕ УМНОЖЕНИЕ ДВОИЧНЫХ ЧИСЕЛ 16*16=32
07A9 CD3006 CALL    Y32B    ;(B,C,H,L)-ПРОИЗВЕДЕНИЕ (ПР)
07AC 7C      MOV     A,H
07AD 17      RAL
07AE D2B207 JNC     ПЕР
07B1 03      INX     B      ;ОКРУГЛЕНИЕ СТБ,СРБ ПР
07B2 F1      PER:    POP    PSW ;ВОССТАНОВЛЕНИЕ ПРИЗНАКОВ
07B3 FC5000 CM      ДОПВ    ;ДОПОЛНЕНИЕ ПР,ЕСЛИ ПР <0
07B6 E1      POP     H
07B7 D1      POP     D
07B8 C9      RET
0000      END

```

Программа УДФ17, как и УДФ16, использует обращение к подпрограмме беззнакового умножения УЗ2Б и вспомогательным программам ДОПВ, ДОПД, но в отличие от программы УДФ16 не производит сдвига влево произведения, так как в данном случае цифровые части сомножителей можно интерпретировать как дробные беззнаковые числа. Кроме того, в программе УДФ17 используется при переходе от полноформатного произведения к сокращенному симметричный метод округления

(1.9). Эта программа является базовой при построении программ умножения чисел с плавающей запятой (об этом см. в гл. 2). Тестовые данные для программы УДФ17 можно получить из табл. 1.12 путем сдвига ее данных на один двоичный разряд влево.

#### 1.3.4.4. Формат 24 · 24 = 24

Программа УДФ24, как и УДФ17, оперирует с сомножителями, размещенными в памяти, причем каждый сомножитель содержит три байта (СТБ, СРБ, МЛБ) со знаковым разрядом в старшем бите СТБ:

0810		ORG	810H
0108	ДОПЗ	SET	108H
0050	ДОПВ	SET	50H
05D0	У24А	SET	5D0H
0570	У88Б1	SET	570H

УДФ24:

```

;*****
;ПОДПРОГРАММА УМНОЖЕНИЯ ДВОИЧНЫХ ЧИСЕЛ В ДОПОЛНИТЕЛЬНОМ
;КОДЕ С ФИКСИРОВАННОЙ ПОСЛЕ ЗНАКОВОГО РАЗРЯДА ЗАПЯТОЙ
;ФОРМАТА 24*24=24.
;ВХОДНЫЕ ПАРАМЕТРЫ: (D,E) - АДРЕС МНОЖИМОГО ММ В ПАМЯТИ,
; (H,L) - АДРЕС МНОЖИТЕЛЯ МН В ПАМЯТИ. ВЫХОДНОЙ ПАРАМЕТР:
; (A,B,C) - ТРИ СТАРШИХ БАЙТА ПРОИЗВЕДЕНИЯ. ИСПОЛЗУЮТСЯ ВСЕ
;РЕГИСТРЫ, СОХРАНЯЮТСЯ (D,E), (H,L). ГЛУБИНА СТЕКА - 16. ИСП-
;ПОЛЗУЮТСЯ ПОДПРОГРАММЫ: *ДОПЗ*, *ДОПВ*, *У24А*, *У88Б1*.
;ОЦЕНКА: ДЛИНА - 114 БАЙТ (+51 БАЙТ ПОДПРОГРАММ), ВРЕМЯ - НЕ
;БОЛЕЕ 2682 ТАКОВ (С УЧЕТОМ ПОДПРОГРАММ).
;*****

```

0810 D5	PUSH	D	; СОХРАНЕНИЕ АДРЕСА ММ
0811 E5	PUSH	H	; СОХРАНЕНИЕ АДРЕСА МН
; ПРОВЕРКА ЗНАКОВ СОМНОЖИТЕЛЕЙ НА СОВПАДЕНИЕ			
0812 1A	LDAX	D	
0813 AE	XRA	M	; S=1, ЕСЛИ ЗНАКИ РАЗНЫЕ
0814 F5	PUSH	PSW	; СОХРАНЕНИЕ ПРИЗНАКОВ
; ДОПОЛНЕНИЕ ОТРИЦАТЕЛЬНЫХ СОМНОЖИТЕЛЕЙ			
0815 1A	LDAX	D	
0816 B7	ORA	A	; ПРОЯВЛЕНИЕ ЗНАКА ММ
0817 EB	XCHG		
0818 FC0801	CM	ДОПЗ	; ДОПОЛНЕНИЕ ММ
081B EB	XCHG		
081C 7E	MOV	A, M	
081D B7	ORA	A	; ПРОЯВЛЕНИЕ ЗНАКА МН
081E FC0801	CM	ДОПЗ	; ДОПОЛНЕНИЕ МН
; ЗАГРУЗКА МНОЖИМОГО В РЕГИСТРЫ			
0821 E5	PUSH	H	; СОХРАНЕНИЕ АДРЕСА МН
0822 EB	XCHG		; (H,L) - АДРЕС ММ
0823 56	MOV	D, M	
0824 23	INX	H	
0825 5E	MOV	E, M	
0826 23	INX	H	
0827 7E	MOV	A, M	; (D,E,A) - СТБ, СРБ, МЛБ ММ
0828 E1	POP	H	; ВОССТАНОВЛЕНИЕ АДРЕСА МН

	;УМНОЖЕНИЕ МЛБ МНОЖИТЕЛЯ НА СТВ,СРБ МНОЖИМОГО		
0829 E5	PUSH	H	;СОХРАНЕНИЕ АДРЕСА МН
082A 23	INX	H	
082B 23	INX	H	; (H,L)-АДРЕС МЛБ МН
082C F5	PUSH	PSW	;СОХРАНЕНИЕ МЛБ МН
082D 7E	MOV	A,M	; (A)-МЛБ МН
082E CDD005	CALL	Y24A	; (A,H,L)-ПРОИЗВЕДЕНИЕ ПР1
0831 47	MOV	B,A	
0832 4C	MOV	C,H	; (B,C)-СТВ,СРБ ПР1
0833 F1	POP	PSW	;ВОССТАНОВЛЕНИЕ МЛБ МН
	;УМНОЖЕНИЕ МЛБ МНОЖИМОГО НА СТВ МНОЖИТЕЛЯ		
0834 E1	POP	H	;ВОССТАНОВЛЕНИЕ АДРЕСА МН
0835 D5	PUSH	D	;СОХРАНЕНИЕ СТВ,СРБ МН
0836 5F	MOV	E,A	; (E)-МЛБ МН
0837 7E	MOV	A,M	; (A)-СТВ МН
0838 E5	PUSH	H	;СОХРАНЕНИЕ АДРЕСА МН
0839 C5	PUSH	B	;СОХРАНЕНИЕ ПР1
083A CD7005	CALL	Y88B1	; (H,L)-ПРОИЗВЕДЕНИЕ ПР2
083D C1	POP	B	;ВОССТАНОВЛЕНИЕ ПР1
	;СЛОЖЕНИЕ ПРОИЗВЕДЕНИИ ПР1 И ПР2		
083E 3E00	MVI	A,0	
0840 09	DAD	B	
0841 8F	ADC	A	;УЧЕТ ПЕРЕНОСА
0842 44	MOV	B,H	
0843 4D	MOV	C,L	; (A,B,C)-СУММА ПР1+ПР2
0844 E1	POP	H	;ВОССТАНОВЛЕНИЕ АДРЕСА МН
0845 D1	POP	D	;ВОССТАНОВЛЕНИЕ СТВ,СРБ МН
	;УМНОЖЕНИЕ СРБ МНОЖИТЕЛЯ НА СТВ,СРБ МНОЖИМОГО		
0846 D5	PUSH	D	;СОХРАНЕНИЕ СТВ,СРБ МН
0847 E5	PUSH	H	;СОХРАНЕНИЕ АДРЕСА МН
0848 23	INX	H	; (H,L)-АДРЕС СРБ МН
0849 F5	PUSH	PSW	
084A C5	PUSH	B	;СОХРАНЕНИЕ СУММЫ ПР1+ПР2
084B 7E	MOV	A,M	; (A)-СРБ МН
084C CDD005	CALL	Y24A	; (A,H,L)-ПРОИЗВЕДЕНИЕ ПР3
084F C1	POP	B	
0850 D1	POP	D	; (D,B,C)-СУММА ПР1+ПР2
	;СЛОЖЕНИЕ СУММЫ ПР1+ПР2 С ПР3		
0851 5A	MOV	E,D	
0852 1600	MVI	D,0	
0854 09	DAD	B	
0855 8B	ADC	E	
0856 D25A08	JNC	ПЕР1	
0859 13	INX	D	;УЧЕТ ПЕРЕНОСА
085A 47	ПЕР1: MOV	B,A	
085B 7A	MOV	A,D	
085C 4C	MOV	C,H	; (A,B,C)-СУММА ПР1+ПР2+ПР3
085D E1	POP	H	;ВОССТАНОВЛЕНИЕ АДРЕСА МН
085E D1	POP	D	;ВОССТАНОВЛЕНИЕ СТВ,СРБ МН
	;УМНОЖЕНИЕ СТВ МНОЖИТЕЛЯ НА СТВ,СРБ МНОЖИМОГО		
085F F5	PUSH	PSW	;СОХРАНЕНИЕ МЛБ СУММЫ
0860 C5	PUSH	B	;СОХРАНЕНИЕ СТВ,СРБ СУММЫ
0861 7E	MOV	A,M	; (A)-СТВ МН
0862 CDD005	CALL	Y24A	; (A,H,L)-ПРОИЗВЕДЕНИЕ ПР4
0865 C1	POP	B	
0866 D1	POP	D	; (D,B,C)-СУММА ПР1+ПР2+ПР3

```

;СЛОЖЕНИЕ СУММЫ ПР1+ПР2+ПР3 С ПР4
0867 09      DAD      B
0868 82      ADD      D
0869 D26D08  JNC      ПЕР2
086C 3C      INR      A      ;УЧЕТ ПЕРЕНОСА
086D 29      ПЕР2: DAD      H      ;САВИГ ВЛЕВО СУММЫ В (A,H,L)
086E 8F      ADC      A
086F 44      MOV      B,H
0870 4D      MOV      C,L
0871 67      MOV      H,A      ; (H,B,C) -СУММА ПР1+ПР2+ПР3+ПР4

;ДОПОЛНЕНИЕ ПРОИЗВЕДЕНИЯ ПР
0872 F1      POP      PSW      ;ВОССТАНОВЛЕНИЕ ПРИЗНАКОВ
0873 7C      MOV      A,H
0874 F27F08  JP      ПЕР3      ;ЕСЛИ ЗНАК ПР "+"
0877 5F      MOV      E,A
0878 CD5000  CALL     ДОПВ      ;ДОПОЛНЕНИЕ СРБ,МЛБ ПР
087B 7B      MOV      A,E
087C 2F      CMA
087D CE00    ACI      0      ;ДОПОЛНЕНИЕ СТБ ПР
087F E1      ПЕР3: POP      H      ;ВОССТАНОВЛЕНИЕ АДРЕСА МН
0880 D1      POP      D      ;ВОССТАНОВЛЕНИЕ АДРЕСА ММ
0881 C9      RET

```

Программа преобразует сомножители в прямой код, обращаясь к вспомогательной программе ДОПЗ, которая реализует дополнение трехбайтного числа непосредственно в памяти:

```

0108                                ORG      108H
                                ДОПЗ:
;*****
;ПОДПРОГРАММА ПРЕОБРАЗОВАНИЯ 3-БАЙТНОГО ДВОИЧНОГО ЧИСЛА
;В ПАМЯТИ В ДОПОЛНИТЕЛЬНЫЙ КОД.
;ВХОДНОЙ ПАРАМЕТР: (H,L) -СТАРШИЙ БАЙТ ЧИСЛА.ИСПОЛЬЗУЕТСЯ
;РЕГИСТР А,СОХРАНЯЕТСЯ (H,L).
;ОЦЕНКА:ДЛИНА-20 БАЙТ,ВРЕМЯ-105 ТАКТОВ.
;*****
0108 23      INX      H
0109 23      INX      H
010A 7E      MOV      A,M
010B 2F      CMA      ;ИНВЕРСИЯ МЛБ
010C C601    ADI      01
010E 77      MOV      M,A
010F 2B      DCX      H
0110 7E      MOV      A,M
0111 2F      CMA      ;ИНВЕРСИЯ СРБ
0112 CE00    ACI      0
0114 77      MOV      M,A
0115 2B      DCX      H
0116 7E      MOV      A,M
0117 2F      CMA      ;ИНВЕРСИЯ СТБ
0118 CE00    ACI      0
011A 77      MOV      M,A
011B C9      RET
0000      END

```

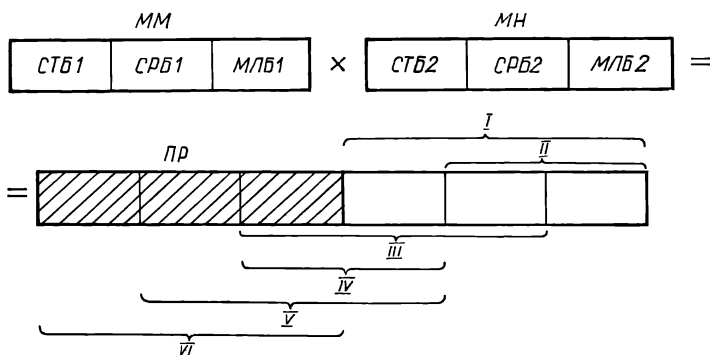


Рис. 1. 7. Схема формирования произведения дробных чисел формата  $24 \times 24 = 24$ :

$I - \text{СРБ}2 \cdot \text{МЛБ}1$ ;  $II - \text{МЛБ}2 \cdot \text{МЛБ}1$ ;  $III - \text{ПР}1 = \text{МЛБ}2 \cdot (\text{СТБ}1, \text{СРБ}1)$ ;  $IV - \text{ПР}2 = \text{СТБ}2 \cdot \text{МЛБ}1$ ;  $V - \text{ПР}3 = \text{СРБ}2 \cdot (\text{СТБ}1, \text{СРБ}1)$ ;  $VI - \text{СТБ}2 \times (\text{СТБ}1, \text{СРБ}1)$

Программа УДФ24 выполняет серию промежуточных умножений путем вызова подпрограмм У24А и У88Б1. Вычисление конечного произведения происходит по схеме, приведенной на рис. 1.7. Полноформатное произведение  $24 \cdot 24 = 48$  содержит все шесть комбинаций промежуточных произведений форматов  $8 \cdot 8 = 16$  и  $8 \cdot 16 = 24$ , образуемых умножением соответствующих байтов ММ и МН. Поскольку сокращенное произведение (заштриховано на рис. 1.7) содержит только три старших байта полного произведения, то, очевидно, нет необходимости вычислять все промежуточные произведения. В программе УДФ24 полностью используются промежуточные произведения ПР4, ПР3, ПР2 и частично (без своего МЛБ) ПР1. Отбрасывание невычисляемых произведений  $\text{СРБ}2 \times \text{МЛБ}1$ ,  $\text{МЛБ}2 \cdot \text{МЛБ}1$  и  $\text{МЛБ} \text{ ПР}1$  дает граничную абсолютную ошибку не более единицы младшего разряда сокращенного произведения. С учетом последующего сдвига произведения на один разряд влево эта ошибка увеличивается в два раза. Таким образом, граничная относительная ошибка округления полного произведения становится сравнимой с ошибкой операции умножения по формуле (1.18). Точность произведения можно повысить в два раза, если за счет усложнения программы вычислять дополнительно СТБ промежуточного произведения  $\text{СРБ}2 \cdot \text{МЛБ}1$ . Тестовые данные для программы УДФ24 приведены в табл. 1.13.



**Табл. 1.13. Тесты умножения формата 24·24=24  
(дробные двоичные числа в дополнительном коде)**

Представление чисел	
шестнадцатеричное	десятичное
400000·400000=200000	0,5·0,5=0,25
7FFFFFF·7FFFFFF=7FFFFC	0,9999998·0,9999998=0,9999996
555555·555555=38E38C	0,6666666·0,6666666=0,4444443
AAAAAA·AAAAAA=38E38E	(-0,6666666)·(-0,6666666)=0,4444443
555555·AAAAAA=C71C72	0,6666666·(-0,6666666)=-0,4444443
FFFFFF·FFFFFF=000000	(-0,0000001)·(-0,0000001)=0,0000000

### 1.3.4.5. Формат 16·16=32

Программа УДФ32 в отличие от предыдущих программ умножения знаковых чисел выполняет операции над знаковыми разрядами так же, как над цифровыми. В основе ее лежит алгоритм Бута [5, 61]:

07C0	ORG	7C0H
------	-----	------

```

УДФ32:
;*****
;ПОДПРОГРАММА УМНОЖЕНИЯ ДВОИЧНЫХ ЧИСЕЛ В ДОПОЛНИТЕЛЬ-
;НОМ КОДЕ С ФИКСИРОВАННОЙ ПОСЛЕ ЗНАКОВОГО РАЗРЯДА
;ЗАПЯТОЙ ФОРМАТА 16*16=32.
;МЕТОД УМНОЖЕНИЯ-АЛГОРИТМ БУТА.
;ВХОДНЫЕ ПАРАМЕТРЫ: (В,С)-МНОЖИТЕЛЬ, (D,E)-МНОЖИМОЕ. ВЫ-
;ХОДНЫЕ ПАРАМЕТРЫ: (H,L,V,C)-ПРОИЗВЕДЕНИЕ. ИСПОЛЬЗУЮТСЯ
;ВСЕ РЕГИСТРЫ, СОХРАНЯЕТСЯ (D,E). ГЛУБИНА СТЕКА-2.
;ОЦЕНКА: ДЛИНА-72 БАЙТ, ВРЕМЯ-НЕ БОЛЕЕ 2703 ТАКТА.
;*****
;ОБНУЛЕНИЕ ТЕКУЩЕЙ СУММЫ ЧП
07C0 AF      XRA      A
07C1 67      MOV      H,A
07C2 6F      MOV      L,A      ; (H,L)=0
;ПРОВЕРКА МНОЖИМОГО НА 0
07C3 B2      ORA      D
07C4 B3      ORA      E
07C5 C2CB07  JNZ      PER1      ; ЕСЛИ МНОЖИМОЕ НЕ 0
07C8 47      MOV      B,A
07C9 4F      MOV      C,A      ; (B,C)=0
07CA C9      RET
;ПРОВЕРКА МНОЖИТЕЛЯ НА 0
07CB 78      PER1: MOV      A,B
07CC B1      ORA      C
07CD C8      RZ          ; ЕСЛИ МНОЖИТЕЛЬ=0
07CE 3E10    MVI      A,16      ; СЧЕТЧИК ЦИКЛОВ
07D0 F5      PUSH     PSW      ; СОХРАНЕНИЕ СЧЕТЧИКА
07D1 AF      XRA      A      ; (CY)=0
07D2 81      ADD      C      ; (A)-МЛБ МНОЖИТЕЛЯ
;АНАЛИЗ МЛАДШЕГО БИТА МНОЖИТЕЛЯ И ПЕРЕНОСА: (X,CY)=?
07D3 D2DF07  JCPL: JNC      PER2 ; ЕСЛИ (X,CY)=(X,0)

```

07D6 E601	ANI	1	;ВЫДЕЛЕНИЕ БИТА
07D8 C2EA07	JNZ	ПЕР3	;ЕСЛИ (X,CY)=(1,1)
			;СЛОЖЕНИЕ СУММЫ ЧП С МНОЖИМЫМ ПРИ (X,CY)=(0,1)
07DB 19	DAD	D	
07DC C3EA07	JMP	ПЕР3	
			;АНАЛИЗ ПАРЫ БИТ (X,CY)=(X,0)
07DF E601	ПЕР2: ANI	1	;ВЫДЕЛЕНИЕ БИТА
07E1 CAEA07	JZ	ПЕР3	;ЕСЛИ (X,CY)=(0,0)
			;ВЫЧИТАНИЕ МНОЖИМОГО ИЗ СУММЫ ЧП ПРИ (X,CY)=(1,0)
07E4 7D	MOV	A,L	
07E5 93	SUB	E	
07E6 6F	MOV	L,A	
07E7 7C	MOV	A,H	
07E8 9A	SBB	D	
07E9 67	MOV	H,A	
			;ПРОВЕРКА КОНЦА ЦИКЛА
07EA F1	ПЕР3: POP	PSW	
07EB 3D	DCR	A	
07EC F5	PUSH	PSW	;СОХРАНЕНИЕ СЧЕТЧИКА
07ED CA0208	JZ	ПЕР4	;ЕСЛИ СЧЕТЧИК=0
			;АРИФМЕТИЧЕСКИЙ СДВИГ ВПРАВО СУММЫ ЧП И МНОЖИТЕЛЯ
07F0 7C	MOV	A,H	
07F1 FE80	CPI	128	
07F3 3F	CMC		; (CY)=ЗНАК СУММЫ ЧП
07F4 1F	RAR		
07F5 67	MOV	H,A	
07F6 7D	MOV	A,L	
07F7 1F	RAR		
07F8 6F	MOV	L,A	
07F9 78	MOV	A,B	
07FA 1F	RAR		
07FB 47	MOV	B,A	
07FC 79	MOV	A,C	
07FD 1F	RAR		
07FE 4F	MOV	C,A	
07FF C3D307	JMP	ЦИКЛ	;ЗАЦИКЛИВАНИЕ
			;КОНЕЦ ЦИКЛА
0802 3EFE	ПЕР4: MVI	A,0F0H	;ОБНУЛЕНИЕ МЛАДШЕГО
0804 A1	ANA	C	;БИТА ПРОИЗВЕДЕНИЯ
0805 4F	MOV	C,A	
0806 F1	POP	PSW	;БАЛАНС СТЕКА
0807 C9	RET		
0C00	END		

Особенность алгоритма состоит в сложении (вычитании) в каждом цикле умножения СЧП и ММ в зависимости не от значения очередного разряда МН, как обычно, а от изменения очередного разряда множителя относительно его предыдущего разряда. При изменении вида  $0 \rightarrow 1$  производится сложение СЧП с ММ, вида  $1 \rightarrow 0$  — вычитание ММ из СЧП, вида  $0 \rightarrow 0$  или  $1 \rightarrow 1$  — сдвиг СЧП. Сопоставление данной программы с наиболее близкой к ней по функциям УД32 показывает, что по быстро-

действию программа УДФ32 более чем в два раза проигрывает, но по затратам памяти на 29 % экономичнее (с учетом подпрограмм для УД32). Тестовые данные для программы УДФ32 можно легко получить из табл. 1.10.

## 1.4. УМНОЖЕНИЕ ДЕСЯТИЧНЫХ ЧИСЕЛ

*Умножение десятичных чисел*, как и умножение двоичных чисел, производится путем нахождения ЧП и СЧП, где ЧП представляет собой двоично-десятичное произведение десятичной цифры МН на десятичную цифру ММ. Отличия десятичного умножения от двоичного состоят в том, что, во-первых, выделение цифр МН и ММ для их умножения не сводится к однократному двоичному сдвигу, а требует специальной процедуры *распаковки* (выделения цифр); во-вторых, формирование ЧП двух десятичных цифр не сводится к альтернативному выбору между значением множимого и нулем, а требует специальной процедуры умножения цифр; в-третьих, формирование СЧП может быть реализовано сложением только однобайтных чисел с применением команды десятичной коррекции DAA. Указанные отличия существенно усложняют программы умножения десятичных чисел, увеличивая время их выполнения и затраты памяти. Поэтому десятичное умножение целесообразно использовать для сомножителей малой разрядности, а обработку чисел большой разрядности вести средствами двоичной арифметики с предварительным преобразованием десятичных чисел в двоичные.

Рассмотрим две программы умножения десятичных чисел. Программа У410 реализует точное умножение формата  $4 \cdot 4 = 8$  двух десятичных цифр методом двоично-десятичного накопления ММ в соответствии со значением МН:

0890

ORG

890H

У410:

```

;*****
;ПОДПРОГРАММА ДЕСЯТИЧНОГО УМНОЖЕНИЯ ОДНОРАЗРЯДНЫХ БЕЗ-
;ЗНАКОВЫХ ДВОИЧНО-ДЕСЯТИЧНЫХ (КОД 8421) ЧИСЕЛ ФОРМАТА 4.
;ВХОДНЫЕ ПАРАМЕТРЫ: (С) -ЦИФРА МНОЖИТЕЛЯ, (Е) -ЦИФРА МНОЖИ-
;МОГО. ВЫХОДНОЙ ПАРАМЕТР: (А) -ДВОИЧНО-ДЕСЯТИЧНОЕ ПРОИЗВЕ-
;ДЕНИЕ. СОХРАНЯЮТСЯ РЕГИСТРЫ (Е), (С). ГЛУБИНА СТЕКА -2.
;ОЦЕНКА: ДЛИНА -16 БАЙТ, ВРЕМЯ -НЕ БОЛЕЕ 267 ТАКТОВ.
;*****

```

```

;ПРОВЕРКА СМНОЖИТЕЛЕЙ НА 0
0890 AF      XRA      A
0891 83      ADD      E
0892 C8      RZ              ;ЕСЛИ МНОЖИМОЕ=0
0893 AF      XRA      A
0894 81      ADD      C
0895 C8      RZ              ;ЕСЛИ МНОЖИТЕЛЬ=0
0896 AF      XRA      A      ; (A)=0
0897 C5      PUSH     B      ;СОХРАНЕНИЕ МНОЖИТЕЛЯ

;ДВОИЧНО-ДЕСЯТИЧНОЕ НАКОПЛЕНИЕ СУММЫ
0898 83      ЦИКЛ: ADD     E
0899 27      DAA
089A 0D      DCR      C
089B C2980B  JNZ      ЦИКЛ   ;ЗАЦИКЛИВАНИЕ
089E C1      POP      B      ;ВОССТАНОВЛЕНИЕ МНОЖИТЕЛЯ
089F C9      RET
0000        END

```

Программа У810 реализует точное умножение формата  $(2 \cdot 4) \cdot (2 \cdot 4) = 4 \cdot 4$  двух десятичных двухразрядных чисел методом поочередного умножения младших (МЛЦ) и старших (СТЦ) цифр сомножителей:

```

08A0                ORG      8A0H
0890                Y410    SET      890H
08F0                C4410   SET      8F0H

У810:
;*****
;ПОДПРОГРАММА ДЕСЯТИЧНОГО УМНОЖЕНИЯ 2-РАЗРЯДНЫХ БЕЗ-
;ЗНАКОВЫХ ДВОИЧНО-ДЕСЯТИЧНЫХ (КОД 8421) ЧИСЕЛ ФОРМАТА 8.
;ВХОДНЫЕ ПАРАМЕТРЫ: (С)-МНОЖИТЕЛЬ, (Е)-МНОЖИМОЕ. ВЫХОДНОЙ
;ПАРАМЕТР: (H,L)-ДВОИЧНО-ДЕСЯТИЧНОЕ ПРОИЗВЕДЕНИЕ. ИСПОЛЬ-
;ЗУЮТСЯ ВСЕ РЕГИСТРЫ, ГЛУБИНА СТЕКА-6. ИСПОЛЬЗУЮТСЯ ПОД-
;ПРОГРАММЫ: *У410*, *С4410*.
;ОЦЕНКА: ДЛИНА-54 (+37 БАЙТА ПОДПРОГРАММ) БАЙТА, ВРЕМЯ-
;НЕ БОЛЕЕ 1543 ТАКТОВ (С УЧЕТОМ ПОДПРОГРАММ).
;*****
08A0 2600        MVI      H,0      ;ОБНУЛЕНИЕ СУММЫ ЧП
;ВЫДЕЛЕНИЕ СТАРШИХ ЦИФР СТЦ СМНОЖИТЕЛЕЙ
08A2 3EFO        MVI      A,0F0H   ;(A)-МАСКА НА СТЦ
08A4 A3          ANA      E
08A5 0F          RRC
08A6 0F          RRC
08A7 0F          RRC
08A8 0F          RRC
08A9 57          MOV      D,A      ;(D)-СТЦ1 МНОЖИМОГО
08AA 3EFO        MVI      A,0F0H   ;(A)-МАСКА НА СТЦ
08AC A1          ANA      C
08AD 0F          RRC
08AE 0F          RRC
08AF 0F          RRC
08B0 0F          RRC
08B1 47          MOV      B,A      ;(B)-СТЦ2 МНОЖИТЕЛЯ
;ВЫДЕЛЕНИЕ МЛАДШИХ ЦИФР МЛЦ СМНОЖИТЕЛЕЙ
08B2 3EFO        MVI      A,0FH    ;(A)-МАСКА НА МЛЦ
08B4 A3          ANA      E

```

```

08B5 5F      MOV     E,A      ; (E) - МЛЦ1 МНОЖИМОГО
08B6 3E0F    MVI     A,0FH    ; (A) - МАСКА НА МЛЦ
08B8 A1      ANA     C
08B9 4F      MOV     C,A      ; (C) - МЛЦ2 МНОЖИТЕЛЯ
; УМНОЖЕНИЕ МЛЦ1*МЛЦ2, СЛОЖЕНИЕ С СУММОЙ ЧП
08BA CD9008  CALL    Y410     ; (A) - ПРОИЗВЕДЕНИЕ
08BD 6F      MOV     L,A      ; (H,L) - СУММА ЧП
; УМНОЖЕНИЕ СТЦ1*МЛЦ2, СЛОЖЕНИЕ С СУММОЙ ЧП
08BE D5      PUSH    D        ; СОХРАНЕНИЕ МЛЦ1
08BF 5A      MOV     E,D      ; (E) - СТЦ1
08C0 CD9008  CALL    Y410     ; (A) - ПРОИЗВЕДЕНИЕ
08C3 CDF008  CALL    C4410    ; (H,L) - СУММА ЧП
; УМНОЖЕНИЕ МЛЦ1*СТЦ2, СЛОЖЕНИЕ С СУММОЙ ЧП
08C6 48      MOV     C,B      ; (C) - СТЦ2
08C7 D1      POP     D        ; (E) - МЛЦ1
08C8 CD9008  CALL    Y410     ; (A) - ПРОИЗВЕДЕНИЕ
08CB CDF008  CALL    C4410    ; (H,L) - СУММА ЧП
; УМНОЖЕНИЕ СТЦ1*СТЦ2, СЛОЖЕНИЕ С СУММОЙ ЧП
08CE 5A      MOV     E,D      ; (E) - СТЦ1
08CF CD9008  CALL    Y410     ; (A) - ПРОИЗВЕДЕНИЕ
08D2 84      ADD     H
08D3 27      DAA
08D4 67      MOV     H,A
08D5 C9      RET
0000 END

```

Программа У810 обращается к подпрограмме У410 и вспомогательной программе сложения СЧП и ЧП — программе С4410:

```

08F0                      ORG     8F0H
C4410:
; *****
; ПОДПРОГРАММА ДЕСЯТИЧНОГО СЛОЖЕНИЯ 4- И 2-РАЗРЯДНЫХ
; БЕЗЗНАКОВЫХ ДВОИЧНО-ДЕСЯТИЧНЫХ (КОД 8421) ЧИСЕЛ СО
; СДВИГОМ ВТОРОГО СЛАГАЕМОГО НА ТЕТРАДУ ВЛЕВО.
; ВХОДНЫЕ ПАРАМЕТРЫ: (A) - 2-РАЗРЯДНОЕ СЛАГАЕМОЕ, (H,L) -
; -4-РАЗРЯДНОЕ СЛАГАЕМОЕ. ВЫХОДНОЙ ПАРАМЕТР: (H,L) - ДВО-
; ИЧНО-ДЕСЯТИЧНАЯ СУММА. ИСПОЛЬЗУЮТСЯ РЕГИСТРЫ (A), (C).
; ОЦЕНКА: ДЛИНА-21 БАЙТ, ВРЕМЯ-107 ТАКТОВ.
; *****
08F0 C5      PUSH    B
; ВРАЩЕНИЕ ЦИФР 2-РАЗРЯДНОГО СЛАГАЕМОГО: (МЛЦ, СТЦ)
08F1 0F      RRC
08F2 0F      RRC
08F3 0F      RRC
08F4 0F      RRC
; ВЫДЕЛЕНИЕ СТЦ И МЛЦ СЛАГАЕМОГО
08F5 4F      MOV     C,A      ; (C) - СОХРАНЕНИЕ СЛАГАЕМОГО
08F6 E60F    ANI     0FH      ; МАСКА НА СТЦ
08F8 47      MOV     B,A      ; (B) - СТЦ
08F9 79      MOV     A,C      ; ВОССТАНОВЛЕНИЕ СЛАГАЕМОГО
08FA E6F0    ANI     0F0H     ; МАСКА НА МЛЦ
; ДВОИЧНО-ДЕСЯТИЧНОЕ СЛОЖЕНИЕ МЛЦ С СУММОЙ ЧП
08FC 85      ADD     L
08FD 27      DAA
08FE 6F      MOV     L,A

```

## ДВОИЧНО-ДЕСЯТИЧНОЕ СЛОЖЕНИЕ СЛЦ С СУММОЙ ЧП

08FF 78	MOV	A, B	# (A) — СЛЦ
0900 8C	ADC	H	
0901 27	DAA		
0902 67	MOV	H, A	
0903 C1	POP	B	
0904 C9	RET		
0000	END		

Сравнивая быстродействие и затраты памяти программы У810 с аналогичной по точности программой двоичного умножения У88А1, можно заметить трехкратное преимущество последней по быстродействию и пятикратное по экономии памяти. Быстродействие программ десятичного умножения можно существенно повысить (увеличив затраты памяти) за счет применения табличных методов умножения [20].

## 1.5. ДЕЛЕНИЕ ДВОИЧНЫХ ЧИСЕЛ

### 1.5.1. МЕТОДИКА ДЕЛЕНИЯ

Деление — операция, обратная умножению: нахождение одного из сомножителей (частного) по произведению (делимому) и второму сомножителю (делителю). С другой стороны, операцию деления можно рассматривать как умножение делимого на величину, обратную делителю. В связи с этим для чисел ограниченной точности *границная относительная ошибка деления* определяется выражением (1.18), и, следовательно, значность частного не может быть больше значности наименее точного из делимых чисел. Очевидно, что если делимое и делитель представляются  $n$ -разрядными дробными числами, то частное также не должно содержать более  $n$  разрядов.

Произведение точных целых  $n$ -разрядных чисел имеет разрядность  $2n$ . Поэтому при делении таких чисел прием разрядности делимого и делителя соответственно  $2n$  и  $n$ . Как отмечалось в § 1.1, множество целых чисел незамкнуто относительно операции деления, т. е. частное может быть не только точным целым числом, но и конечной или бесконечной дробью (правильной или неправильной), т. е. числом ограниченной точности. В связи с этим разрядность частного при делении целых чисел определяется необходимой точностью его вычисления и может превышать разрядность делимого или делителя. На практике деление целых чисел выполняется как *деление с остатком*: находятся целое неполное точное частное

(его произведение с делителем дает целое число, не превышающее делимое) и целый остаток, т. е. разность между делимым и произведением неполного частного на делитель. Разрядность неполного частного не превосходит разрядности делимого, а разрядность остатка — разрядности делителя (остаток по абсолютной величине всегда меньше делителя).

Микропроцессор КР580 не содержит команд деления чисел, поэтому для выполнения этой операции необходимы программы. Любой программный метод деления сводится к последовательному нахождению цифр частного, начиная с его старшей цифры, путем вычитания делителя из делимого или остатка и анализа получаемой разности. *Двоичное деление* проще десятичного, поскольку выбор очередной цифры частного производится всего из двух цифр: 0 и 1, причем цифра 1 выбирается при неотрицательной разности (делимое по модулю больше или равно делителю), а цифра 0 — при отрицательной разности (делимое по модулю меньше делителя) [5, 32, 33, 61]. Процедура обычного (неускоренного) деления носит последовательный характер: очередная цифра частного и новый остаток не могут быть вычислены прежде, чем будет получен и исследован предыдущий остаток.

Поскольку частное в отличие от произведения можно получать только со старших его цифр, имеются две *вычислительные схемы деления* (рис. 1.8). Схема 1 (рис. 1.8, а) аналогична схеме 3 умножения (см. рис. 1.5, в), а схема 2 (рис. 1.8, б) — схеме 4 умножения (см. рис. 1.5, г). На схемах рис. 1.8 обозначены: ДМ — делимое; ДЛ — делитель; ЧСТ — частное; ОСТ — остаток (в скобках указаны разрядности данных). Цифры частного формируются по значению признака переноса СУ, устанавливаемого в операциях вычитания делителя из делимого или остатка, и сдвигаются влево. При этом в схеме 1 для получения нового остатка предыдущий остаток сдвигается влево при неподвижном делителе, а в схеме 2, наоборот, при неподвижном остатке сдвигается вправо делитель. Схема 1 проще и не требует команд сдвига вправо многобайтных данных (в микропроцессоре КР580 такие команды отсутствуют). Поэтому рассматриваемые ниже программы деления все выполнены по схеме 1.

Есть два способа получения цифр частного и остатка:

1) деление с восстановлением остатка; 2) деление без вос-

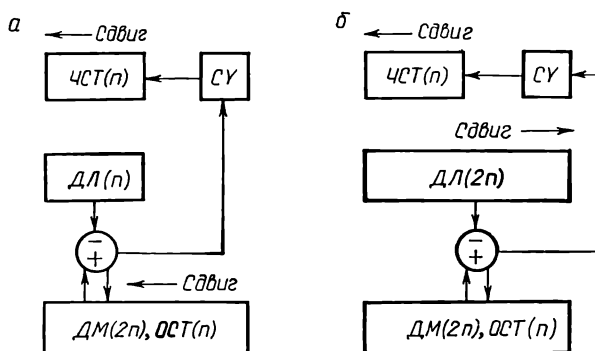


Рис. 1. 8. Вычислительные схемы операции деления

становления остатка. Рассмотрим способ *деления с восстановлением остатка* для класса целых беззнаковых двоичных чисел (он применим и для дробных беззнаковых чисел). Пусть  $X$ ,  $Y$ ,  $Z$ ,  $Q$  — целые беззнаковые ( $X$ ,  $Y$ ,  $Q$  —  $n$ -разрядные числа, а  $Z$  —  $2n$ -разрядные) числа, такие, что  $X \cdot Y + Q = Z$ , где  $Q < X$ . Тогда справедливы выражения:

$$Z:Y \approx X; Z:Y = (X, Q),$$

где  $X$  — неполное частное;  $Q$  — остаток. Если  $Z < Y$ , то  $X = 0$ ,  $Q = Z$ ; если  $Z = Y$ , то  $X = 1$ ,  $Q = 0$ ; если  $Z = X \cdot Y$ , то  $Q = 0$ .

Обозначим через  $Q_i$  разность между остатком  $Q_{i-1}$  и делителем, где  $i \in \{1, 2, \dots, n\}$ ;  $Q_0 = Z$ , а через  $x_j$  — цифру частного  $X$ , причем  $j = n - i$ ,  $j \in \{0, 1, \dots, n - 1\}$ . Тогда процедуру деления с восстановлением остатка можно записать в виде рекуррентной формулы

$$Q_i, x_{n-i} = \begin{cases} 2Q_{i-1} - Y; & x_{n-i} = 1 \text{ при } 2Q_{i-1} - Y \geq 0; \\ 2Q_{i-1}; & x_{n-i} = 0 \text{ при } 2Q_{i-1} - Y < 0. \end{cases} \quad (1.25)$$

Эта процедура состоит из  $n$  циклов деления, причем в каждом  $i$ -м цикле осуществляется сдвиг влево на один двоичный разряд остатка  $Q_{i-1}$ , полученного во время выполнения предыдущего  $(i-1)$ -го цикла, и вычисление разности между удвоенным остатком и делителем:  $2Q_{i-1} - Y$ . Если разность положительна, то в очередной  $(n-i)$ -й разряд частного записывается цифра  $x_{n-i} = 1$ ,



а разность становится новым остатком  $Q_i$ . Если же разность отрицательна, то в разряд частного записывается  $x_{n-i} = 0$ , а остаток  $2Q_{i-1}$  восстанавливается (либо путем сложения  $2Q_{i-1} - Y + Y$ , либо извлечением из области его сохранения) и интерпретируется как новый остаток  $Q_i$ . После выполнения  $n$ -го цикла деления сформированы все  $n$  старшие цифры частного и получен остаток  $Q = Q_n < X$ . Этот остаток можно использовать для повторного деления с целью вычисления дробной части частного и повышения тем самым точности его вычисления. Многократно повторяя последнюю операцию, можно вычислить частное с любой наперед заданной точностью.

Способ деления без восстановления остатка определяется выражениями:

$$Q_i = \begin{cases} 2Q_{i-1} - Y, & \text{если } Q_{i-1} \geq 0; \\ 2Q_{i-1} + Y, & \text{если } Q_{i-1} < 0; \end{cases}$$

$$x_{n-i} = \begin{cases} 1, & \text{если } Q_i \geq 0; \\ 0, & \text{если } Q_i < 0. \end{cases}$$

В этой процедуре при получении отрицательной разности  $Q_{i-1} = 2Q_{i-2} - Y$  последняя не восстанавливается, как в формуле (1.25):  $Q_{i-1} = 2Q_{i-2}$ , а сдвигается в следующем,  $i$ -м цикле влево, и к ней прибавляется делитель:  $Q_i = 2Q_{i-1} + Y$ .

Преобразование  $Q_i = 2Q_{i-1} + Y = 2(2Q_{i-2} - Y) + Y = 2(2Q_{i-2} - Y)$  показывает эквивалентность обоих способов. При способе с восстановлением остатка в каждом цикле производятся одна либо две операции: вычитание или вычитание и сложение (при восстановлении), а при втором способе — только одна операция: вычитание или сложение. Поэтому второй способ имеет потенциально большее быстродействие, но это его преимущество проявляется лишь при аппаратных реализациях деления. Программная реализация второго способа сложнее, что связано с необходимостью хранения знака остатка до начала следующего цикла деления с целью определения типа выполняемой операции (сложение или вычитание делителя) и коррекции конечного остатка  $Q_n$  в случае, если  $Q_{n-1} < 0$  (необходимо вычесть делитель). Вследствие этого деление без восстановления остатка редко используется в программах (в иллюстративных целях далее будет приведена программа деления Д16А, выполняемого по этому способу).

Последовательность действий при делении с восстановлением и без восстановления остатка поясняется на примере деления двоичных чисел  $00100000:1111 = (0010, 0010)$  соответственно в табл. 1.14 и 1.15, в которых приняты следующие обозначения операций:  $\leftarrow Q_i$  — сдвиг влево остатка;  $\mp Y$  — суммирование или вычитание делителя;  $= Q_i$  — присвоение значения остатку. Заметим, что остаток считается положительным при делении с восстановлением остатка, если в результате вычитания  $2Q_{i-1} - Y$  признак переноса  $CY = 0$ , а при делении без восстановления остатка, если признак переноса  $CY$  не меняет своего значения между операциями  $2Q_{i-1}$  и  $2Q_{i-1} \pm Y$ , т. е.  $CY: 1 \rightarrow 1$  или  $CY: 0 \rightarrow 0$ .

При выполнении деления  $2n$ -разрядного делимого на  $n$ -разрядный делитель возможно *переполнение*  $n$ -разрядного формата *частного*: появление абсолютной величины, непредставимой в  $n$ -разрядном формате. Очевидно, это имеет место в случае, когда число в старших  $n$  разрядах делимого больше или равно делителю. Переполнение необходимо выявлять программным способом и блоки-

Табл. 1.14. Деление с восстановлением остатка

$i$	Остаток		$x_{4-i}$	Тип операции
	$CY$	$Q_i$		
1	0	01000000 1111	0	$\leftarrow Q_0$ $-Y$
	1	01010000 1111		$= Q_1 < 0$ $+Y$
	1	01000000		$= Q_1 > 0$
2	0	10000000 1111	0	$\leftarrow Q_1$ $-Y$
	1	10010000 1111		$= Q_2 < 0$ $+Y$
	1	10000000		$= Q_2 > 0$
3	1	00000000 1111	1	$\leftarrow Q_2$ $-Y$
	0	00010000		$= Q_3 > 0$
4	0	00100000 1111	0	$\leftarrow Q_3$ $-Y$
	1	00110000 1111		$= Q_4 < 0$ $+Y$
	1	00100000		$= Q_4 > 0$

Табл. 1.15. Деление без восстановления остатка

$i$	Остаток		$x_{4-i}$	Тип операции
	СУ	$Q_i$		
1	0	01000000 1111	0	$\leftarrow Q_0$ $-Y$
	1	01010000		$=Q_1 < 0$
2	0	10100000 1111	0	$\leftarrow Q_1$ $+Y$
	1	10010000		$=Q_2 < 0$
3	1	00100000 1111	1	$\leftarrow Q_2$ $+Y$
	1	00010000		$=Q_3 > 0$
4	0	00100000 1111	0	$\leftarrow Q_3$ $-Y$
	1	00110000 1111		$=Q_4 < 0$ $+Y$
	1	00100000		$=Q_4 > 0$

ровать дальнейшие вычисления вследствие их некорректности. Избежать переполнения можно за счет увеличения разрядности частного до разрядности делимого (но при этом в два раза увеличивается и количество циклов деления). Помимо выявления переполнения, при делении чисел необходимо выявлять ситуацию деления на нуль. Заметим, что при анализе переполнения деления целых чисел автоматически выявляется и деление на нуль.

## 1.5.2. ЦЕЛЫЕ БЕЗЗНАКОВЫЕ ЧИСЛА

### 1.5.2.1. Формат 16:8 = [8,8]

Программа Д16 реализует деление с восстановлением остатка и с размещением сдвигаемых влево разрядов частного в освобождаемых младших разрядах делимого:

0A00

ORG

0A00H

D16:

```

;*****
;ПОДПРОГРАММА ДЕЛЕНИЯ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ БЕЗ ЗНАКА
;ФОРМАТА 16:8=(8,8).
;МЕТОД ДЕЛЕНИЯ С ВОССТАНОВЛЕНИЕМ ОСТАТКА.
;ВХОДНЫЕ ПАРАМЕТРЫ: (H,L) - ДЕЛИМОЕ, (C) - ДЕЛИТЕЛЬ. ВЫХОД-
;НЫЕ ПАРАМЕТРЫ: (H) - ОСТАТОК, (L) - ЧАСТНОЕ, СУ=0 - ПРИЗНАК

```

```

; ПЕРЕПОЛНЕНИЯ ЧАСТНОГО. ИСПОЛЪЗУЮТСЯ ВСЕ РЕГИСТРЫ,
; КРОМЕ (D,E); СОХРАНЯЕТСЯ (C).
; ОЦЕНКА: ДЛИНА-28 БАЙТ, ВРЕМЯ-НЕ БОЛЕЕ 619 ТАКТОВ.
; *****
; ПРОВЕРКА ЧАСТНОГО НА ПЕРЕПОЛНЕНИЕ: ЧАСТНОЕ > В БИТ?
0A00 7C      MOV     A,H
0A01 91      SUB     C
0A02 D0      RNC           ; ЕСЛИ ПЕРЕПОЛНЕНИЕ, CY=0
0A03 060B    MVI     B,B      ; СЧЕТЧИК ЦИКЛОВ
; СДВИГ ВЛЕВО ОСТАТКА И ЧАСТНОГО В (H,L)
0A05 29      DAD     H      ЦИКЛ:
0A06 7C      MOV     A,H      ; (A)-СТВ ОСТАТКА
0A07 DA130A   JC      PER1    ; ЕСЛИ ПЕРЕПОЛНЕНИЕ ОСТАТКА
; ВЫЧИТАНИЕ ДЕЛИТЕЛЯ ИЗ ОСТАТКА
0A0A 91      SUB     C
0A0B D2140A   JNC     PER2    ; ЕСЛИ РАЗНОСТЬ > 0
; РАЗНОСТЬ < 0, ВОССТАНОВЛЕНИЕ ОСТАТКА
0A0E 81      ADD     C
0A0F 67      MOV     H,A
0A10 C3160A   JMP     PER3
; ПЕРЕПОЛНЕНИЕ ОСТАТКА, РАЗРЯД ЧАСТНОГО=1
0A13 91      PER1: SUB     C
0A14 67      PER2: MOV     H,A
0A15 23      INX     H      ; +1 В ЧАСТНОЕ
; ПРОВЕРКА КОНЦА ЦИКЛА
0A16 05      PER3: DCR     B
0A17 C2050A   JNZ     ЦИКЛ   ; ЗАЦИКЛИВАНИЕ
0A1A 37      STC           ; CY=1
0A1B C9      RET
0000      END

```

Максимальная длительность цикла деления равна 73, время программы  $T \leq 35 + 73 \times 8 = 619$  тактам.

Программа Д16А в отличие от предыдущей и последующих программ деления реализует способ деления без восстановления остатка:

```

0A20      ORG     0A20H
Д16А:
; *****
; ПОДПРОГРАММА ДЕЛЕНИЯ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ БЕЗ ЗНАКА
; ФОРМАТА 16:8=(8,8).
; МЕТОД ДЕЛЕНИЯ БЕЗ ВОССТАНОВЛЕНИЯ ОСТАТКА.
; ВХОДНЫЕ ПАРАМЕТРЫ: (H,L) - ДЕЛИМОЕ, (C) - ДЕЛИТЕЛЬ. ВЫХОД-
; НЫЕ ПАРАМЕТРЫ: (H) - ОСТАТОК, (L) - ЧАСТНОЕ, CY=0 - ПРИЗНАК
; ПЕРЕПОЛНЕНИЯ ЧАСТНОГО. ИСПОЛЪЗУЮТСЯ ВСЕ РЕГИСТРЫ,
; КРОМЕ (D,E); СОХРАНЯЕТСЯ (C).
; ОЦЕНКА: ДЛИНА-58 БАЙТ, ВРЕМЯ-НЕ БОЛЕЕ 850 ТАКТОВ.
; *****
; ПРОВЕРКА ЧАСТНОГО НА ПЕРЕПОЛНЕНИЕ: ЧАСТНОЕ > В БИТ?
0A20 7C      MOV     A,H
0A21 91      SUB     C
0A22 D0      RNC           ; ЕСЛИ ПЕРЕПОЛНЕНИЕ, CY=0
0A23 060B    MVI     B,B      ; СЧЕТЧИК ЦИКЛОВ
0A25 37      STC           ; CY=1-ПРИЗНАК ОСТАТКА = ИЛИ 0

```

		;ПРОВЕРКА ЗНАКА ОСТАТКА: "+", ЕСЛИ CY=1; "-", ЕСЛИ CY=0	
0A26 D23B0A	ЦИКЛ:	JNC	ПЕР1 ;ЕСЛИ ОСТАТОК < 0
0A29 29		DAD	H ;СДВИГ ВЛЕВО ОСТАТКА
0A2A 7C		MOV	A,H
0A2B DA360A		JC	ПЕР2 ;ЕСЛИ ПЕРЕПОЛНЕНИЕ ОСТАТКА
		;ВЫЧИТАНИЕ ДЕЛИТЕЛЯ ИЗ ОСТАТКА	
0A2E 91		SUB	C
0A2F 67		MOV	H,A
0A30 D24A0A	ПЕР6:	JNC	ПЕР3 ;ЕСЛИ РЕЗУЛЬТАТ=ИЛИ > 0
0A33 C34F0A		JMP	ПЕР4 ;РЕЗУЛЬТАТ < 0
		;ВЫЧИТАНИЕ ДЕЛИТЕЛЯ ИЗ ОСТАТКА	
0A36 91	ПЕР2:	SUB	C
0A37 67		MOV	H,A
0A38 C34A0A		JMP	ПЕР3 ;РЕЗУЛЬТАТ= ИЛИ > 0
		;СДВИГ ВЛЕВО ОСТАТКА И ЧАСТНОГО В (H,L) ПРИ ОСТАТКЕ<0	
0A3B 29	ПЕР1:	DAD	H
0A3C 7C		MOV	A,H
0A3D DA450A		JC	ПЕР5 ;ЕСЛИ ПЕРЕПОЛНЕНИЕ ОСТАТКА
		;СЛОЖЕНИЕ ДЕЛИТЕЛЯ С ОТРИЦАТЕЛЬНЫМ ОСТАТКОМ	
0A40 81		ADD	C
0A41 67		MOV	H,A
0A42 C3300A		JMP	ПЕР6
		;СЛОЖЕНИЕ ДЕЛИТЕЛЯ С ОТРИЦАТЕЛЬНЫМ ОСТАТКОМ	
0A45 81	ПЕР5:	ADD	C
0A46 67		MOV	H,A
0A47 D24F0A		JNC	ПЕР4 ;ЕСЛИ СУММА < 0
		;УСТАНОВКА РАЗРЯДА ЧАСТНОГО В "1"	
0A4A 23	ПЕР3:	INX	H
0A4B 37		STC	;CY=1-ПОЛОЖИТЕЛЬНЫЙ ОСТАТОК
0A4C C3500A		JMP	ПЕР7
0A4F AF	ПЕР4:	XRA	A ;CY=0-ОТРИЦАТЕЛЬНЫЙ ОСТАТОК
		;ПРОВЕРКА КОНЦА ЦИКЛА	
0A50 05	ПЕР7:	DCR	B
0A51 C2260A		JNZ	ЦИКЛ ;ЗАЦИКЛИВАНИЕ
0A54 D8		RC	;ЕСЛИ ПОЛОЖИТЕЛЬНЫЙ ОСТАТОК
		;ВОССТАНОВЛЕНИЕ ОСТАТКА	
0A55 7C		MOV	A,H
0A56 81		ADD	C
0A57 67		MOV	H,A
0A58 37		STC	;CY=1
0A59 C9		RET	

Программа имеет много команд ветвления в связи с необходимостью выполнения альтернативных операций сложения и вычитания делителя из остатка и последующей проверки каждой из этих операций на знак результата. Затраты памяти и время выполнения этой программы значительно больше, чем программы Д16, что иллюстрирует недостатки программной реализации метода деления без восстановления остатка (по крайней мере для рассматриваемого типа микрокалькулятора).

Тестовые данные для программ приведены в табл. 1.16.

Табл. 1.16. Тесты деления формата 16:8=(8,8)  
(целые беззнаковые двоичные числа)

Представление чисел	
шестнадцатеричное	десятичное
FE00:FF=(FE, FE)	65024:255=(254, 254)
0100:FF=(01, 01)	256:255=(1,1)
0E10:F0=(0F, 00)	3600:240=(15,0)
3874:AA=(55, 02)	14452:170=(85,2)
3F00:7F=(7E, 7E)	16128:127=(126, 126)

### 1.5.2.2. Формат 16:8=(16,8)

В программах Д16 и Д16А СТБ делимого должен быть обязательно меньше делителя, иначе произойдет переполнение формата частного. В ряде приложений такие ограничения неприемлемы. Программа Д1616 устраняет эти ограничения и позволяет находить частное при превышении значением делимого значения делителя:

ОБС0	ORG	ОБСОН
	Д1616:	
	;*****	
	;ПОДПРОГРАММА ДЕЛЕНИЯ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ БЕЗ ЗНАКА	
	;ФОРМАТА 16:8=(16,8).	
	;ВХОДНЫЕ ПАРАМЕТРЫ: (Н, L) -ДЕЛИМОЕ, (С) -ДЕЛИТЕЛЬ. ВЫХОД-	
	;НЫЕ ПАРАМЕТРЫ: (D, E) -ЧАСТНОЕ, (Н) -ОСТАТОК, CY=1-ПРИЗНАК	
	;НУЛЕВОГО ДЕЛИТЕЛЯ ИЛИ ПРЕВЫШЕНИЯ ВЕЛИЧИНЫ ДЕЛИТЕЛЯ	
	;НАД ДЕЛИМЫМ. ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ, ГЛУБИНА СТЕКА=2.	
	;ОЦЕНКА: ДЛИНА=88 БАЙТ, ВРЕМЯ=НЕ БОЛЕЕ 1831 ТАКТОВ.	
	;*****	
	;ПРОВЕРКА: ДЕЛИМОЕ > ДЕЛИТЕЛЯ ?	
ОБС0 AF	XRA	A
ОБС1 84	ADD	H
ОБС2 C2C80B	JNZ	ПЕР1
ОБС5 85	ADD	L
ОБС6 B9	CMF	C
ОБС7 D8	RC	
	;ЕСЛИ ДЕЛИМОЕ < ДЕЛИТЕЛЯ	
	;ПРОВЕРКА ДЕЛИТЕЛЯ НА 0	
ОБС8 AF	ПЕР1: XRA	A
ОБС9 81	ADD	C
ОБСА 37	STC	
ОБСВ С8	RZ	
ОБСС 0608	MVI	В, 8
	;НОРМАЛИЗАЦИЯ ДЕЛИТЕЛЯ: УСТРАНЕНИЕ ЛЕВЫХ НУЛЕЙ	
ОБСЕ 3F	CMC	
ОБСF 17	ЦИКЛ1: RAL	
ОБD0 04	INR	В
ОБD1 D2CF0B	JNC	ЦИКЛ1
ОБD4 1F	RAR	
ОБD5 4F	MOV	С, А

OBD6 05	DCR	B	;УМЕНЬШЕНИЕ СЧЕТЧИКА
	;СОХРАНЕНИЕ ВЕЛИЧИНЫ СДВИГОВ ДЕЛИТЕЛЯ		
OBD7 78	MOV	A,B	
OBD8 DE08	SBI	8	
OBD9 F5	PUSH	PSW	
	;НОРМАЛИЗАЦИЯ ДЕЛИМОГО:УСТРАНЕНИЕ ЛЕВЫХ НУЛЕЙ		
OBD9 AF	ЦИКЛ2:	XRA	A
OBD8 84		ADD	H
			; (A)-СТБ ДЕЛИМОГО
OBD0 FAE50B	JM	ПЕР2	;ЕСЛИ СТАРШИЙ РАЗРЯД=1
OBE0 29	DAD	H	;СДВИГ ДЕЛИМОГО ВЛЕВО
OBE1 05	DCR	B	;УМЕНЬШЕНИЕ СЧЕТЧИКА
OBE2 C3DB0B	JMP	ЦИКЛ2	;ЗАЦИКЛИВАНИЕ
OBE5 110000	ПЕР2:	LXI	D,0
			;ОБНУЛЕНИЕ ЧАСТНОГО
			;ВЫЧИТАНИЕ ДЕЛИТЕЛЯ ИЗ ДЕЛИМОГО ((A)-СТБ ДЕЛИМОГО)
OBE8 91		SUB	C
OBE9 13		INX	D
			;УСТАНОВКА РАЗРЯДА ЧАСТНОГО=1
OBEA D2EF0B	JNC	ПЕР3	;ЕСЛИ РАЗНОСТЬ > 0
OBED 1B	DCX	D	;СВРОС РАЗРЯДА ЧАСТНОГО=0
OBE6 81		ADD	C
			;ВОССТАНОВЛЕНИЕ ДЕЛИМОГО
OBEF 67	ПЕР3:	MOV	H,A
			;ПРОВЕРКА СЧЕТЧИКА ЦИКЛОВ НА 0
OBFO AF	XRA	A	
OBF1 80	ADD	B	
OBF2 CA0D0C	JZ	КОН	;ЕСЛИ СЧЕТЧИК=0
			;СДВИГ ВЛЕВО ЧАСТНОГО (D,E) И ОСТАТКА (H,L)
OBF5 EB	ЦИКЛ:	XCHG	
OBF6 29		DAD	H
OBF7 EB		XCHG	
OBF8 29		DAD	H
OBF9 7C		MOV	A,H
			; (A)-СТБ ОСТАТКА
OBFA DA060C	JC	ПЕР4	;ЕСЛИ ПЕРЕПОЛНЕНИЕ ОСТАТКА
			;ВЫЧИТАНИЕ ДЕЛИТЕЛЯ ИЗ ОСТАТКА
OBFD 91		SUB	C
OBFE D2070C		JNC	ПЕР5
			;ЕСЛИ РАЗНОСТЬ > 0
			;РАЗНОСТЬ < 0,ВОССТАНОВЛЕНИЕ ОСТАТКА
OC01 81		ADD	C
OC02 67		MOV	H,A
OC03 C3090C		JMP	ПЕР6
			;ПЕРЕПОЛНЕНИЕ ОСТАТКА,УСТАНОВКА РАЗРЯДА ЧАСТНОГО=1
OC06 91	ПЕР4:	SUB	C
			;ВЫЧИТАНИЕ ДЕЛИТЕЛЯ
OC07 67	ПЕР5:	MOV	H,A
OC08 13		INX	D
			;РАЗРЯД ЧАСТНОГО=1
			;ПРОВЕРКА КОНЦА ЦИКЛА
OC09 05	ПЕР6:	DCR	B
OC0A C2F50B		JNZ	ЦИКЛ
			;ЗАЦИКЛИВАНИЕ
			;КОРРЕКЦИЯ ОСТАТКА:СДВИГ ВПРАВО
OC0D F1	КОН:	POP	PSW
OC0E CB		RZ	
			; (A)-СЧЕТЧИК СДВИГОВ
OC0F 4F		MOV	C,A
OC10 AF		XRA	A
OC11 84		ADD	H
			; (A)-ОСТАТОК
OC12 1F	ЦИКЛ3:	RAR	
OC13 0D		DCR	C
OC14 C2120C		JNZ	ЦИКЛ3
			;ЗАЦИКЛИВАНИЕ
OC17 67		MOV	H,A
OC18 C9		RET	
0000		END	

В основу программы положен алгоритм, описанный в работе [49]. Программа выполняет деление, если делимое не меньше делителя и делитель не равен нулю. Если деление возможно, программа устраняет незначащие нули делимого и делителя путем сдвига этих чисел влево с одновременной фиксацией разности их сдвигов. Например, если делимое и делитель имеют по 8 значащих цифр, то после сдвига делимого влево на 8 разрядов счетчик циклов деления в регистре (В) имеет значение  $8 + 0 - 8 = 0$ , и деление практически выполнять не надо (достаточно вычесть делитель из делимого). Если же делимое имеет 16, а делитель — одну значащие цифры, то после нормализации чисел счетчик циклов в регистре (В) имеет значение  $8 + 7 - 0 = 15$ , и в этом случае потребуется 15 циклов деления для получения результата. После окончания деления остаток отделяется от частного и приводится к своему истинному значению путем сдвига вправо на величину начального сдвига делителя влево. Расширение диапазона обрабатываемых чисел в программе Д1616 требует дополнительных затрат памяти и времени по сравнению с программами Д16 и Д16А. Тестовые данные для программы приведены в табл. 1.17.

**Табл. 1.17. Тесты деления формата 16:8=(16, 16)  
(целые беззнаковые двоичные числа)**

Представление чисел	
шестнадцатеричное	десятичное
FFFF:FF=(0101, 00)	65535:255=(257,0)
FFFF:F0=(0111, 0F)	65535:240=(273, 15)
FFFF:0F=(1111, 00)	65535:15=(4369, 0)
5555:AA=(0080, 55)	21845:170=(128, 85)
AAAA:55=(0202, 00)	43690:85=(514, 0)

### **1.5.2.3. Формат 24:16=[8,16]**

Программа Д24 реализует деление с предварительным преобразованием делителя в дополнительный код и восстановлением остатка путем сохранения и извлечения его из стека (аналогичная программа приведена в работе [21]):



0A60		ORG	0A60H
0050	ДОПВ	SET	50H

Д24:  
 ;\*\*\*\*\*  
 ;ПОДПРОГРАММА ДЕЛЕНИЯ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ БЕЗ ЗНАКА  
 ;ФОРМАТА 24:16=(8,16).  
 ;МЕТОД ДЕЛЕНИЯ С ВОССТАНОВЛЕНИЕМ ОСТАТКА.  
 ;ВХОДНЫЕ ПАРАМЕТРЫ: (H,L,E)-ДЕЛИМОЕ, (B,C)-ДЕЛИТЕЛЬ.  
 ;ВЫХОДНЫЕ ПАРАМЕТРЫ: (E)-ЧАСТНОЕ, (H,L)-ОСТАТОК, CY=0-  
 ;-ПРИЗНАК ПЕРЕПОЛНЕНИЯ ЧАСТНОГО.ИСПОЛЗУЮТСЯ ВСЕ  
 ;РЕГИСТРЫ,КРОМЕ D,ГЛУБИНА СТЕКА-2.ИСПОЛЬЗУЕТСЯ  
 ;ПОДПРОГРАММА \*ДОПВ\*.  
 ;ОЦЕНКА:ДЛИНА-48 БАЙТ,ВРЕМЯ-НЕ БОЛЕЕ 994 ТАКТА.  
 ;\*\*\*\*\*  
 ;ПРОВЕРКА ЧАСТНОГО НА ПЕРЕПОЛНЕНИЕ:ЧАСТНОЕ > 8 БИТ?

0A60	7D	MOV	A,L	
0A61	91	SUB	C	
0A62	7C	MOV	A,H	
0A63	98	SBB	B	
0A64	D0	RNC		;ЕСЛИ ПЕРЕПОЛНЕНИЕ,CY=0
		;ДОПОЛНЕНИЕ ДЕЛИТЕЛЯ		
0A65	CD5000	CALL	ДОПВ	; (B,C)-ДОПОЛНИТЕЛЬНЫЙ КОД
0A68	7B	MOV	A,E	; (A)-МЛБ ДЕЛИМОГО
0A69	1E08	MVI	E,8	;СЧЕТЧИК ЦИКЛОВ
		;СДВИГ ОСТАТКА И ЧАСТНОГО ВЛЕВО В (H,L,A)		
0A6B	29	ЦИКЛ: DAD	H	
0A6C	DA810A	JC	ПЕР1	;ЕСЛИ ПЕРЕПОЛНЕНИЕ ОСТАТКА
0A6F	87	ADD	A	
0A70	D2740A	JNC	ПЕР2	
0A73	23	INX	H	;УЧЕТ ПЕРЕНОСА
		;СЛОЖЕНИЕ ОСТАТКА С ДОПОЛНИТЕЛЬНЫМ КОДОМ ДЕЛИТЕЛЯ		
0A74	E5	ПЕР2: PUSH	H	;СОХРАНЕНИЕ ОСТАТКА
0A75	09	DAD	B	
0A76	DA8A0A	JC	ПЕР3	;ЕСЛИ СУММА > 0
		;СУММА < 0,ВОССТАНОВЛЕНИЕ ОСТАТКА		
0A79	E1	POP	H	
		;ПРОВЕРКА КОНЦА ЦИКЛА		
0A7A	1D	ПЕР5: DCR	E	
0A7B	C26B0A	JNZ	ЦИКЛ	;ЗАЦИКЛИВАНИЕ
0A7E	5F	MOV	E,A	; (E)-ЧАСТНОЕ
0A7F	37	STC		;CY=1
0A80	C9	RET		
		;ПЕРЕПОЛНЕНИЕ ОСТАТКА.РАЗРЯД ЧАСТНОГО=1		
0A81	8F	ПЕР1: ADC	A	;СДВИГ,+1 В ЧАСТНОЕ
0A82	D2860A	JNC	ПЕР4	
0A85	23	INX	H	;УЧЕТ ПЕРЕНОСА
0A86	09	ПЕР4: DAD	B	;ОЧЕРЕДНОЙ ОСТАТОК
0A87	C37A0A	JMP	ПЕР5	
		;СУММА > 0.РАЗРЯД ЧАСТНОГО=1		
0A8A	33	ПЕР3: INX	SP	
0A8B	33	INX	SP	;БАЛАНС СТЕКА
0A8C	3C	INR	A	;+1 В ЧАСТНОЕ
0A8D	C37A0A	JMP	ПЕР5	
0000		END		

В программе использован прием баланса стека с помощью команд изменения непосредственно указателя стека INX SP вместо обычно применяемых команд извлечения из стека типа POP. Такой прием полезен в случаях, когда все регистры заняты, но необходимо реализовать баланс стека. Тестовые данные программы приведены в табл. 1.18.

**Табл. 1.18. Тесты деления формата 16:8=(16, 16)  
(целые беззнаковые двоичные числа)**

Представление чисел	
шестнадцатеричное	десятичное
FFFF01:FFFF=(FF, 0000)	16711425:65535=(255, 0)
0E0FFF:F000=(0E, EFFF)	921599:61440=(14, 61439)
38AA73:5555=(AA, 0001)	3713651:21845=(170, 1)
FE01FF:FF00=(FF, 00FF)	16646655:65280=(255, 255)
FE00FF:F000=(FE, FEFF)	16646399:65280=(254, 65279)

#### 1.5.2.4. Формат 32:16=(16,16)

Программа Д32, подобно программе Д16, реализует деление по классической схеме с вычитанием делителя, восстановлением остатка путем его сложения с делителем и размещением частного на месте младших байтов делимого (аналогичная программа приведена в работе [71]):

```

0A90                                ORG      0A90H
                                D32:
                                ;*****
                                ;ПОДПРОГРАММА ДЕЛЕНИЯ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ БЕЗ ЗНАКА
                                ;ФОРМАТА 32:16=(16,16).
                                ;МЕТОД ДЕЛЕНИЯ С ВОССТАНОВЛЕНИЕМ ОСТАТКА.
                                ;ВХОДНЫЕ ПАРАМЕТРЫ: (H,L,D,E)-ДЕЛИМОЕ, (B,C)-ДЕЛИТЕЛЬ.
                                ;ВЫХОДНЫЕ ПАРАМЕТРЫ: (D,E)-ЧАСТНОЕ, (H,L)-ОСТАТОК, CY=0-
                                ;-ПРИЗНАК ПЕРЕПОЛНЕНИЯ ЧАСТНОГО. ИСПОЛЗУЮТСЯ ВСЕ РЕ-
                                ;ГИСТРЫ, СОХРАНЯЕТСЯ (B,C). ГЛУБИНА СТЕКА-2.
                                ;ОШЕНКА: ДЛИНА-41 БАЙТ, ВРЕМЯ-НЕ БОЛЕЕ 2316 ТАКТОВ.
                                ;*****
                                ;ПРОВЕРКА ЧАСТНОГО НА ПЕРЕПОЛНЕНИЕ: ЧАСТНОЕ > 16 БИТ?
0A90 7D      MOV      A,L
0A91 91      SUB      C
0A92 7C      MOV      A,H
0A93 98      SBB      B
0A94 D0      RNC                      ;ЕСЛИ ПЕРЕПОЛНЕНИЕ, CY=0
0A95 3E10    MVI      A,16           ;СЧЕТЧИК ЦИКЛОВ
                                ;СДВИГ ВЛЕВО ОСТАТКА И ЧАСТНОГО В (H,L,D,E)
0A97 29      ЦИКЛ:  DAD      H
0A98 F5      PUSH     PSW           ;СОХРАНЕНИЕ СЧЕТЧИКА, ПЕРЕНОСА
0A99 EB      XCHG
0A9A 29      DAD      H

```

```

0A9B EB          XCHG
0A9C D2A00A      JNC     ПЕР1
0A9F 23          INX     H      ;УЧЕТ ПЕРЕНОСА
                        ;ВЫЧИТАНИЕ ДЕЛИТЕЛЯ ИЗ ОСТАТКА
0AA0 7D          ПЕР1: MOV     A,L
0AA1 91          SUB     C
0AA2 6F          MOV     L,A
0AA3 7C          MOV     A,H
0AA4 98          SBB     B
0AA5 67          MOV     H,A
                        ;ПРОВЕРКА ЗНАКА РАЗНОСТИ
0AA6 DAAE0A      JC      ПЕР2      ;ЕСЛИ РАЗНОСТЬ < 0
0AA9 F1          POP     PSW      ;ВОССТАНОВЛЕНИЕ СЧЕТЧИКА
0AAA 13          ПЕР3: INX     D      ;РАЗРЯД ЧАСТНОГО=1
0AAB C3B30A      JMP     ПЕР4
                        ;ПРОВЕРКА ПЕРЕНОСА ОТ СДВИГА ОСТАТКА
0AAE F1          ПЕР2: POP     PSW      ;ВОССТАНОВЛЕНИЕ ПЕРЕНОСА
0AAF DAAA0A      JC      ПЕР3      ;ЕСЛИ БЫЛ ПЕРЕНОС
                        ;ВОССТАНОВЛЕНИЕ ОСТАТКА
0AB2 09          DAD     B
                        ;ПРОВЕРКА КОНЦА ЦИКЛА
0AB3 3D          ПЕР4: DCR     A
0AB4 C2970A      JNZ     ЦИКЛ      ;ЗАЦИКЛИВАНИЕ
0AB7 37          STC
0AB8 C9          RET
0000            END

```

Из-за увеличения разрядности данных в программе Д32 необходимо выполнять в отличие от предыдущих программ 16 циклов деления, что существенно увеличивает общее время выполнения программы. Это время можно несколько уменьшить, если заменить в цикле операцию вычитания делителя из остатка сложением остатка с дополнительным кодом делителя. Эта модификация выполнена в программе Д32А:

```

0AC0            ORG     0AC0H
0050            ДОПВ   SET     50H

Д32А:
;*****
;ПОДПРОГРАММА ДЕЛЕНИЯ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ БЕЗ ЗНАКА
;ФОРМАТА 32:16=(16,16).
;ВХОДНЫЕ ПАРАМЕТРЫ: (H,L,D,E) -ДЕЛИМОЕ, (B,C) -ДЕЛИТЕЛЬ.
;ВЫХОДНЫЕ ПАРАМЕТРЫ: (D,E) -ЧАСТНОЕ, (H,L) -ОСТАТОК, CY=0-
;-ПРИЗНАК ПЕРЕПОЛНЕНИЯ ЧАСТНОГО.ИСПОЛЗУЮТСЯ ВСЕ РЕ-
;ГИСТРЫ,ГЛУБИНА СТЕКА-2.ИСПОЛЗУЕТСЯ ПОДПРОГРАММА
;ДОПВ*.
;ОЦЕНКА:ДЛИНА-46 (+8 *ДОПВ*) БАЙТ,ВРЕМЯ-НЕ БОЛЕЕ
;2087 ТАКТОВ.
;*****
;ПРОВЕРКА ЧАСТНОГО НА ПЕРЕПОЛНЕНИЕ: ЧАСТНОЕ > 16 БИТ?
0AC0 7D          MOV     A,L
0AC1 91          SUB     C
0AC2 7C          MOV     A,H
0AC3 98          SBB     B
0AC4 D0          RNC
                        ;ЕСЛИ ПЕРЕПОЛНЕНИЕ, CY=0

```

```

;ДОПОЛНЕНИЕ ДЕЛИТЕЛЯ
OAC5 CD5000      CALL ДОПВ      ; (В,С) - ДОПОЛНИТЕЛЬНЫЙ КОД
OACB AF          XRA A          ; (А)=0 - СЧЕТЧИК ЦИКЛОВ
;СДВИГ ВЛЕВО ОСТАТКА И ЧАСТНОГО В (Н, L, D, E)
OAC9 29          DAD H
OACA 1F          RAR            ;СОХРАНЕНИЕ ПЕРЕНОСА
OACB EB          XCHG
OACC 29          DAD H
OACD EB          XCHG
OACE D2D20A      JNC ПЕР1
OAD1 23          INX H          ;УЧЕТ ПЕРЕНОСА
;СЛОЖЕНИЕ ОСТАТКА С ДОПОЛНИТЕЛЬНЫМ КОДОМ ДЕЛИТЕЛЯ
OAD2 E5          ПЕР1: PUSH H    ;СОХРАНЕНИЕ ОСТАТКА
OAD3 09          DAD B
;ПРОВЕРКА ЗНАКА СУММЫ
OAD4 D2E20A      JNC ПЕР2      ;ЕСЛИ СУММА < 0
OAD7 17          RAL            ;ВОССТАНОВЛЕНИЕ ПЕРЕНОСА
OAD8 13          ПЕР3: INX D     ;РАЗРЯД ЧАСТНОГО=1
OAD9 33          INX SP
OADA 33          INX SP        ;БАЛАНС СТЕКА
;ПРОВЕРКА КОНЦА ЦИКЛА
OADB C610        ADI 16
OADD D2C90A      JNC ЦИКЛ      ;ЗАЦИКЛИВАНИЕ
OAE0 37          STC            ;CY=1
OAE1 C9          RET
;ПРОВЕРКА ПЕРЕНОСА ОТ СДВИГА ОСТАТКА
OAE2 17          ПЕР2: RAL      ;ВОССТАНОВЛЕНИЕ ПЕРЕНОСА
OAE3 DAD80A      JC ПЕР3
;ВОССТАНОВЛЕНИЕ ОСТАТКА, ПРОВЕРКА КОНЦА ЦИКЛА
OAE6 E1          POP H
OAE7 C610        ADI 16
OAE9 D2C90A      JNC ЦИКЛ      ;ЗАЦИКЛИВАНИЕ
OAE3 37          STC            ;CY=1
OAE0 C9          RET
0000            END

```

Программа Д32А на 10 % сокращает время деления, но требует на 30 % больше затрат памяти. Заметим, что в программе Д32А счетчик цикла выполнен по принципу счета до переполнения, что позволяет использовать его дополнительно для хранения признака переноса при сдвиге остатка влево в регистровой паре (Н, L). Тестовые данные для программ Д32 и Д32А приведены в табл. 1.19.

Табл. 1.19. Тесты деления формата 32:16=(16, 16)  
(целые беззнаковые двоичные числа)

Представление чисел	
шестнадцатеричное	десятичное
FFFE0001:FFFF=(FFFF, 0000)	4294836225:65535=(65535, 0)
FFFE0002:FFFF=(FFFF, 0001)	4294836226:65535=(65535, 1)
FFFDFFFF:FFFF=(FFFE, FFFD)	4294836223:65535=(65534, 65533)
38E31C74:AAAA=(5555, 0002)	954408052:43690=(21845, 2)
3FFF0000:7FFF=(7FFE, 7FFE)	1073676288:32767=(32766, 32766)

### 1.5.2.5. Формат 16:16= (16,16)

Программа Д162 в отличие от предыдущих программ выполняет деление целых чисел одинаковой разрядности, причем соотношение абсолютных величин делимого и делителя, которые ограничены в предыдущих программах, не имеет значения, т. е. делитель может быть как меньше, так и больше делимого. В любом случае программа дает неполное точное целое частное и целый остаток:

0AF0		ORG	0AF0H
	Д162:		
	;*****		
	;ПОДПРОГРАММА ДЕЛЕНИЯ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ БЕЗ ЗНАКА		
	;ФОРМАТА 16:16=(16,16) С ЦЕЛЫМ ЧАСТНЫМ.		
	;МЕТОД ДЕЛЕНИЯ С ВОССТАНОВЛЕНИЕМ ОСТАТКА.		
	;ВХОДНЫЕ ПАРАМЕТРЫ: (Н, L) - ДЕЛИМОЕ, (В, С) - ДЕЛИТЕЛЬ. ВЫХОД-		
	;НЫЕ ПАРАМЕТРЫ: (D, E) - ЧАСТНОЕ, (Н, L) - ОСТАТОК, CY=0 - ПРИЗ-		
	;НАК НУЛЕВОГО ДЕЛИТЕЛЯ. ИСПОЛЗУЮТСЯ ВСЕ РЕГИСТРЫ, СОХ-		
	;РАНЯЕТСЯ (В, С). ГЛУБИНА СТЕКА-2.		
	;ОЦЕНКА: ДЛИНА-35 БАЙТ, ВРЕМЯ-НЕ БОЛЕЕ 2052 ТАКТОВ.		
	;*****		
	;ПРОВЕРКА ДЕЛИТЕЛЯ НА 0		
0AF0 AF	XRA	A	
0AF1 B0	ORA	B	
0AF2 B1	ORA	C	
0AF3 C8	RZ		;ЕСЛИ ДЕЛИТЕЛЬ=0
0AF4 EB	XCHG		; (D, E) - ДЕЛИМОЕ
0AF5 210000	LXI	H, 0	; ОБНУЛЕНИЕ ОСТАТКА
0AFB 3E10	MVI	A, 16	; СЧЕТЧИК ЦИКЛОВ
	;СДВИГ ВЛЕВО ОСТАТКА (Н, L) И ДЕЛИМОГО (ЧАСТНОГО) (D, E)		
0AFA F5	PUSH	PSW	;СОХРАНЕНИЕ СЧЕТЧИКА
0AFB 29	DAD	H	
0AFC AF	XRA	A	
0AFD EB	XCHG		
0AFE 29	DAD	H	
0AFF EB	XCHG		
	;ВЫЧИТАНИЕ ДЕЛИТЕЛЯ ИЗ ОСТАТКА		
0B00 8D	ADC	L	;УЧЕТ ПЕРЕНОСА ОТ СДВИГА
0B01 91	SUB	C	
0B02 6F	MOV	L, A	
0B03 7C	MOV	A, H	
0B04 98	SBV	B	
0B05 67	MOV	H, A	
	;ПРОВЕРКА ЗНАКА РАЗНОСТИ И КОРРЕКЦИЯ ЧАСТНОГО		
0B06 13	INX	D	;УСТАНОВКА РАЗРЯДА ЧАСТНОГО=1
0B07 D20C0B	JNC	ПЕР	;ЕСЛИ РАЗНОСТЬ > 0
0B0A 09	DAD	B	;ВОССТАНОВЛЕНИЕ ОСТАТКА
0B0B 1B	DCX	D	;СБРОС РАЗРЯДА ЧАСТНОГО=0
	;ПРОВЕРКА КОНЦА ЦИКЛА		
0B0C F1	PER:	POP	PSW ;ВОССТАНОВЛЕНИЕ СЧЕТЧИКА
0B0D 3D		DCR	A
0B0E C2FA0A		JNZ	ЦИКЛ ;ЗАЦИКЛИВАНИЕ
0B11 37		STC	;CY=1

0B12 C9  
0000

RET  
END

По своей структуре программа аналогична программе Д32, но отличается от последней представлением полноформатного делимого в регистрах (Н, L, D, E) с двумя нулевыми старшими байтами, что облегчает анализ ситуаций. В результате программа Д162 проще и выполняется быстрее, чем Д32. Тестовые данные для программы приведены в табл. 1.20.

Табл. 1.20. Тесты деления формата 16:16=(16, 16)  
(целые беззнаковые двоичные числа)

Представление чисел	
шестнадцатеричное	десятичное
FFFF:00FF=(0101, 0000)	65535:255=(257, 0)
00FF:FFFF=(0000, 00FF)	255:65535=(0, 255)
FFFF:F000=(0001, 0FFF)	65535:61440=(1, 4095)
5555:07FF=(000A, 055F)	21845:2047=(10, 1375)
07FF:000A=(00CC, 0007)	2047:10=(204, 7)

Программа Д216 в отличие от Д162 и других вышеприведенных программ выполняет деление целых чисел, когда делимое заведомо меньше, чем делитель:

0B20  
0050

ДОПВ      ORG      0B20H  
             SET      50H

Д216:

```

;*****
;ПОДПРОГРАММА ДЕЛЕНИЯ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ БЕЗ ЗНАКА
;ФОРМАТА 16:16=(16,16) С ДРОБНЫМ ЧАСТНЫМ.
;МЕТОД ДЕЛЕНИЯ С ВОССТАНОВЛЕНИЕМ ОСТАТКА.
;ВХОДНЫЕ ПАРАМЕТРЫ: (Н, L) - ДЕЛИМОЕ, (В, С) - ДЕЛИТЕЛЬ. ВЫ-
;ХОДНЫЕ ПАРАМЕТРЫ: (D, E) - ЧАСТНОЕ, (Н, L) - ОСТАТОК, CY=0 - ПРИЗ-
;НАК ПЕРЕПОЛНЕНИЯ ЧАСТНОГО. ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ,
;ГЛУБИНА СТЕКА-2. ИСПОЛЬЗУЕТСЯ ПОДПРОГРАММА *ДОПВ*.
;ОЦЕНКА: ДЛИНА- 40 (+8 *ДОПВ*) БАЙТ, ВРЕМЯ- НЕ БОЛЕЕ
;1832 ТАКТОВ.
;*****
;ПРОВЕРКА ЧАСТНОГО НА ПЕРЕПОЛНЕНИЕ: ЧАСТНОЕ > 1 ?
;(ДЕЛИТЕЛЬ ДОЛЖЕН БЫТЬ > ДЕЛИМОГО)

```

```

0B20 7D      MOV     A, L
0B21 91      SUB     C
0B22 7C      MOV     A, H
0B23 98      SUB     B
0B24 D0      RNC          ; ЕСЛИ ПЕРЕПОЛНЕНИЕ, CY=0
; ДОПОЛНЕНИЕ ДЕЛИТЕЛЯ
0B25 CD5000  CALL    ДОПВ ; (В, С) - ДОПОЛНИТЕЛЬНЫЙ КОД

```

0B2B 3E10	D216A: MVI	A,16	;СЧЕТЧИК ЦИКЛОВ
			;СДВИГ ЧАСТНОГО (D,E) И ОСТАТКА (H,L) ВЛЕВО
0B2A EB	ЦИКЛ: XCHG		
0B2B 29	DAD	H	;СДВИГ ЧАСТНОГО
0B2C EB	XCHG		
0B2D 29	DAD	H	;СДВИГ ОСТАТКА
0B2E DA3C0B	JC	ПЕР1	;ЕСЛИ ПЕРЕПОЛНЕНИЕ ОСТАТКА
			;СЛОЖЕНИЕ ОСТАТКА С ДОПОЛНИТЕЛЬНЫМ КОДОМ ДЕЛИТЕЛЯ
0B31 E5	PUSH	H	;СОХРАНЕНИЕ ОСТАТКА
0B32 09	DAD	B	
0B33 D2410B	JNC	ПЕР2	;ЕСЛИ СУММА < 0
			;СУММА > 0.РАЗРЯД ЧАСТНОГО=1
0B36 33	INX	SP	
0B37 33	INX	SP	;БАЛАНС СТЕКА
0B3B 13	INX	D	;+1 В ЧАСТНОЕ
0B39 C3420B	JMP	ПЕР3	
			;ПЕРЕПОЛНЕНИЕ ОСТАТКА.РАЗРЯД ЧАСТНОГО=1
0B3C 09	ПЕР1: DAD	B	;ФОРМИРОВАНИЕ ОСТАТКА
0B3D 13	INX	D	;+1 В ЧАСТНОЕ
0B3E C3420B	JMP	ПЕР3	
			;СУММА < 0.ВОССТАНОВЛЕНИЕ ОСТАТКА
0B41 E1	ПЕР2: POP	H	
			;ПРОВЕРКА КОНЦА ЦИКЛА
0B42 3D	ПЕР3: DCR	A	
0B43 C22A0B	JNZ	ЦИКЛ	;ЗАЦИКЛИВАНИЕ
0B46 37	STC		;CY=1
0B47 C9	RET		
0000	END		

Результат операции содержит точное неполное дробное частное и дробный остаток. Эта программа может использоваться при делении точных целых чисел с получением приближенного частного, содержащего целую часть, которая вычисляется, например, с помощью программ типа Д32, и дробную часть, определяемую с помощью программы Д216.

Программа Д216 по структуре подобна программам Д24 и Д32А, т. е. использует дополнительный код делителя для вычисления очередного остатка.

Действия программы Д216 можно также интерпретировать как деление целого делимого с множителем  $2^{16}$  (этот множитель смещает запятую в делимом и частном (остатке) на 16 разрядов вправо) и получение целочисленных частного и остатка. Такая интерпретация упрощает представление тестовых данных программы (см. табл. 1.21).

**Табл. 1.21. Тесты деления формата 16:16=(16, 16)  
(целые беззнаковые двоичные числа)**

Представление чисел	
шестнадцатеричное	десятичное
FFFE·2 <sup>16</sup> :FFFF=(FFFE, FFFE)	65534·2 <sup>16</sup> :65535=(65534, 65534)
F0FF·2 <sup>16</sup> :FFFF=(F0FF, F0FF)	61695·2 <sup>16</sup> :65535=(61695, 61695)
5555·2 <sup>16</sup> :AAAA=(8000, 0000)	21845·2 <sup>16</sup> :43690=(32768, 0)
7FFF·2 <sup>16</sup> :8000=(FFFE, 0000)	32767·2 <sup>16</sup> :32768=(65534, 0)
0009·2 <sup>16</sup> :000A=(E666, 0004)	9·2 <sup>16</sup> :10=(58982, 4)

### 1.5.3. ЦЕЛЫЕ ЧИСЛА СО ЗНАКОМ

*Деление целых двоичных чисел* в дополнительных кодах целесообразно выполнять путем их преобразования в прямые коды, деления беззнаковых чисел и коррекции результата с учетом знаков делимого и делителя. Знаки частного и остатка в зависимости от знаков делимого и делителя будут определяться следующими правилами: (+) : (+) = (+, +); (+) : (−) = (−, +); (−) : (+) = (−, −) и (−) : (−) = (+, −). Процедура деления целых чисел со знаком рассматривается ниже на примере программы ДД216.

Программа ДД216 выполняет деление целых двоичных чисел в дополнительном коде с получением дробных частного и остатка:

```

0B50                                ORG 0B50H
0B20                                SET    0B20H
0050                                ДОПВ  SET    50H
0058                                ДОПД  SET    58H
0060                                ДОПН  SET    60H
0070                                ПОСД  SET    70H
0078                                ПОСН  SET    78H

```

ДД216:

```

; *****
; ПОДПРОГРАММА ДЕЛЕНИЯ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ В ДОПОЛНИ-
; ТЕЛЬНОМ КОДЕ ФОРМАТА 16:16=(16,16) С ДРОБНЫМ ЧАСТНЫМ.
; ВХОДНЫЕ ПАРАМЕТРЫ: (Н, L) - ДЕЛИМОЕ, (В, С) - ДЕЛИТЕЛЬ. ВЫ-
; ХОДНЫЕ ПАРАМЕТРЫ: (D, E) - ЧАСТНОЕ, (Н, L) - ОСТАТОК, CY=0 -
; ПРИЗНАК ПЕРЕПОЛНЕНИЯ ЧАСТНОГО. ИСПОЛЬЗУЮТСЯ ВСЕ РЕ-
; ГИСТРЫ, ГЛУБИНА СТЕКА В. ИСПОЛЬЗУЮТСЯ ПОДПРОГРАММЫ
; *Д216*, *ДОПВ*, *ДОПД*, *ДОПН*, *ПОСД*, *ПОСН*.
; ОЦЕНКА: ДЛИНА - 59 (+80 БАЙТ ПОДПРОГРАММ) БАЙТА,
; ВРЕМЯ - НЕ БОЛЕЕ 2300 ТАКОВ (С УЧЕТОМ ПОДПРОГРАММ).
; *****
; ПРОВЕРКА ЗНАКА ДЕЛИМОГО

```

```

0B50 7C                                MOV    A, H

```



0B51 17	RAL		
0B52 F5	PUSH	PSW	;СОХРАНЕНИЕ ЗНАКА ДЕЛИМОГО
0B53 D2590B	JNC	ПЕР1	;ЕСЛИ ЗНАК "+"
	;ДОПОЛНЕНИЕ ДЕЛИМОГО		
0B56 CD6000	CALL	ДОПН	; (H,L) - ДОПОЛНИТЕЛЬНЫЙ КОД
	;ПРОВЕРКА ЗНАКА ДЕЛИТЕЛЯ		
0B59 78	ПЕР1: MOV	A,B	
0B5A 17	RAL		
0B5B F5	PUSH	PSW	;СОХРАНЕНИЕ ЗНАКА ДЕЛИТЕЛЯ
0B5C D2620B	JNC	ПЕР2	;ЕСЛИ ЗНАК "+"
	;ДОПОЛНЕНИЕ ДЕЛИТЕЛЯ		
0B5F CD5000	CALL	ДОПВ	; (B,C) - ДОПОЛНИТЕЛЬНЫЙ КОД
	;ДЕЛЕНИЕ БЕЗЗНАКОВЫХ ЧИСЕЛ		
0B62 CD200B	ПЕР2: CALL	D216	; (D,E) - ЧАСТНОЕ, (H,L) - ОСТАТОК
0B65 D2870B	JNC	ПЕР	;ЕСЛИ ПЕРЕПОЛНЕНИЕ ЧАСТНОГО
	;САВИГ ВПРАВО ЧАСТНОГО И ОСТАТКА		
0B6B CB7000	CALL	ПОСА	
0B6B CB7800	CALL	ПОСН	
	;АНАЛИЗ ЗНАКОВ ДЕЛИМОГО И ДЕЛИТЕЛЯ		
0B6E F1	POP	PSW	;ВОССТАНОВЛЕНИЕ ЗНАКА ДЕЛИТЕЛЯ
0B6F DA7E0B	JC	ПЕР3	;ЕСЛИ ЗНАК "--"
0B72 F1	POP	PSW	;ВОССТАНОВЛЕНИЕ ЗНАКА ДЕЛИМОГО
0B73 D27C0B	JNC	ПЕР5	;ЕСЛИ ЗНАКИ (+, +)
	;ДЕЛИМОЕ < 0, ДЕЛИТЕЛЬ > 0		
0B76 CD5800	CALL	ДОПА	;ДОПОЛНЕНИЕ ЧАСТНОГО
0B79 CD6000	ПЕР4: CALL	ДОПН	;ДОПОЛНЕНИЕ ОСТАТКА
0B7C 37	ПЕР5: STC		;CY=1
0B7D C9	RET		
	;ДЕЛИТЕЛЬ < 0		
0B7E F1	ПЕР3: POP	PSW	;ВОССТАНОВЛЕНИЕ ЗНАКА ДЕЛИМОГО
0B7F DA790B	JC	ПЕР4	;ЕСЛИ ЗНАКИ (-, -)
0B82 CD5800	CALL	ДОПА	;ДОПОЛНЕНИЕ ЧАСТНОГО
0B85 37	STC		;CY=1
0B86 C9	RET		
	;ПЕРЕПОЛНЕНИЕ ЧАСТНОГО		
0B87 F1	ПЕР: POP	PSW	
0B88 F1	POP	PSW	;БАЛАНС СТЕКА
0B89 AF	XRA	A	;CY=0
0B8A C9	RET		
0000	END		

Программа обращается к подпрограмме беззнакового деления D216, известным вспомогательным подпрограммам получения дополнительного кода ДОПВ, ДОПД, ДОПН и новым вспомогательным подпрограммам правого одноразрядного сдвига числа соответственно в регистровых парах (B, C), (D, E) и (H, L):

006B

ORG 6BH

ПОСВ:

\*\*\*\*\*  
;ПОДПРОГРАММА ПРАВОГО ОДНОРАЗЯДНОГО СДВИГА ЧИСЛА В РЕ-  
;ГИСТРОВОЙ ПАРЕ (B,C). ВДВИГАЕТСЯ БИТ CY=0, ЕСЛИ ВХОД В  
;ПРОГРАММУ "ПОСВ", И БИТ CY=0 ИЛИ 1, ЕСЛИ ВХОД "ПОСВ+1".  
;ВХОДНОЙ ПАРАМЕТР: (B,C) - ИСХОДНОЕ ЧИСЛО. ВЫХОДНОЙ ПАРА-

;МЕТР: (В,С)–ЧИСЛО ПОСЛЕ СДВИГА.ИСПОЛЬЗУЕТСЯ РЕГИСТР А.  
;ОЦЕНКА:ДЛИНА–8 БАЙТ,ВРЕМЯ–42 ТАКТА.

\*\*\*\*\*

```
0068 AF      XRA      A      ;CY=0
0069 78      MOV      A,B
006A 1F      RAR
006B 47      MOV      B,A
006C 79      MOV      A,C
006D 1F      RAR
006E 4F      MOV      C,A
006F C9      RET
```

ПОСД:

\*\*\*\*\*

;ПОДПРОГРАММА ПРАВОГО ОДНОРАЗЯДНОГО СДВИГА ЧИСЛА В РЕ-  
;ГИСТРОВОЙ ПАРЕ (D,E).ВДВИГАЕТСЯ БИТ CY=0,ЕСЛИ ВХОД В  
;ПРОГРАММУ "ПОСД",И БИТ CY=0 ИЛИ 1,ЕСЛИ ВХОД "ПОСД+1".  
;ВХОДНОЙ ПАРАМЕТР: (D,E)–ИСХОДНОЕ ЧИСЛО.ВЫХОДНОЙ ПАРА-  
;МЕТР: (D,E)–ЧИСЛО ПОСЛЕ СДВИГА.ИСПОЛЬЗУЕТСЯ РЕГИСТР А.  
;ОЦЕНКА:ДЛИНА–8 БАЙТ,ВРЕМЯ–42 ТАКТА.

\*\*\*\*\*

```
0070 AF      XRA      A      ;CY=0
0071 7A      MOV      A,D
0072 1F      RAR
0073 57      MOV      D,A
0074 7B      MOV      A,E
0075 1F      RAR
0076 5F      MOV      E,A
0077 C9      RET
```

ПОСН:

\*\*\*\*\*

;ПОДПРОГРАММА ПРАВОГО ОДНОРАЗЯДНОГО СДВИГА ЧИСЛА В РЕ-  
;ГИСТРОВОЙ ПАРЕ (H,L).ВДВИГАЕТСЯ БИТ CY=0,ЕСЛИ ВХОД В  
;ПОДПРОГРАММУ "ПОСН",И БИТ CY=0 ИЛИ 1,ЕСЛИ ВХОД "ПОСН+1"  
;ВХОДНОЙ ПАРАМЕТР: (H,L)–ИСХОДНОЕ ЧИСЛО.ВЫХОДНОЙ ПАРА-  
;МЕТР: (H,L)–ЧИСЛО ПОСЛЕ СДВИГА.ИСПОЛЬЗУЕТСЯ РЕГИСТР А.  
;ОЦЕНКА:ДЛИНА–8 БАЙТ,ВРЕМЯ–42 ТАКТА.

\*\*\*\*\*

```
0078 AF      XRA      A      ;CY=0
0079 7C      MOV      A,H
007A 1F      RAR
007B 67      MOV      H,A
007C 7D      MOV      A,L
007D 1F      RAR
007E 6F      MOV      L,A
007F C9      RET
0000      END
```

Эти подпрограммы неоднократно применяются в дальнейшем при разработке новых программ. В контексте данной программы подпрограммы ПОСД и ПОСН используются для сдвига вправо соответственно частного и остатка с целью внесения в формат их знаков, поскольку деление целых беззнаковых чисел дает беззнаковые дробные частное и остаток. После сдвига прямых кодов этих чисел вправо и освобождения (обнуления) крайнего ле-

вого разряда знак минус вводится, если необходимо, дополнением частного или остатка. Тестовые данные программы приведены в табл. 1.22.

**Табл. 1.22. Тесты деления формата 16:16=(16, 16)  
(целые двоичные числа в дополнительном коде)**

Представление чисел	
шестнадцатеричное	десятичное
FF00:FE00=(4000, 0000)	(-256):(-512)=(+0,5; 0)
F000:2000=(C000, 0000)	(-4096):( +8192)=(-0,5; 0)
5555:AAAA=(8002, 0001)	(+21845):(-21846)=(-0,99999; +0,00001)
7FFF:8000=(8001, 0000)	(+32767):(-32768) = (-0,99999; 0)
4000:5000=(6666, 2000)	(+16384):( +20480)=(+0,75; +0,25)

### 1.5.4. ДРОБНЫЕ ЧИСЛА СО ЗНАКОМ

#### 1.5.4.1. Деление дробных чисел

*Деление дробных чисел*  $Z:Y \approx X$  в дополнительных кодах можно выполнять теми же методами, что и деление целых чисел со знаком, т. е. преобразованием в прямые коды, делением беззнаковых чисел и коррекцией результата с учетом знаков делимых чисел. Далее будет приведена программа ДДФ17, реализующая такой способ деления. Вместе с тем деление дробных чисел в дополнительных кодах можно выполнять и непосредственно, без предварительного преобразования их в прямые коды, а на основе модифицированного алгоритма деления без восстановления остатка [61]:

$$Q_i = \begin{cases} 2Q_{i-1} - Y, & \text{если } S(Q_{i-1}) = S(Y); \\ 2Q_{i-1} + Y, & \text{если } S(Q_{i-1}) \neq S(Y); \end{cases} \quad (1.26)$$

$$x_{n-i} = \begin{cases} 1, & \text{если } S(Q_i) = S(Y); \\ 0, & \text{если } S(Q_i) \neq S(Y), \end{cases} \quad (1.27)$$

где  $S(Q_i)$ ,  $S(Y)$  — знаки соответственно остатка  $Q_i$  и делителя  $Y$ ;  $Q_0 = Z$  и  $S(Q_0) = S(Z)$ . Иными словами, если делимое  $Z$  и делитель  $Y$  представлены в дополнительных кодах, то на очередном  $i$ -м цикле деления делитель вычитается из удвоенного остатка, если знаки делителя и остатка (делимого) совпадают, и делитель складывается

с удвоенным остатком, если их знаки разные. В результате выполнения алгоритма за  $n$  циклов деления автоматически образуются цифровые и знаковый разряды частного.

#### 1.5.4.2. Формат 16:16=16

Программа ДДФ16 реализует алгоритм деления (1.26), (1.27):

```

0B90                                ORG      0B90H
                                ДДФ16:
                                ;*****
                                ;ПОДПРОГРАММА ДЕЛЕНИЯ ДВОИЧНЫХ ЧИСЕЛ В ДОПОЛНИТЕЛЬНОМ
                                ;КОДЕ С ФИКСИРОВАННОЙ ПОСЛЕ ЗНАКОВОГО РАЗРЯДА ЗАПЯТОЙ
                                ;ФОРМАТА 16:16=16 (ДЕЛИТЕЛЬ ПО АБСОЛЮТНОЙ ВЕЛИЧИНЕ
                                ;ДОЛЖЕН БЫТЬ БОЛЬШЕ ДЕЛИМОГО).
                                ;ВХОДНЫЕ ПАРАМЕТРЫ: (H,L) - ДЕЛИМОЕ, (B,C) - ДЕЛИТЕЛЬ. ВЫХОД-
                                ;НОЙ ПАРАМЕТР: (D,E) - ЧАСТНОЕ. ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ,
                                ;СОХРАНЯЕТСЯ (B,C). ГЛУБИНА СТЕКА - 2.
                                ;ОЦЕНКА: ДЛИНА - 40 БАЙТ, ВРЕМЯ - НЕ БОЛЕЕ 2258 ТАКОВ.
                                ;*****
0B90 3E1C                          MVI      A,16      ;СЧЕТЧИК ЦИКЛОВ
                                ;АНАЛИЗ ЗНАКОВ ОСТАТКА И ДЕЛИТЕЛЯ НА СОВПАДЕНИЕ
0B92 F5                          ЦИКЛ:  PUSH   PSW      ;СОХРАНЕНИЕ СЧЕТЧИКА
0B93 7C                          MOV     A,H
0B94 A8                          XRA     B          ;S=0, ЕСЛИ ЗНАКИ ОДИНАКОВЫ
0B95 37                          STC     ;CY=1 (ЗНАКИ ОДИНАКОВЫ)
0B96 F29A0B                     JP      PER1      ;ЕСЛИ ЗНАКИ ОДИНАКОВЫ
0B99 3F                          CMC     ;CY=0 (ЗНАКИ РАЗНЫЕ)
                                ;СДВИГ ВЛЕВО ОСТАТКА В (H,L) И ЧАСТНОГО В (D,E)
                                ;УСТАНОВКА ПО CY=1 РАЗРЯДА ЧАСТНОГО
0B9A 7B                          PER1:  MOV     A,E
0B9B 17                          RAL
0B9C 5F                          MOV     E,A
0B9D 7A                          MOV     A,D
0B9E 17                          RAL
0B9F 57                          MOV     D,A
0BA0 29                          DAD     H
0BA1 F2A0B                      JP      PER2      ;ЕСЛИ ЗНАКИ ОДИНАКОВЫ
                                ;СЛОЖЕНИЕ ОСТАТКА С ДЕЛИТЕЛЕМ ПРИ РАЗНЫХ ЗНАКАХ
0BA4 09                          DAD     B
0BA5 C3AE0B                     JMP     PER3
                                ;ВЫЧИТАНИЕ ДЕЛИТЕЛЯ ИЗ ОСТАТКА ПРИ ОДИНАКОВЫХ ЗНАКАХ
0BA8 7D                          PER2:  MOV     A,L
0BA9 91                          SUB     C
0BAA 6F                          MOV     L,A
0BAB 7C                          MOV     A,H
0BAC 98                          SBB     B
0BAD 67                          MOV     H,A
                                ;ПРОВЕРКА КОНЦА ЦИКЛА
0BAE F1                          PER3:  POP     PSW
0BAF 3D                          DCR     A
0BB0 C2920B                     JNZ     ЦИКЛ      ;ЗАЦИКЛИВАНИЕ
                                ;КОРРЕКЦИЯ ЗНАКА ЧАСТНОГО
0BB3 7A                          MOV     A,D

```

0BB4 C6B0	ADI	80H
0BB6 57	MOV	D,A
0BB7 C9	RET	
0000	END	

Знаковый разряд частного определяется в программе на первом цикле деления и при совпадении знаков делимого и делителя устанавливается в единицу (хотя, очевидно, должен быть равен нулю), что позволяет достигнуть регулярности в теле цикла. Но по окончании деления знаковый разряд корректируется:  $1 \rightarrow 0$  или  $0 \rightarrow 1$ . Заметим, что для правильной работы программы делитель должен быть по абсолютной величине больше делимого, иначе возможно переполнение — появление неправильных дробей.

#### 1.5.4.3. Формат 17:17 = 17

Программа ДДФ17 реализует деление чисел, размещенных в памяти и имеющих знаковый разряд в старшем бите знакового байта:

0C20		ORG	0C20H
0050	ДОПВ	SET	50H
0060	ДОПН	SET	60H
0068	ПОСВ	SET	68H
0B28	Д216А	SET	0B28H

ДДФ17:

```

;*****
;ПОДПРОГРАММА ДЕЛЕНИЯ ДВОИЧНЫХ НОРМАЛИЗОВАННЫХ ЧИСЕЛ В
;ДОПОЛНИТЕЛЬНОМ КОДЕ С ФИКСИРОВАННОЙ ПОСЛЕ ЗНАКОВОГО
;РАЗРЯДА ЗАПЯТОЙ ФОРМАТА (1,16):(1,16)=(1,16),ГДЕ (1,16)=
;=(ЗНАК,СТБ,МЛБ).
;ВХОДНЫЕ ПАРАМЕТРЫ:(D,E)—АДРЕС ДЕЛИМОГО ДМ В ПАМЯТИ,
;(H,L)—АДРЕС ДЕЛИТЕЛЯ ДЛ В ПАМЯТИ.ВЫХОДНЫЕ ПАРАМЕТРЫ:
;(B,C)—ЧАСТНОЕ,CY=1—ПРИЗНАК ПЕРЕПОЛНЕНИЯ ЧАСТНОГО.ИС-
;ПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ,СОХРАНЯЮТСЯ (D,E),(H,L).ГЛУБИНА
;СТЕКА=8.ИСПОЛЬЗУЮТСЯ ПОДПРОГРАММЫ:*ДОПВ*,*ДОПН*,*Д216А*
;ОЦЕНКА:ДЛИНА=61 БАЙТ (+56 БАЙТ ПОДПРОГРАММ),ВРЕМЯ=НЕ
;БОЛЕЕ 2243 ТАКТОВ.
;*****

```

0C20 D5	PUSH	D	; СОХРАНЕНИЕ АДРЕСА ДМ
0C21 E5	PUSH	H	; СОХРАНЕНИЕ АДРЕСА ДЛ
	;ПРОВЕРКА ЗНАКОВ ДЕЛИМОГО И ДЕЛИТЕЛЯ НА СОВПАДЕНИЕ		
0C22 1A	LDA	D	
0C23 AE	XRA	M	; (A7=1),S=1,ЕСЛИ РАЗНЫЕ ЗНАКИ
0C24 F5	PUSH	PSW	; СОХРАНЕНИЕ ПРИЗНАКОВ
	;ПЕРЕСМЛКА ДЕЛИТЕЛЯ ИЗ ПАМЯТИ В РЕГИСТРЫ (B,C)		
0C25 7E	MOV	A,M	
0C26 B7	ORA	A	; ПРОЯВЛЕНИЕ ЗНАКА ДЕЛИТЕЛЯ
0C27 23	INX	H	
0C28 46	MOV	B,M	
0C29 23	INX	H	

0C2A 4E	MOV	C,M	; (B,C) - ДЕЛИТЕЛЬ
0C2B F45000	CP	ДОПВ	; ДОПОЛНЕНИЕ ДЛ, ЕСЛИ ДЛ > 0
	; ПЕРЕСЫЛКА ДЕЛИМОГО ИЗ ПАМЯТИ В РЕГИСТРЫ (H,L)		
0C2E EB	XCHG		; (H,L) - АДРЕС ДМ
0C2F 7E	MOV	A,M	
0C30 87	ORA	A	; ПРОЯВЛЕНИЕ ЗНАКА ДЕЛИМОГО
0C31 23	INX	H	
0C32 56	MOV	D,M	
0C33 23	INX	H	
0C34 5E	MOV	E,M	
0C35 EB	XCHG		; (H,L) - ДЕЛИМОЕ
0C36 FC6000	CM	ДОПН	; ДОПОЛНЕНИЕ ДМ, ЕСЛИ ДМ < 0
	; ПРОВЕРКА ПЕРЕПОЛНЕНИЯ ЧАСТНОГО: > 1?		
0C39 E5	PUSH	H	; СОХРАНЕНИЕ ДЕЛИМОГО
0C3A 09	DAD	B	; (H,L) - РАЗНОСТЬ ДМ И ДЛ
0C3B DA420C	JC	ПЕР1	; ЕСЛИ ЧАСТНОЕ > 1
0C3E E1	POP	H	; ВОССТАНОВЛЕНИЕ ДЕЛИМОГО
0C3F C3440C	JMP	ПЕР2	
0C42 33	ПЕР1: INX	SP	
0C43 33	INX	SP	; БАЛАНС СТЕКА
0C44 F5	ПЕР2: PUSH	PSW	; СОХРАНЕНИЕ ПРИЗНАКА ПЕРЕПОЛНЕНИЯ
	; БЕЗЗНАКОВОЕ ДЕЛЕНИЕ ДВОИЧНЫХ ЧИСЕЛ ФОРМАТА 16:16=16		
0C45 CD280B	CALL	D216A	; (D,E) - ЧАСТНОЕ
0C48 42	MOV	B,D	
0C49 4B	MOV	C,E	; (B,C) - ЧАСТНОЕ
	; КОРРЕКЦИЯ ЧАСТНОГО ПРИ ПЕРЕПОЛНЕНИИ		
0C4A F1	POP	PSW	; ВОССТАНОВЛЕНИЕ ПРИЗНАКА
0C4B 1600	MVI	D,0	; (D)=0 - ПРИЗНАК НЕПЕРЕПОЛНЕНИЯ
0C4D D2540C	JNC	ПЕР3	; ЕСЛИ НЕТ ПЕРЕПОЛНЕНИЯ
0C50 CD6900	CALL	ПОСВ+1	; СДВИГ ЧАСТНОГО ВПРАВО
0C53 14	INR	D	; (D)=1 - ПРИЗНАК ПЕРЕПОЛНЕНИЯ
0C54 F1	ПЕР3: POP	PSW	; ВОССТАНОВЛЕНИЕ ПРИЗНАКОВ
0C55 FC5000	CM	ДОПВ	; ДОПОЛНЕНИЕ ЧАСТНОГО
0C58 7A	MOV	A,D	; (A) - ПРИЗНАК ПЕРЕПОЛНЕНИЯ
0C59 1F	RAR		; ПРОЯВЛЕНИЕ ПРИЗНАКА CY
0C5A E1	POP	H	; ВОССТАНОВЛЕНИЕ АДРЕСА ДЛ
0C5B D1	POP	D	; ВОССТАНОВЛЕНИЕ АДРЕСА ДМ
0C5C C9	RET		; CY=1, ЕСЛИ ПЕРЕПОЛНЕНИЕ ЧАСТНОГО
0000	END		

Программа пересылает делимые числа из памяти в регистры, преобразовывает их в прямые коды, выполняет беззнаковое деление этих кодов посредством подпрограммы Д216А (дополнительный вход в подпрограмму Д216), анализирует переполнение частного (если делимое оказывается по модулю больше делителя), устраняет одно-разрядное переполнение формата частного и преобразует частное в дополнительный код, если знаки делимых чисел различны. Она используется в программе деления чисел с плавающей запятой (см. об этом в гл. 2). Тестовые данные

для программы приведены в табл. 1.23. Эти данные можно использовать и для проверки программы ДДФ16 (предварительно сдвинув шестнадцатеричные числа на один двоичный разряд вправо с учетом знака).

**Табл. 1.23. Тесты деления формата 17:17=17  
(дробные двоичные числа в дополнительном коде)**

Представление чисел	
шестнадцатеричное	двоичное
(+8000):(+C000)=+AAAA	(+0,5):(+0,75)=+0,6666
(+8000):(+AAAA)=+C000	(+0,5):(+0,6666)=+0,75
(+8000):(+CCCC)=+A000	(+0,5):(+0,8)=+0,625
(-8000):(-4000)=+AAAA	(-0,5):(-0,75)=+0,6666
(-8000):(+C000)=-5556	(-0,5):(+0,75)=-0,6666

## 1.6. ДЕЛЕНИЕ ДЕСЯТИЧНЫХ ЧИСЕЛ

Деление десятичных чисел значительно сложнее деления двоичных чисел, поскольку для получения текущей десятичной цифры частного необходимо последовательно вычитать делитель из делимого или остатка, вплоть до получения отрицательной разности. Цифра частного соответствует выполненному числу вычитаний, не считая последнего вычитания, дающего отрицательную разность. Вычисление очередной цифры частного выполняется аналогично двоичному делению сдвигом остатка влево (но на четыре двоичных разряда, т. е. на один десятичный разряд) и очередной последовательностью вычитаний делителя (или суммированием остатка с дополнительным десятичным кодом делителя). Этот простейший алгоритм деления десятичных чисел иллюстрируется ниже двумя программами деления.

Программа Д3210 выполняет деление трехразрядного десятичного числа на двухразрядное с получением одноразрядного частного и двухразрядного остатка:

OC70

ORG

OC70H

Д3210:

```

;*****
;ПОДПРОГРАММА ДЕЛЕНИЯ БЕЗЗНАКОВЫХ ЦЕЛЫХ ДВОИЧНО-ДЕСЯ-
;ТИЧНЫХ ЧИСЕЛ ФОРМАТА (3*4):(2*4)=(1*4,2*4).
;ВХОДНЫЕ ПАРАМЕТРЫ:(E,N)-ДЕЛИМОЕ (3 ЦИФРЫ),(C)-ДОПОЛНИ-
;ТЕЛЬНЫЙ ДВОИЧНО-ДЕСЯТИЧНЫЙ КОД ДЕЛИТЕЛЯ (2 ЦИФРЫ).ДЕ-
;ЛИТЕЛЬ БОЛЬШЕ ДВУХ СТАРШИХ ЦИФР ДЕЛИМОГО.(L)-ЧАСТНОЕ:
; (L)=0.ВЫХОДНЫЕ ПАРАМЕТРЫ:(N)-ДВОИЧНО-ДЕСЯТИЧНЫЙ ОСТА-

```

```

;ТОК (2 ЦИФРЫ), (L) - ЧАСТНОЕ (1 ЦИФРА). ИСПОЛЬЗУЮТСЯ ВСЕ
;РЕГИСТРЫ.
;ОЦЕНКА: ДЛИНА - 16 БАЙТ, ВРЕМЯ - НЕ БОЛЕЕ 600 ТАКТОВ.
;*****
;СЛОЖЕНИЕ ДЕЛИМОГО (СТЦ, СРЦ, МЛЦ) С ДОПОЛНИТЕЛЬНЫМ КОДОМ
;ДЕЛИТЕЛЯ (СТЦ, СРЦ, МЛЦ)
0C70 7C      ЦИКЛ:  MOV     A, H
0C71 81      ADD     C
0C72 27      DAA
0C73 57      MOV     D, A      ;ВРЕМЕННОЕ ХРАНЕНИЕ
0C74 3E99    MVI     A, 99H    ;ДОПОЛНЕНИЕ СТЦ ДЕЛИТЕЛЯ
0C76 8B      ADC     E
0C77 27      DAA
0C78 47      MOV     B, A      ;ВРЕМЕННОЕ ХРАНЕНИЕ
0C79 D0      RNC              ;ЕСЛИ РАЗНОСТЬ < 0
;УВЕЛИЧЕНИЕ ЦИФРЫ ЧАСТНОГО, ЗАПОМИНАНИЕ ОСТАТКА
0C7A 2C      INR     L
0C7B 62      MOV     H, D
0C7C 58      MOV     E, B
0C7D C3700C  JMP     ЦИКЛ      ;ЗАЦИКЛИВАНИЕ
0000      END

```

Эта программа используется в качестве подпрограммы в программе Д4210 деления четырехразрядного десятичного числа на двухразрядное:

```

0C80      ORG     0C80H
0CA0      LC4H   SET     0CA0H
0C70      D3210  SET     0C70H

Д4210:
;*****
;ПОДПРОГРАММА ДЕЛЕНИЯ БЕЗЗНАКОВЫХ ЦЕЛЫХ ДВОИЧНО-ДЕСЯ-
;ТИЧНЫХ ЧИСЕЛ ФОРМАТА (4*4) : (2*4) = (2*4, 2*4) .
;ВХОДНЫЕ ПАРАМЕТРЫ: (H, L) - ДЕЛИМОЕ (4 ЦИФРЫ), (C) - ДЕЛИ-
;ТЕЛЬ (2 ЦИФРЫ). ВЫХОДНЫЕ ПАРАМЕТРЫ: (H) - ОСТАТОК (2 ЦИФ-
;РЫ), (L) - ЧАСТНОЕ (2 ЦИФРЫ), CY=0 - ПРИЗНАК ПЕРЕПОЛНЕНИЯ
;ЧАСТНОГО. ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ.
;ОЦЕНКА: ДЛИНА - 21 БАЙТ (+27 БАЙТ ПОДПРОГРАММ), ВРЕМЯ -
;НЕ БОЛЕЕ 1616 ТАКТОВ (С УЧЕТОМ ПОДПРОГРАММ)
;*****
;ПРОВЕРКА ЧАСТНОГО НА ПЕРЕПОЛНЕНИЕ: ЧАСТНОЕ > (2*4) ?
0C80 7C      MOV     A, H
0C81 91      SUB     C
0C82 D0      RNC              ;ЕСЛИ ПЕРЕПОЛНЕНИЕ, CY=0
;ДВОИЧНО-ДЕСЯТИЧНОЕ ДОПОЛНЕНИЕ ДЕЛИТЕЛЯ
0C83 3E9A    MVI     A, 9AH
0C85 91      SUB     C
0C86 4F      MOV     C, A      ;(C) - ДОПОЛНИТЕЛЬНЫЙ КОД
;СДВИГ ДЕЛИМОГО ВЛЕВО НА ДЕСЯТИЧНЫЙ РАЗРЯД
0C87 CDA00C  CALL    LC4H      ;(E, H, LH) - ДЕЛИМОЕ ПОСЛЕ СДВИГА
;ВЫЧИСЛЕНИЕ СТАРШЕЙ СТЦ ЦИФРЫ ЧАСТНОГО
0C8A CD700C  CALL    D3210     ;(H) - ОСТАТОК, (LL) - СТЦ ЧАСТНОГО
;СДВИГ ОСТАТКА И ЧАСТНОГО ВЛЕВО НА ДЕСЯТИЧНЫЙ РАЗРЯД
0C8D CDA00C  CALL    LC4H      ;(E, H) - ОСТАТОК, (LH) - СТЦ ЧАСТНОГО
;ВЫЧИСЛЕНИЕ МЛАДШЕЙ МЛЦ ЦИФРЫ ЧАСТНОГО
0C90 CD700C  CALL    D3210     ;(H) - ОСТАТОК, (L) - СТЦ, МЛЦ

```



0C93 37	STC	;CY=1
0C94 C9	RET	
0000	END	

Программа Д4210 обращается к вспомогательной программе ЛС4Н сдвига влево на четыре разряда содержимого регистровой пары (Н, L):

0CA0		ORG	0CA0H
------	--	-----	-------

ЛС4Н:

```

;*****
;ПОДПРОГРАММА СДВИГА ВЛЕВО НА 4 РАЗРЯДА СОДЕРЖИМОГО
;Н-ПАРЫ РЕГИСТРОВ С СОХРАНЕНИЕМ ВЫДВИНУТЫХ РАЗРЯДОВ В
;(Е) И ЗАПОЛНЕНИЕМ СВОБОДНЫХ ПРАВЫХ РАЗРЯДОВ НУЛЯМИ.
;ВХОДНОЙ ПАРАМЕТР: (Н, L) - ИСХОДНОЕ ЧИСЛО. ВЫХОДНОЙ ПАРА-
;МЕТР: (Е, Н, L) - СДВИНУТОЕ ЧИСЛО. ИСПОЛЗУЮТСЯ РЕГИСТРЫ В, А.
;ОЦЕНКА: ДЛИНА-11 БАЙТ, ВРЕМЯ-142 ТАКТА.
;*****
0CA0 0604      MVI    В, 4      ;СЧЕТЧИК СДВИГОВ
0CA2 AF        XRA     А        ; (А)=0
;СДВИГИ ЧИСЛА ВЛЕВО С УЧЕТОМ ПЕРЕНОСА
ЦИКЛ: 0CA3 29   DAD     Н
0CA4 17        RAL
0CA5 05        DCR     В
0CA6 C2A30C    JNZ     ЦИКЛ    ;ЗАЦИКЛИВАНИЕ
0CA9 5F        MOV     Е, А
0CAA C9        RET
0000          END

```

Аналогично можно строить программы деления и мно-  
горазрядных десятичных чисел, но они по сравнению с  
программами деления двоичных чисел значительно слож-  
нее и выполняются медленнее.

## 2. ПРОГРАММЫ АРИФМЕТИКИ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ

### 2.1. ОБЩИЕ СВЕДЕНИЯ

Обработка числовых данных в форме с плавающей запятой считалась до недавнего времени нетипичной для микропроцессорных систем [5]. Однако расширение областей использования микропроцессоров, в частности создание микропроцессорных информационно-измерительных систем и распределенных сетей сбора данных, потребовало существенного увеличения диапазона обрабатываемых чисел при сохранении их ограниченной значности, определяемой точностью датчиков первичной информации [23, 24]. В этих условиях применение арифметики с фиксированной запятой приводит к резкому увеличению разрядности чисел, росту затрат памяти и времени выполнения программ, что снижает в целом производительность и экономичность микропроцессорных систем. Кроме того, с увеличением разрядности чисел с фиксированной запятой и усложнением алгоритмов обработки возрастает неопределенность прогнозирования вычислений и затрудняется их предварительное масштабирование, призванное гарантировать корректность и требуемую точность результатов. Эти проблемы снимает *арифметика с плавающей запятой*, поскольку она обеспечивает раздельное представление диапазона и точности чисел (мантиссы и порядка) и реализует их автоматическое масштабирование в процессе вычислений [13, 31, 33, 34, 47, 50, 61, 66].

В данной главе рассматриваются особенности представления чисел в форме с плавающей запятой, а также алгоритмы и программы сложения, умножения и деления двоичных чисел в дополнительных кодах с плавающей запятой форматов обычной и повышенной точности.

*Числа с плавающей запятой* имеют следующую общую форму представления [с учетом выражений (1.1) и (1.6)]:

$$A_R = R^m A_{R\Phi}, \quad m = \text{var}, \quad (2.1)$$

где  $R^m$  — характеристика  $R$ -ичного числа  $A_R$ ;  $m$  — *целочисленный порядок*:  $m \in \{0, \pm 1, \pm 2, \dots, \pm m_{\max}\}$ , т. е.  $|m| \leq m_{\max}$ ;  $A_{R\Phi}$  — *мантисса* числа  $A_R$  — правильная дробь, определяемая выражениями:

$$\left. \begin{aligned} A_{R\Phi} &= \pm \sum_{i=1}^n a_i R^{-i} = \pm, a_1 a_2 \dots a_n; \\ R^{-n} &\leq |A_{R\Phi}| \leq 1 - R^{-n} \text{ или } A_{R\Phi} = 0, \end{aligned} \right\} \quad (2.2)$$

где  $a_i \in \{0, 1, \dots, R-1\}$  —  $R$ -ичные цифры. В мантиссе запятая фиксирована перед ее старшим цифровым разрядом, но фактическое положение запятой в представлении числа определяется независимо от мантиссы порядком  $m$  и изменяется — «плавает» — в зависимости от его величины и знака (см. § 1.1).

Существует много вариантов представления конкретного числа в форме с плавающей запятой. Например, смешанное число 12,34 можно представить в следующем виде:  $10^2 \cdot 0,1234$ ;  $10^3 \cdot 0,01234$ ;  $10^4 \cdot 0,001234$  и т. п. На практике один из вариантов представления выбирают в качестве стандартного. Если мантисса удовлетворяет условию

$$R^{-1} \leq |A_{R\Phi}| < 1, \quad (2.3)$$

такую мантиссу и число с плавающей запятой называют *нормализованными*. Признаком нормализованности абсолютной величины числа является наличие в старшем цифровом разряде мантиссы ненулевой цифры, т. е.  $a_1 \neq 0$ . Все остальные числа, не удовлетворяющие условию (2.3), называют *ненормализованными*. Так, в рассмотренном выше примере первый вариант представления числа с плавающей запятой определяет нормализованное, а другие — ненормализованные числа. Преимущества нормализованных чисел заключаются в том, что они, во-первых, определяются единственным образом и, во-вторых, обеспечивают максимально возможную точность представления чисел в выбранном  $n$ -разрядном формате, поскольку не тратят его разряды на изображение незначащих нулей. В нормализованном виде можно представить, вообще говоря, любое число, отличное от нуля. Для представления нуля в форме с плавающей запятой на практике используют несколько вариантов: 1)  $A_{R\Phi} = 0$ ,  $m$  — произвольно; 2)  $A_{R\Phi}$  — произвольно,  $m = m_{\min}$ ;

3)  $A_{R\phi} = 0$ ,  $m = 0$ ; 4)  $A_{R\phi} = 0$ ,  $m = m_{\min}$ , где  $m_{\min} = -m_{\max}$ . Чаще всего применяются варианты третий и четвертый [33, 47, 61].

Машинное представление формата чисел с плавающей запятой в отличие от формата чисел с фиксированной запятой содержит, помимо разряда знака  $S_m$  и  $n$  разрядов цифровой части числа — мантииссы, дополнительно разряд знака  $S_n$  и  $k$  цифровых разрядов порядка (рис. 2.1). В соответствии с этим представлением и учетом формулы (1.7) порядок  $m$  числа  $A_R$  определяется выражениями:

$$m = \pm \sum_{i=0}^{k-1} b_i R^i; \quad 0 \leq |m| \leq R^k - 1 = m_{\max}, \quad (2.4)$$

где  $b_i \in \{0, 1, \dots, R-1\}$  —  $R$ -ичные цифры порядка.

Оценим с учетом выражений (2.1) — (2.4) диапазон представления чисел с плавающей запятой:

$$\begin{aligned} R^{-m_{\max}} \cdot R^{-1} &= |A_R|_{\min} \leq |A_R| \leq |A_R|_{\max} = \\ &= R^{m_{\max}} \cdot (1 - R^{-n}), \end{aligned} \quad (2.5)$$

где значение  $|A_R|_{\min}$  справедливо для нормализованных чисел (в случае ненормализованных чисел  $|A_R|_{\min} = R^{-m_{\max} - n}$ ).

Из неравенств (2.3) — (2.5) следует, что *диапазон нормализованных чисел* практически не зависит от *разрядности мантииссы*, а определяется в основном величиной основания системы счисления и *разрядностью порядка*. Размещение диапазона чисел с плавающей запятой на числовой оси аналогично размещению чисел с фиксированной запятой (см. рис. 1.2, а). При появлении в процессе вычислений чисел, не принадлежащих диапазону, возникают, как и для чисел с фиксированной запятой, ошибки переполнения  $|A_R| > |A_R|_{\max}$  и антипереполнения (машинного нуля)  $|A_R| < |A_R|_{\min}$ , которые для чисел с плавающей запятой выражаются как одноименные ошибки порядка:  $m > m_{\max}$  и  $m < m_{\min}$ .

Точность представления чисел с плавающей запятой определяется теми же общими формулами (1.12), что и для чисел с фиксированной запятой. Граничная абсолютная ошибка  $\Delta_r$  этих чисел в отличие от чисел с фиксированной запятой непостоянна и зависит не только от разряд-

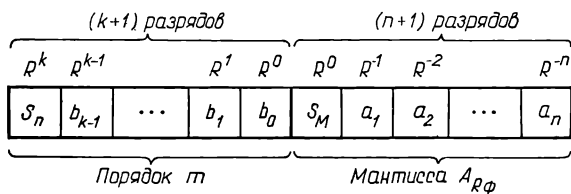


Рис. 2.1. Формат  $R$ -ичного числа с плавающей запятой

ности мантиссы и метода ее округления, но и от величины порядка:  $\Delta_r = R^m \cdot R^{-n}$  — для несимметричного и  $\Delta_r = R^m \cdot 0,5 \cdot R^{-n}$  — для симметричного округления (1.9). Значение этой ошибки принадлежит диапазону (для несимметричного округления)

$$R^{-m_{\max} - n} \leq \Delta_r \leq R^{m_{\max} - n}.$$

Величина граничной относительной ошибки с учетом формулы (1.13) определяется выражением

$$\frac{R^m \cdot R^{-n}}{R^m \cdot (1 - R^{-n})} \approx R^{-n} \leq \delta_r \leq \frac{R^m \cdot R^{-n}}{R^m \cdot R^{-1}} = R^{-n+1}, \quad (2.6)$$

где  $R^m \cdot R^{-n}$  — граничная абсолютная ошибка числа с плавающей запятой (при данном порядке  $m$ );  $R^m \cdot R^{-1}$ ,  $R^m \cdot (1 - R^{-n})$  — соответственно минимальное и максимальное абсолютные значения числа при нормализованной мантиссе. Из неравенства (2.6) следует, что граничная относительная ошибка представления чисел с плавающей запятой не зависит от величины порядка, определяется разрядностью мантиссы и практически одинакова для любых чисел, как малых, так и больших. Заметим, что этот вывод справедлив только для нормализованных чисел. Для ненормализованных чисел граничная относительная ошибка может достигать стопроцентного значения. Сопоставление диапазона и точности чисел подтверждает заключение, что в формате числа с плавающей запятой диапазон и точность чисел отделены друг от друга: разрядность мантиссы определяет точность представления чисел, а разрядность порядка — их диапазон.

Арифметические операции над числами с плавающей запятой выполняют действия как над мантиссами, так и

над порядками, причем те и другие представлены в виде знаковых чисел в соответствии с формулами (2.2) и (2.4). Операции над порядками (сложение, вычитание, сравнение) существенно упрощаются, если вместо их знакового представления (например, в дополнительном коде) использовать беззнаковое представление в виде неотрицательных чисел, или так называемое представление со «смещенным порядком», применяемое в ряде ЭВМ [30, 61, 65]:

$$m_{\text{см}} = m + R^k, \quad (2.7)$$

где  $m$  — несмещенный порядок в дополнительном коде;  $R^k$  — смещение;  $m_{\text{см}}$  — смещенный порядок. Диапазон представления порядка  $m$  с учетом выражений (2.4) равен  $-R^k \leq m \leq R^k - 1$ . Корректируя этот диапазон согласно формуле (2.7), получаем неравенство  $0 \leq m_{\text{см}} \leq 2R^k - 1$ , гарантирующее, что  $m_{\text{см}}$  — всегда неотрицательное число, представляемое теми же  $k + 1$  разрядами, что и несмещенный порядок  $m$  с  $k$  цифровыми и одним знаковым разрядами. Схема соответствия несмещенного и смещенного порядков для двоичных чисел ( $R = 2$ ) приведена на рис. 2.2. Очевидно важное свойство смещенных порядков, используемое при их сравнении: если  $m_1 \geq m_2$ , то всегда  $m_{1\text{ см}} \geq m_{2\text{ см}}$ .

На практике существует большое разнообразие представлений чисел с плавающей запятой, определяемое, во-первых, выбором различных систем счисления: двоичной ( $R = 2$ ) и двоично-смешанных ( $Q = 8, 10, 16$ ); во-вторых, различными способами кодирования знаковых чисел мантииссы и порядка: прямым, обратным, дополнительным или кодом со смещением; в-третьих, различными разрядностью формата и размещением порядка, мантииссы и их знаков относительно друг друга в формате числа [30, 56, 61, 65]. Например, в ЕС ЭВМ для представления короткого формата числа с плавающей запятой отводятся 32 двоичных разряда, из которых 25 используются для представления шестнадцатеричной мантииссы в прямом коде (6 шестнадцатеричных разрядов), а 7 — для представления шестнадцатеричной характеристики

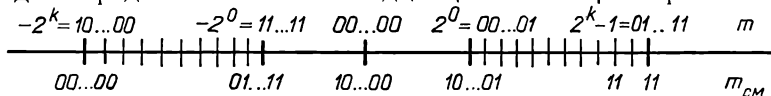


Рис. 2.2. Схема соответствия несмещенного  $m$  и смещенного  $m_{\text{см}}$  порядков

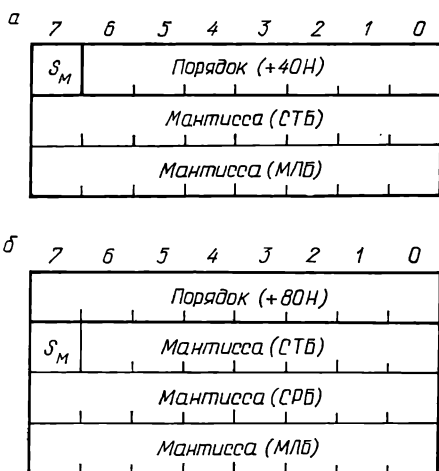


Рис. 2.3. Форматы двоичных чисел с плавающей запятой

со смещенным двоичным порядком (смещение  $+64$ ). В данном пособии используется представление двоичных чисел с плавающей запятой со смещенным порядком и мантиссой в дополнительном коде, причем числа имеют два формата: трехбайтный обычной точности и четырехбайтный повышенной точности.

На рис. 2.3, *a* изображен трехбайтный формат двоичного числа с плавающей запятой обычной точности. Над рамкой формата проставлены номера двоичных разрядов байта. Число хранится в памяти микропроцессорной системы в виде последовательности трех байтов, размещенных в порядке возрастания адресов памяти. Первый байт числа в старшем разряде содержит знак мантиссы  $S_M$  и смещенный порядок (смещение равно  $64_{10} = 40_{16}$ ), а два остальных байта — СТБ и МЛБ мантиссы. Сокращенно формат запишем в виде (8,16). Пограничные значения чисел этого формата указаны в табл. 2.1.

Диапазон представления абсолютных величин ненулевых чисел имеет вид

$$0,27 \cdot 10^{-19} \leq |A| \leq 0,92 \cdot 10^{19}. \quad (2.8)$$

Граничная относительная ошибка такого представления, согласно формуле (2.6), не превышает величины

$$\delta_r < 2^{-16+1} \approx 0,3 \cdot 10^{-4}. \quad (2.9)$$

Табл. 2.1. Пограничные числа формата (8,16)

A	A <sub>16</sub>			A <sub>10</sub>	
	m <sub>см</sub>	A <sub>16ф</sub>	m	2 <sup>m<sub>10</sub></sup> · A <sub>10ф</sub>	A <sub>10</sub>
+A <sub>max</sub>	7F	FFFF	+3F	+2 <sup>63</sup> · (1-2 <sup>-16</sup> )	+0,92 · 10 <sup>19</sup>
-A <sub>max</sub>	FF	0001	+3F	-2 <sup>63</sup> · (1-2 <sup>-16</sup> )	-0,92 · 10 <sup>19</sup>
+A <sub>min</sub>	00	8000	-40	+2 <sup>-64</sup> · 2 <sup>-1</sup>	+0,27 · 10 <sup>-19</sup>
-A <sub>min</sub>	80	8000	-40	-2 <sup>-64</sup> · 2 <sup>-1</sup>	-0,27 · 10 <sup>-19</sup>

Заметим, что неравенства (2.8) и (2.9) получены для нормализованных чисел, округленных несимметричным способом при их представлении в  $n$ -разрядном ( $n = 16$ ) формате мантиисы. В случае симметричного округления (1.9) величина  $\delta_r$  уменьшается в два раза. Для обеспечения эквивалентной точности и диапазона в случае чисел с фиксированной запятой потребовалась бы разрядность, в шесть раз бóльшая.

На рис. 2.3, б показан четырехбайтный формат двоичного числа с плавающей запятой повышенной точности. Первый байт содержит значение смещенного порядка (смещение 128<sub>10</sub> = 80<sub>16</sub>), а три остальных — СТБ, СРБ, МЛБ мантиисы, причем в старшем разряде СТБ мантиисы указывается знак мантиисы. Пограничные значения чисел формата (8,24) указаны в табл. 2.2.

Табл. 2.2. Пограничные числа формата (8, 24)

A	A <sub>16</sub>			A <sub>10</sub>	
	m <sub>см</sub>	A <sub>16ф</sub>	m	2 <sup>m<sub>10</sub></sup> · A <sub>10ф</sub>	A <sub>10</sub>
+A <sub>max</sub>	FF	7FFFFFFF	+7F	+2 <sup>127</sup> · (1-2 <sup>-23</sup> )	+0,17 · 10 <sup>39</sup>
-A <sub>max</sub>	FF	800001	+7F	-2 <sup>127</sup> · (1-2 <sup>-23</sup> )	-0,17 · 10 <sup>39</sup>
+A <sub>min</sub>	00	400000	-80	+2 <sup>-128</sup> · 2 <sup>-1</sup>	+1,4 · 10 <sup>-39</sup>
-A <sub>min</sub>	00	C00000	-80	-2 <sup>-128</sup> · 2 <sup>-1</sup>	-1,4 · 10 <sup>-39</sup>

Граничная относительная ошибка представления (8, 24) не превышает  $\delta_r \leq 2^{-31+1} \approx 0,9 \cdot 10^{-9}$ . Таким образом, данный формат существенно расширяет диапазон и точность обрабатываемых чисел. Отметим, что в ряде приложений арифметики с плавающей запятой нет необходимости в использовании формата повышенной точности, поскольку точность исходной информации в микропроцессорных системах зачастую ограничена классом



точности первичных датчиков на уровне  $0,1...0,05\%$ , т. е. превышает неточность представления чисел в формате (8, 16) в соответствии с формулой (2.9). Использование формата повышенной точности связано с дополнительными затратами памяти и времени работы микропроцессора и целесообразно для задач, требующих повышенной точности обработки или большего диапазона представления чисел. Числа с плавающей запятой являются числами ограниченной точности. Поэтому для арифметики с плавающей запятой справедливы все свойства АОТ, рассмотренные в § 1.1. Условимся в дальнейшем изображать нулевое число с плавающей запятой в виде нулевой мантииссы  $A_{2ф} = 0$  и минимального порядка  $m = m_{\min}$ . Поскольку порядок имеет смещение, то, очевидно,  $m_{\text{см min}} = 0$ .

## 2.2. СЛОЖЕНИЕ И ВЫЧИТАНИЕ ДВОИЧНЫХ ЧИСЕЛ

### 2.2.1. МЕТОДИКА СЛОЖЕНИЯ И ВЫЧИТАНИЯ

Ранее, в § 1.2, было показано, что сложение чисел с фиксированной запятой — точная операция, поскольку она не использует округление результата и не вносит в него какую-либо дополнительную погрешность, кроме погрешностей слагаемых. Если слагаемые точны, то точна и сумма. Если слагаемые — числа ограниченной точности, то точность суммы определяется выражениями (1.15) и (1.16). В отличие от этого сложение чисел с плавающей запятой является в общем случае неточной операцией, вносящей дополнительную погрешность в результат. Поэтому сумма чисел с плавающей запятой даже при точных слагаемых будет, вообще говоря, числом ограниченной точности. Операцию вычитания чисел с плавающей запятой, представленных в дополнительных кодах, можно свести к операции сложения уменьшаемого с дополнительным кодом вычитаемого. Поэтому в дальнейшем обсудим операцию алгебраического сложения (выполняет сложение и вычитание чисел в зависимости от знаков слагаемых).

Рассмотрим сложение  $z = x + y$  двоичных чисел с плавающей запятой:  $x = 2^{m_x} \cdot X$  и  $y = 2^{m_y} \cdot Y$ . Если порядки слагаемых равны ( $m_x = m_y$ ), то мантииссы  $X$  и  $Y$  можно непосредственно сложить, поскольку фактическое положение запятой в обоих слагаемых одинаково, т. е. слагаемые совмещены разрядами мантиисс с одинаковыми весами.

В общем случае, когда числа имеют различные порядки ( $m_x \neq m_y$ ), непосредственно складывать мантиссы нельзя, поскольку фактическое положение запятой в слагаемых различно, т. е. они совмещены разрядами мантисс с несовпадающими весами. Поэтому необходимо предварительно преобразовать слагаемое с меньшим порядком: увеличить его порядок до величины большего порядка и соответствующим образом уменьшить значение его мантиссы, чтобы в целом сохранить величину числа. Эту операцию называют *выравниванием порядков*. При ее выполнении прежде всего определяется разность порядков  $\Delta m = m_x - m_y$ .

Увеличение порядка слагаемого с меньшим порядком, например  $y$ , равносильно умножению  $y \cdot 2^{\Delta m}$ . Тогда уменьшение мантиссы  $Y$  эквивалентно делению  $Y : 2^{\Delta m}$  (или умножению  $Y \cdot 2^{-\Delta m}$ ), что обеспечивается сдвигом мантиссы  $Y$  вправо — *денормализацией мантиссы* — на  $\Delta m$  разрядов. С учетом этого процесса сложение чисел с плавающей запятой можно определить следующими выражениями:

$$z = 2^{m_x} \cdot X + 2^{m_y} \cdot Y = 2^{m'_z} \cdot Z' = 2^{m_z} \cdot Z; \quad (2.10)$$

$$Z' = \begin{cases} X + Y \cdot 2^{-|\Delta m|}, & \text{если } \Delta m > 0; \\ Y + X \cdot 2^{-|\Delta m|}, & \text{если } \Delta m < 0; \\ X + Y, & \text{если } \Delta m = 0; \end{cases} \quad (2.11)$$

$$m'_z = \max(m_x, m_y);$$

$$Z, m_z = \begin{cases} Z'; m'_z, & \text{если } Z' < 1; \\ Z' \cdot 2^{-1}; m'_z + 1, & \text{если } Z' \geq 1, \end{cases} \quad (2.12)$$

где  $\Delta m = m_x - m_y$ .

Формула (2.11) определяет процесс выравнивания порядков и алгебраического суммирования мантисс с равными порядками по правилам арифметики с фиксированной запятой. При сложении мантисс возможны две ситуации: *переполнение мантиссы суммы* и *отсутствие переполнения*. В первом случае необходимо устранить переполнение путем сдвига мантиссы вправо на один двоичный разряд (делением на 2) и соответствующим увеличением на 1 порядка суммы. Эту операцию называют *нормализацией вправо*. Во втором случае необходимость в нормализации вправо отпадает. Действия на этапе нормализации вправо отражает формула (2.12).

При сложении близких по величине чисел с **разными** знаками возможно уменьшение значности мантиссы

суммы, т. е. появление в левых разрядах формата мантиссы незначащих нулей. В этом случае необходимо нормализовать мантиссу согласно условию (2.3). Эта операция, называемая *нормализацией влево*, заключается в сдвиге мантиссы влево на  $r$  ( $r < n$ ) разрядов (умножении мантиссы на  $2^r$ ) и соответствующем уменьшении порядка суммы:

$$Z, m_z = \begin{cases} Z'; m'_z, & \text{если } 2^{-1} \leq |Z| < 1; \\ Z' \cdot 2^r; m'_z - r, & \text{если } |Z| < 2^{-1}, \end{cases} \quad (2.13)$$

где  $r$  — число левых нулевых двоичных разрядов мантиссы суммы. При программной реализации нормализации влево величина  $r$ , как правило, не определяется, а сдвиги влево и коррекция порядка производятся до тех пор (при ненулевой мантиссе), пока не будет выполнено условие (2.3).

При реализации процессов, описываемых формулами (2.10) — (2.13), возможно появление четырех особых ситуаций, которые должны выявляться программными средствами. При выравнивании порядков возможен случай, когда  $\Delta m \geq n$ , т. е. разность порядков слагаемых превышает разрядность цифровой части мантиссы. Очевидно, что при такой денормализации мантиссы меньшего слагаемого она полностью выходит за рамки разрядной сетки, т. е. становится равной нулю. В этом случае результат равен большему слагаемому, и нет необходимости выполнять сложение. При сложении равных чисел с разными знаками мантисса суммы равна нулю. Такая нулевая мантисса не может быть нормализована, и факт ее появления используется для обнуления порядка суммы. При нормализации мантиссы суммы вправо возможно *переполнение порядка суммы*:  $m_z = m'_z + 1 > m_{\max}$ , а при нормализации влево — *антипереполнение порядка суммы*:  $m = m'_z - r < m_{\min}$ . Появление таких ошибок свидетельствует о неправильном выборе формата с точки зрения представления диапазона чисел в конкретном вычислительном процессе. Такие ошибки должны обязательно выявляться, и в случае переполнения необходимо прекратить процесс вычислений.

Оценим точность операции сложения чисел с плавающей запятой. Эта точность ограничивается округлением чисел при их сдвигах за рамки основного формата в процессах денормализации мантиссы меньшего слагаемого, нормализации суммы вправо и влево, а также при

обнулении результата в случае антипереполнения порядка [32—34]. Основная погрешность возникает при денормализации мантиссы, поскольку при последующей нормализации суммы вправо эта первоначальная погрешность может только уменьшиться, а при нормализации суммы влево погрешность можно вообще устранить, если сохранить хотя бы два первых выдвинутых разряда денормализованной мантиссы и использовать их при нормализации суммы влево (незначащие нули могут появиться лишь при вычитании близких чисел с  $\Delta m \leq 2$ ). Ошибку обнуления результата при антипереполнении порядка учитывать не будем, считая, что либо эта ошибка в процессе вычислений не возникает, либо процесс вычислений останавливается при ее появлении.

Если бы при сложении нормализованной мантиссы большего слагаемого с денормализованной мантиссой меньшего слагаемого в соответствии с формулой (2.11) использовались все  $n + \Delta m$  разрядов слагаемых, то сумма была бы точным числом (при условии, что точны слагаемые). Однако выдвинутые за рамки  $n$ -разрядного формата денормализованные разряды мантиссы меньшего слагаемого не участвуют в сложении, т. е. сумма образуется путем округления. Очевидно, что в случае отбрасывания  $\Delta m$  разрядов граничная абсолютная ошибка округления мантиссы суммы не превышает величины  $2^{-n}$ , а граничная относительная ошибка округления суммы (в наихудшем случае отсутствия переполнения суммы и при минимальном значении ее нормализованной мантиссы) равна

$$\delta_{г.окр}(z) = \frac{\Delta_{г.окр}(z)}{z_{\min}} = \frac{2^{m_2} \cdot 2^{-n}}{2^{m_2} \cdot 2^{-1}} = 2^{-n+1}. \quad (2.14)$$

Исходные слагаемые являются числами ограниченной точности, граничные относительные ошибки представления которых определяются неравенством (2.6). Согласно формуле (1.14), граничная относительная ошибка суммы не превосходит ошибки наименее точного слагаемого. Однако округление суммы увеличивает эту ошибку на величину, вычисленную по формуле (2.14), т. е. в 2 раза, и сумма становится менее точной, чем слагаемые. Ошибку округления можно уменьшить, если использовать правило симметричного округления (1.9).

Далее приведены программы сложения чисел с плавающей запятой СДПЗЗ и СДПЗ4, реализующие рассмотрен-

ные алгоритмы сложения для короткого (8,16) и длинного (8, 24) форматов чисел.

### 2.2.2. ФОРМАТ (8, 16) + (8, 16) = (8, 16)

Программа СДПЗЗ выполняет сложение чисел, представленных в формате (8, 16) (см. рис. 2.3, а):

1000		ORG	1000H
0090	КОМЗ	SET	90H
00B8	ПМЗ	SET	0B8H
00D6	ОВМЗ	SET	0D6H
10B0	ДМАН2	SET	10B0H
10D0	НМАН2	SET	10D0H
10F1	ПМАН2	SET	10F1H

СДПЗЗ:

```

;*****
;ПОДПРОГРАММА СЛОЖЕНИЯ 3- БАЙТНЫХ ДВОИЧНЫХ ЧИСЕЛ С ПЛА-
;ВАЮЩЕЙ ЗАПЯТОЙ В ДОПОЛНИТЕЛЬНОМ КОДЕ ФОРМАТА (8,16)=
;=(ПОР,МАН),ГДЕ БАЙТ ПОРЯДКА СОДЕРЖИТ БИТ ЗНАКА МАНТИССЫ
;И ЦЕЛОЧИСЛЕННЫЙ ДВОИЧНЫЙ ПОРЯДОК СО СМЕЩЕНИЕМ +40H,А
;ДВА БАЙТА МАНТИССЫ-СТБ И МЛБ-ДВОИЧНОЕ ДРОБНОЕ НОРМАЛИ-
;ЗОВАННОЕ ЧИСЛО В ДОПОЛНИТЕЛЬНОМ КОДЕ.
;ВХОДНЫЕ ПАРАМЕТРЫ: (В,С)-АДРЕС СЛАГАЕМОГО 1, (Н,Л)-АДРЕС
;СЛАГАЕМОГО 2.ВЫХОДНЫЕ ПАРАМЕТРЫ: (В,С)-АДРЕС СУММЫ (СУМ-
;МА НА МЕСТЕ СЛАГАЕМОГО 1),СУ=1-ПРИЗНАК ПЕРЕПОЛНЕНИЯ ИЛИ
;АНТИПЕРЕПОЛНЕНИЯ ПОРЯДКА СУММЫ.ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИ-
;СТРЫ,СОХРАНЯЮТСЯ (В,С), (Н,Л).ГЛУБИНА СТЕКА-10.ИСПОЛЬЗУ-
;ЮТСЯ ПОДПРОГРАММЫ:*КОМЗ*,*ПМЗ*,*ОВМЗ*,*ДМАН2*,*ПМАН2*,
;*НМАН2*.
;ОЦЕНКА:ДЛИНА-161 БАЙТ (+127 БАЙТ ПОДПРОГРАММ),ВРЕМЯ-НЕ
;ВОЛЕЕ 2563 ТАКТОВ (С УЧЕТОМ ПОДПРОГРАММ).
;*****

```

```

;ПРОВЕРКА СЛАГАЕМОГО 2 (СЛ2) НА НОЛЬ
1000 CD9000      CALL    КОМЗ      ;Z=1,ЕСЛИ СЛ2=0
1003 C8          RZ              ;ЕСЛИ СЛ2=0, (СУ=0)
;ПРОВЕРКА СЛАГАЕМОГО 1 (СЛ1) НА НОЛЬ
1004 C5          PUSH    В        ;СОХРАНЕНИЕ АДРЕСА СЛ1
1005 50          MOV     Д,В
1006 59          MOV     Е,С      ; (Д,Е)-АДРЕС СЛ1
1007 EB          XCHG
1008 CD9000      CALL    КОМЗ      ;Z=1,ЕСЛИ СЛ1=0
100B EB          XCHG
100C C21410      JNZ     ПЕР1      ;ЕСЛИ СЛ1 НЕ НОЛЬ
;ПЕРЕМЕЩЕНИЕ СЛ2 НА МЕСТО СЛ1 (СЛ1=0,СЛ2 НЕ 0)
100F CDB800      CALL    ПМЗ      ;СЛ2 НА МЕСТЕ СЛ1
1012 C1          POP     В        ;ВОССТАНОВЛЕНИЕ АДРЕСА СЛ1
1013 C9          RET             ;ЕСЛИ СЛ1=0, (СУ=0)
;ОБА СЛАГАЕМЫЕ НЕ НОЛЬ.ПОЛУЧЕНИЕ МОДИФИЦИРОВАННЫХ КОДОВ
;ЗНАКОВ СЛАГАЕМЫХ В (В,С):00="+",11="-"
1014 1A          ПЕР1: LDAX    Д      ; (А)-ПОР1
1015 E680        ANI     80H        ;ВЫДЕЛЕНИЕ ЗНАКА
1017 07          RLC
1018 17          RAL
1019 47          MOV     В,А      ; (В)-КОД ЗНАКА СЛ1
101A 7E          MOV     А,М

```

101B E680	ANI	80H	
101D 07	RLC		
101E 17	RAL		
101F 4F	MOV	C,A	; (C)-КОД ЗНАКА СЛ2
;ОПРЕДЕЛЕНИЕ РАЗНОСТИ ПОРЯДКОВ			
1020 E5	PUSH	H	;СОХРАНЕНИЕ АДРЕСА СЛ2
1021 C5	PUSH	B	;СОХРАНЕНИЕ КОДОВ ЗНАКОВ
1022 7E	MOV	A,M	; (A)-ПОР2
1023 E67F	ANI	7FH	;ИСКЛЮЧЕНИЕ ЗНАКА
1025 47	MOV	B,A	; (B)-ПОР2
1026 1A	LDAH	D	; (A)-ПОР1
1027 E67F	ANI	7FH	;ИСКЛЮЧЕНИЕ ЗНАКА
1029 90	SUB	B	; (A)=ПОР1-ПОР2
102A CA4710	JZ	ПЕР2	;ЕСЛИ ПОР1=ПОР2
102D D23510	JNC	ПЕР3	;ЕСЛИ ПОР1>ПОР2
1030 2F	CMA		;ДОПОЛНЕНИЕ ПРИ ПОР2>ПОР1
1031 3C	INR	A	; (A)=/ПОР1-ПОР2/
1032 CDD600	CALL	ОБМ3	;ОБМЕН СЛАГАЕМЫМИ:СЛ2 НА СЛ1
;ПРОВЕРКА ВЕЛИЧИНЫ РАЗНОСТИ ПОРЯДКОВ: < 16?			
1035 FE10	ПЕР3: CPI	16	
1037 FA3F10	JM	ПЕР4	;ЕСЛИ РАЗНОСТЬ < 16
103A AF	XRA	A	;CY=0
103B C1	POP	B	;БАЛАНС СТЕКА
103C E1	POP	H	;ВОССТАНОВЛЕНИЕ АДРЕСА СЛ2
103D C1	POP	B	;ВОССТАНОВЛЕНИЕ АДРЕСА СЛ1
103E C9	RET		;ЕСЛИ РАЗНОСТЬ ПОРЯДКОВ > 16
;ЗАГРУЗКА СЛ2 В РЕГИСТРЫ,ДЕНОРМАЛИЗАЦИЯ МАНТИССЫ СЛ2			
103F D5	ПЕР4: PUSH	D	;СОХРАНЕНИЕ АДРЕСА СЛ1
1040 CDB010	CALL	ДМАН2	; (B,C,D)-МАНТИССА
1043 E1	POP		; (H,L)-АДРЕС СЛ1
1044 C34F10	JMP	ПЕР5	
;ЗАГРУЗКА СЛ2 В РЕГИСТРЫ (B,C), (D)=0			
1047 D5	ПЕР2: PUSH	D	;СОХРАНЕНИЕ АДРЕСА СЛ1
1048 23	INX	H	
1049 46	MOV	B,M	; (B)-СТБ МАН2
104A 23	INX	H	
104B 4E	MOV	C,M	; (C)-МЛБ МАН2
104C 1600	MVI	D,0	
104E E1	POP	H	; (H,L)-АДРЕС СЛ1
;СЛОЖЕНИЕ МАНТИСС: (M)+(B,C)=(B,C)			
104F 23	ПЕР5: INX	H	
1050 23	INX	H	;АДРЕС МЛБ МАН1
1051 7E	MOV	A,M	
1052 81	ADD	C	
1053 4F	MOV	C,A	
1054 2B	DCX	H	
1055 7E	MOV	A,M	
1056 8B	ADC	B	
1057 47	MOV	B,A	; (B,C)-СУММА МАНТИСС
;АНАЛИЗ СУММЫ НА ПЕРЕПОЛНЕНИЕ ПО МОДИФИЦИРОВАННЫМ КОДАМ			
1058 2B	DCX	H	; (H,L)-АДРЕС ПОР1
1059 E3	XTHL		; (H,L)-КОДЫ ЗНАКОВ СЛ1,СЛ2
105A 7D	MOV	A,L	
105B 8C	ADC	H	;СЛОЖЕНИЕ ЗНАКОВ И ПЕРЕНОСА
105C E603	ANI	3	;ВЫДЕЛЕНИЕ КОДА ЗНАКА СУММЫ

105E E1	POP	H	; (H,L) - АДРЕС ПОР1
	; ЗАНЕСЕНИЕ ЗНАКА СУММЫ В БАЙТ ПОРЯДКА РЕЗУЛЬТАТА		
105F F5	PUSH	PSW	; СОХРАНЕНИЕ ЗНАКОВ
1060 1F	RAR		
1061 1F	RAR		; (CY) - ЗНАК СУММЫ
1062 7E	MOV	A,M	; (A) - ПОР1 СО ЗНАКОМ СЛ1
1063 17	RAL		
1064 0F	RRC		; (A) - ПОР1 СО ЗНАКОМ СУММЫ
1065 77	MOV	M,A	
1066 F1	POP	PSW	; ВОССТАНОВЛЕНИЕ ЗНАКОВ
1067 E28010	JFO	PER8	; ЕСЛИ ПЕРЕПОЛНЕНИЕ: 01 ИЛИ 10
	; АНАЛИЗ ОСОБОГО СЛУЧАЯ ПЕРЕПОЛНЕНИЯ ДЛЯ КОДА ЗНАКОВ 11		
106A FE03	CFI	3	
106C C27610	JNZ	PER6	; ЕСЛИ НЕ КОД 3
106F F5	PUSH	PSW	; СОХРАНИТЬ ПРИЗНАКИ
1070 78	MOV	A,B	
1071 B1	ORA	C	
1072 CA7F10	JZ	PER7	; ЕСЛИ МАНТИССА=0
1075 F1	POP	PSW	; ВОССТАНОВИТЬ ПРИЗНАКИ
	; НОРМАЛИЗАЦИЯ МАНТИССЫ ВЛЕВО		
1076 CDD010	PER6: CALL	ММАН2	; (B,C) - НОРМАЛИЗОВАННАЯ МАНТИССА
1079 D28A10	JNC	PER9	; НЕТ АНТИПЕРЕПОЛНЕНИЯ ПОРЯДКА
107C C39A10	JMP	PER10	; АНТИПЕРЕПОЛНЕНИЕ ПОРЯДКА
	; ПЕРЕПОЛНЕНИЕ: НОРМАЛИЗАЦИЯ МАНТИССЫ ВПРАВО		
107F F1	PER7: POP	PSW	; ВОССТАНОВИТЬ ПРИЗНАКИ
1080 1F	PER8: RAR		; CY - БИТ ПЕРЕПОЛНЕНИЯ
1081 CDF110	CALL	ПМАН2	; (B,C) - НОРМАЛИЗОВАННАЯ МАНТИССА
1084 D28A10	JNC	PER9	; НЕТ ПЕРЕПОЛНЕНИЯ ПОРЯДКА
1087 C39A10	JMP	PER10	; ПЕРЕПОЛНЕНИЕ ПОРЯДКА
	; ОКРУГЛЕНИЕ РЕЗУЛЬТАТА, НОРМАЛИЗАЦИЯ МАНТИССЫ		
108A AF	PER9: XRA	A	
108B 82	ADD	D	
108C F29A10	JP	PER10	; ЕСЛИ КОРРЕКЦИЯ=0
108F 79	MOV	A,C	
1090 C601	ADI	01	
1092 4F	MOV	C,A	
1093 78	MOV	A,B	
1094 CE00	ACI	0	
1096 47	MOV	B,A	
1097 DCF110	CC	ПМАН2	; (B,C) - НОРМАЛИЗОВАННАЯ МАНТИССА
	; ЗАПИСЬ РЕЗУЛЬТАТА В ПАМЯТЬ		
109A 23	PER10: INX	H	
109B 70	MOV	M,B	
109C 23	INX	H	
109D 71	MOV	M,C	
109E E1	POP	H	; ВОССТАНОВЛЕНИЕ АДРЕСА СЛ2
109F C1	POP	B	; ВОССТАНОВЛЕНИЕ АДРЕСА СЛ1
10A0 C9	RET		; CY=1, ЕСЛИ ОШИБКА ПОРЯДКА
0000	END		

Прежде всего программа проверяет слагаемые на нулевое значение, обращаясь к вспомогательной подпрограмме КОМ3:

0090

ORG 90H

КОМ3:

```

;*****
;ПОДПРОГРАММА КОНТРОЛЯ МАССИВА ИЗ ТРЕХ БАЙТ В ПАМЯТИ
;НА НОЛЬ.
;ВХОДНОЙ ПАРАМЕТР: (H,L)-НАЧАЛЬНЫЙ АДРЕС МАССИВА.ВЫХОД-
;НОЙ ПАРАМЕТР: Z=1-ПРИЗНАК РАВЕНСТВА МАССИВА НУЛЮ.ИСПОЛЬ-
;ЗУЕТСЯ РЕГИСТР А,СОХРАНЯЕТСЯ (H,L).
;ОЦЕНКА:ДЛИНА-8 БАЙТ,ВРЕМЯ -51 ТАКТ.
;*****

```

```

0090 7E      MOV     A,M
0091 23      INX     H
0092 B6      ORA     M
0093 23      INX     H
0094 B6      ORA     M
0095 2B      DCX     H
0096 2B      DCX     H
0097 C9      RET             ;Z=1,ЕСЛИ МАССИВ=0

```

КОМ4:

```

;*****
;ПОДПРОГРАММА КОНТРОЛЯ МАССИВА ИЗ ЧЕТЫРЕХ БАЙТ ПАМЯТИ
;НА НОЛЬ.
;ВХОДНОЙ ПАРАМЕТР: (H,L)-НАЧАЛЬНЫЙ АДРЕС МАССИВА.ВЫХОДНОЙ
;ПАРАМЕТР: Z=1-ПРИЗНАК РАВЕНСТВА МАССИВА НУЛЮ.ИСПОЛЬЗУЕТ-
;СЯ РЕГИСТР А,СОХРАНЯЕТСЯ (H,L).
;ОЦЕНКА:ДЛИНА-11 БАЙТ,ВРЕМЯ-68 ТАКТОВ.
;*****

```

```

0098 7E      MOV     A,M
0099 23      INX     H
009A B6      ORA     M
009B 23      INX     H
009C B6      ORA     M
009D 23      INX     H
009E B6      ORA     M
009F 2B      DCX     H
00A0 2B      DCX     H
00A1 2B      DCX     H
00A2 C9      RET             ;Z=1,ЕСЛИ МАССИВ=0
0000      END

```

Аналогичная подпрограмма КОМ4 используется ниже в программе СДПЗ4 для проверки на нуль четырехбайтных чисел. Поскольку сумма размещается на месте первого слагаемого, то при равенстве нулю второго слагаемого нет необходимости выполнять сложение; аналогично при равенстве нулю первого слагаемого сумма равна второму слагаемому, перемещенному на место первого. Перемещение выполняет подпрограмма ПМЗ:

00B8

ORG 00B8H

ПМЗ:

```

;*****
;ПОДПРОГРАММА ПЕРЕМЕЩЕНИЯ ТРЕХ БАЙТ МАССИВА ИЗ ОДНОЙ
;ОБЛАСТИ ПАМЯТИ В ДРУГУЮ.
;ВХОДНЫЕ ПАРАМЕТРЫ: (D,E)-НАЧАЛЬНЫЙ АДРЕС МАССИВА-ПРИЕМ-

```



;НИКА, (H,L) — НАЧАЛЬНЫЙ АДРЕС МАССИВА-ИСТОЧНИКА. ИСПОЛЬЗУ-  
 ;ЕТСЯ РЕГИСТР А, СОХРАНЯЮТСЯ (H,L), (D,E).  
 ;ОЦЕНКА: ДЛИНА—15 БАЙТ, ВРЕМЯ—92 ТАКТА.

;\*\*\*\*\*

```

00B8 7E      MOV     A,M
00B9 12      STAX    D      ;ПЕРЕСЛАТЬ БАЙТ 1
00BA 23      INX     H
00BB 13      INX     D
00BC 7E      MOV     A,M
00BD 12      STAX    D      ;ПЕРЕСЛАТЬ БАЙТ 2
00BE 23      INX     H
00BF 13      INX     D
00C0 7E      MOV     A,M
00C1 12      STAX    D      ;ПЕРЕСЛАТЬ БАЙТ 3
00C2 2B      DCX     H
00C3 2B      DCX     H
00C4 1B      DCX     D
00C5 1B      DCX     D
00C6 C9      RET

```

ПМ4:

;\*\*\*\*\*

;ПОДПРОГРАММА ПЕРЕМЕЩЕНИЯ ЧЕТЫРЕХ БАЙТ МАССИВА ИЗ ОДНОЙ  
 ;ОБЛАСТИ ПАМЯТИ В ДРУГУЮ.

;ВХОДНЫЕ ПАРАМЕТРЫ: (D,E) — НАЧАЛЬНЫЙ АДРЕС МАССИВА-ПРИЕМ-  
 ;НИКА, (H,L) — НАЧАЛЬНЫЙ АДРЕС МАССИВА-ИСТОЧНИКА. ИСПОЛЬЗУ-  
 ;ЮТСЯ РЕГИСТРЫ А, С, СОХРАНЯЮТСЯ (H,L), (D,E). ГЛУБИНА СТЕ-  
 ;КА—4.

;ОЦЕНКА: ДЛИНА—15 БАЙТ, ВРЕМЯ—215 ТАКТОВ.

;\*\*\*\*\*

```

00C7 0E04    MVI     C,4      ;СЧЕТЧИК ЦИКЛОВ
00C9 E5      PUSH    H
00CA D5      PUSH    D
                ;ПЕРЕСЫЛКА ОЧЕРЕДНОГО БАЙТА
00CB 7E      MOV     A,M      ЦИКЛ:
00CC 12      STAX    D
00CD 23      INX     H
00CE 13      INX     D
                ;ПРОВЕРКА КОНЦА ЦИКЛА
00CF 0D      DCR     C
00D0 C2CB00  JNZ     ЦИКЛ      ;ЗАЦИКЛИВАНИЕ
00D3 D1      POP     D
00D4 E1      POP     H
00D5 C9      RET
00D6 00      END

```

Аналогичная подпрограмма ПМ4 используется ниже в программе СДПЗ4 для перемещения четырехбайтного слагаемого (подпрограмма ПМ4 в отличие от бесцикло- вой ПМ3 организует пересылку байтов в цикле, что умень- шает длину подпрограммы). Использование подпрограмм КОМЗ и ПМЗ существенно сокращает среднее время вы- полнения программы СДПЗЗ, поскольку нулевые слагае- мые достаточно часто встречаются в вычислительных про- цессах. При отсутствии необходимости проверки слагае- мых на нуль программу СДПЗЗ можно упростить на 20

байт и ускорить на 180 тактов, исключив фрагменты проверки слагаемых.

При сложении мантисс в дополнительных кодах необходимо выявлять переполнение суммы. В программах сложения с этой целью выполнялся анализ знаков слагаемых и суммы. Другой способ обнаружения переполнения, используемый в данной программе, заключается в применении *модифицированного дополнительного кода* [13, 61]. В таком коде для представления знака используются два двоичных разряда: плюс кодируется 00, а минус — 11. Дублирование знака позволяет автоматически фиксировать переполнение: код 01 определяет переполнение положительной суммы, а код 10 — отрицательной. Исключением из этого правила является случай суммирования дополнительных кодов двух отрицательных чисел, таких, что  $[X]_д + [Y]_д = 11,0...0$ , т. е.  $|X + Y| = 1$  — переполнение.

Непосредственно сложение ненулевых слагаемых начинается с формирования модифицированных кодов знаков слагаемых и их запоминания с целью последующего анализа после выполнения сложения. Далее программа определяет разность порядков: ее абсолютную величину и знак. Когда порядок первого слагаемого оказывается меньше порядка второго, слагаемые меняются местами с помощью подпрограммы ОБМЗ:

00D6		ORG	0D6H
	ОБМЗ:		
	;*****		
	;ПОДПРОГРАММА ОБМЕНА СОДЕРЖИМЫМ ДВУХ ТРЕХБАЙТНЫХ МАССИ-		
	;ВОВ,РАЗМЕШЕННЫХ В НЕПЕРЕСЕКАЮЩИХСЯ ОБЛАСТЯХ ПАМЯТИ.		
	;ВХОДНЫЕ ПАРАМЕТРЫ: (D,E)-АДРЕС МАССИВА 1, (H,L)-АДРЕС		
	;МАССИВА 2.ИСПОЛЬЗУЮТСЯ И СОХРАНЯЮТСЯ ВСЕ РЕГИСТРЫ,ГЛУ-		
	;БИНА СТЕКА-2.		
	;ОЦЕНКА:ДЛИНА-28 БАЙТ,ВРЕМЯ-180 ТАКТОВ.		
	;*****		
00D6 C5	PUSH	B	
00D7 47	MOV	B,A	;СОХРАНЕНИЕ (A)
	;ОБМЕН ПЕРВЫМИ БАЙТАМИ		
00D8 4E	MOV	C,M	
00D9 1A	LDAX	D	
00DA 77	MOV	M,A	
00DB 79	MOV	A,C	
00DC 12	STAX	D	
00DD 13	INX	D	
00DE 23	INX	H	
	;ОБМЕН ВТОРЫМИ БАЙТАМИ		
00DF 4E	MOV	C,M	
00E0 1A	LDAX	D	
00E1 77	MOV	M,A	

00E2 79	MOV	A,C	
00E3 12	STAX	D	
00E4 13	INX	D	
00E5 23	INX	H	
	;ОБМЕН ТРЕТЬИМИ БАЙТАМИ		
00E6 4E	MOV	C,M	
00E7 1A	LDAX	D	
00E8 77	MOV	M,A	
00E9 79	MOV	A,C	
00EA 12	STAX	D	
	;ВОССТАНОВЛЕНИЕ АДРЕСОВ И РЕГИСТРОВ		
00EB 2B	DCX	H	
00EC 2B	DCX	H	
00ED 1B	DCX	D	
00EE 1B	DCX	D	
00EF 78	MOV	A,B	;ВОССТАНОВЛЕНИЕ (A)
00F0 C1	POP	B	
00F1 C9	RET		
0000	END		

Такой обмен повышает регулярность структуры программы и ее ясность, поскольку однозначно понимается, что большее слагаемое находится всегда на месте первого слагаемого. Этот прием уменьшает в дальнейшем количество проверок и длину программы. Но, с другой стороны, выполнение обмена слагаемых требует затрат и памяти, и времени. В данной программе компромисс сделан в пользу ясности.

Вычисленная разность порядков проверяется на предельное значение:  $\Delta m \leq n = 16$ ? При  $\Delta m \geq 16$  нет необходимости продолжать программу, так как результат совпадает со слагаемым, находящимся на месте первого слагаемого. В противном случае производится загрузка слагаемых из памяти в регистры микропроцессора, и подпрограмма ДМАН2 выполняет, если необходимо, денормализацию мантиссы меньшего слагаемого:

10B0	ORG	10B0H
	ДМАН2:	
	;*****	
	;ПОДПРОГРАММА ДЕНОРМАЛИЗАЦИИ ДВУХБАЙТНОЙ МАНТИССЫ ЧИСЛА	
	;С ПЛАВАЮЩЕЙ ЗАПЯТОЙ.	
	;ВХОДНЫЕ ПАРАМЕТРЫ: (A) -РАЗНОСТЬ ПОРЯДКОВ N,ОПРЕДЕЛЯЮЩАЯ	
	;ВЕЛИЧИНУ ДЕНОРМАЛИЗАЦИИ (N(16), (H,L) -АДРЕС ЧИСЛА С ПЛА-	
	;ВАЮЩЕЙ ЗАПЯТОЙ.ВЫХОДНЫЕ ПАРАМЕТРЫ: (B,C) -ДЕНОРМАЛИЗОВАН-	
	;НАЯ МАНТИССА, (D) -ВЫДВИНУТЫЕ РАЗРЯДЫ МАНТИССЫ.ИСПОЛЬЗУ-	
	;ЮТСЯ ВСЕ РЕГИСТРЫ,КРОМЕ L.	
	;ОЦЕНКА:ДЛИНА-25 БАЙТ,ВРЕМЯ-(58+66*N) ТАКТОВ	
	;*****	
	;ЗАГРУЗКА СЛАГАЕМОГО В РЕГИСТРЫ (H,B,C)	
10B0 5F	MOV	E,A
10B1 7E	MOV	A,M ; (A) -ПОР

10B2 23	INX	H	
10B3 46	MOV	B,M	; (B)-СТБ МАН
10B4 23	INX	H	
10B5 4E	MOV	C,M	; (C)-МЛБ МАН
10B6 67	MOV	H,A	; (H)-ПОР
10B7 1600	MVI	D,0	
	;СДВИГ МАНТИССЫ ВПРАВО НА N РАЗРЯДОВ В (B,C,D)		
10B9 7C	MOV	A,H	; (A)-ПОР
10BA 17	RAL		;СУ-ЗНАК МАНТИССЫ
10BB 78	MOV	A,B	
10BC 1F	RAR		
10BD 47	MOV	B,A	
10BE 79	MOV	A,C	
10BF 1F	RAR		
10C0 4F	MOV	C,A	
10C1 7A	MOV	A,D	
10C2 1F	RAR		
10C3 57	MOV	D,A	
	;ПРОВЕРКА КОНЦА ЦИКЛА		
10C4 1D	DCR	E	
10C5 C2B910	JNZ	ЦИКЛ	;ЗАЦИКЛИВАНИЕ
10C8 C9	RET		
0000	END		

Программа ДМАН2 сохраняет последние сдвинутые разряды мантиссы с целью их использования при округлении суммы. После сложения мантисс выполняется анализ суммы на переполнение, и знак суммы заносится в старший разряд байта порядка. При сложении отрицательных мантисс вида 8000H происходит то единственное переполнение, которое не фиксируется в модифицированном коде, но которое необходимо выявлять и устранять. Признаком такого переполнения является отрицательная сумма с нулевой мантиссой. Устранение переполнения производит программа ПМАН2:

10F1	ORG	10F1H
	ПМАН2:	
	;*****	
	;ПОДПРОГРАММА УСТРАНЕНИЯ ПЕРЕПОЛНЕНИЯ ДВУХБАЙТНОЙ МАН-	
	;ТИССЫ ЧИСЛА С ПЛАВАЮЩЕЙ ЗАПЯТОЙ.	
	;ВХОДНЫЕ ПАРАМЕТРЫ: (B,C)-МАНТИССА, (D)-ДОПОЛНИТЕЛЬНЫЕ	
	;МЛАДШИЕ РАЗРЯДЫ МАНТИССЫ,СУ-БИТ ПЕРЕПОЛНЕНИЯ, (H,L)-АД-	
	;РЕС ПОРЯДКА.ВХОДНЫЕ ПАРАМЕТРЫ: (B,C)-НОРМАЛИЗОВАННАЯ	
	;МАНТИССА, (H,L)-АДРЕС ПОРЯДКА,СУ=1-ПРИЗНАК ПЕРЕПОЛНЕНИЯ	
	;ПОРЯДКА.ИСПОЛЬЗУЮТСЯ РЕГИСТРЫ (B,C), (D), (A).	
	;ОЦЕНКА:ДЛИНА-18 БАЙТ,ВРЕМЯ-НЕ БОЛЕЕ 93 ТАКОВ.	
	;*****	
	;СДВИГ ВПРАВО МАНТИССЫ В (B,C,D) С УЧЕТОМ ПЕРЕПОЛНЕНИЯ	
10F1 78	MOV	A,B
10F2 1F	RAR	
10F3 47	MOV	B,A
10F4 79	MOV	A,C
10F5 1F	RAR	

10F6 4F	MOV	C, A	
10F7 7A	MOV	A, D	
10F8 1F	RAR		
10F9 57	MOV	D, A	
;КОНТРОЛЬ ПОРЯДКА НА ПЕРЕПОЛНЕНИЕ И ЕГО КОРРЕКЦИЯ			
10FA 7E	PM2: MOV	A, M	; (A) - ПОРЯДОК
10FB 2F	CMA		
10FC E67F	ANI	7FH	; ИСКЛЮЧЕНИЕ ЗНАКА
10FE 37	STC		; CY=1
10FF C8	RZ		; ЕСЛИ ПОРЯДОК МАКСИМАЛЕН
1100 34	INR	M	
1101 AF	XRA	A	; CY=0
1102 C9	RET		
0000	END		

Программа ПМАН2 контролирует верхнюю границу порядка и в случае опасности переполнения порядка устанавливает признак переноса  $CY = 1$ , который используется для аварийного анализа в основной программе. Если в результате сложения не было переполнения мантиссы суммы, она проверяется на условие нормализации и, если необходимо, нормализуется подпрограммой НМАН2:

10D0		ORG	10D0H
НМАН2:			
;*****			
;ПОДПРОГРАММА НОРМАЛИЗАЦИИ ДВУХБАЙТНОЙ МАНТИССЫ ЧИСЛА			
;С ПЛАВАЮЩЕЙ ЗАПЯТОЙ.			
;ВХОДНЫЕ ПАРАМЕТРЫ: (B,C) -НОРМАЛИЗУЕМАЯ МАНТИССА, (D) -			
;МЛАДШИЕ РАЗРЯДЫ ДЕНОРМАЛИЗОВАННОЙ МАНТИССЫ, (H,L) -АДРЕС			
;ПОРЯДКА. ВЫХОДНЫЕ ПАРАМЕТРЫ: (B,C) -НОРМАЛИЗОВАННАЯ МАН-			
;ТИССА, (D) -ДОПОЛНИТЕЛЬНЫЕ МЛАДШИЕ РАЗРЯДЫ МАНТИССЫ,			
; (H,L) -АДРЕС ПОРЯДКА, CY=1 -ПРИЗНАК АНТИПЕРЕПОЛНЕНИЯ ПО-			
;РЯДКА. ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ, КРОМЕ (E), СОХРАНЯЕТСЯ			
; (H,L).			
;ОЦЕНКА: ДЛИНА-33 БАЙТА, ВРЕМЯ-НЕ БОЛЕЕ (57+122*(N-1))			
;ТАКТОВ, N<16			
;*****			
;ПРОВЕРКА МАНТИССЫ НА НОЛЬ			
10D0 78	MOV	A, B	; (A) -СТБ МАН
10D1 B1	ORA	C	
10D2 C2D710	JNZ	ЦИКЛ	; ЕСЛИ НЕ НОЛЬ
10D5 77	MOV	M, A	; ОБНУЛЕНИЕ ПОРЯДКА
10D6 C9	RET		; ЕСЛИ МАН=0, (CY=0)
;ПРОВЕРКА ЯВНО НОРМАЛИЗОВАННОЙ МАНТИССЫ: ЗНАКОВЫЙ И			
;СТАРШИЙ ЗНАЧАЩИЙ РАЗРЯД РАЗЛИЧНЫ ?			
10D7 78	ЦИКЛ: MOV	A, B	; (A) -СТБ МАН
10D8 AE	XRA	M	
10D9 F8	RM		; ЕСЛИ ЯВНАЯ НОРМАЛИЗАЦИЯ, (CY=0)
;ПРОВЕРКА НЕЯВНО НОРМАЛИЗОВАННОЙ МАНТИССЫ			
10DA 78	MOV	A, B	
10DB E67F	ANI	7FH	; ИСКЛЮЧЕНИЕ СТАРШЕГО РАЗРЯДА

10DD B1	ORA	C	
10DE C8	RZ		; ЕСЛИ НЕЯВНАЯ НОРМАЛИЗАЦИЯ, (CY=0)
			; НОРМАЛИЗАЦИЯ МАНТИССЫ: СДВИГ ВЛЕВО В (B, C, D)
10DF 7A	MOV	A, D	
10E0 17	RAL		
10E1 57	MOV	D, A	
10E2 79	MOV	A, C	
10E3 17	RAL		
10E4 4F	MOV	C, A	
10E5 78	MOV	A, B	
10E6 17	RAL		
10E7 47	MOV	B, A	
			; КОНТРОЛЬ НА АНТИПЕРЕПОЛНЕНИЕ ПОРЯДКА, ЕГО КОРРЕКЦИЯ
10E8 7E	MOV	A, M	; (A) - ПОРЯДОК
10E9 E67F	ANI	7FH	; ИСКЛЮЧЕНИЕ ЗНАКА
10EB 37	STC		; CY=1
10EC C8	RZ		; ЕСЛИ ПОРЯДОК МИНИМАЛЕН
10ED 35	DCR	M	
10EE C3D710	JMP	ЦИКЛ	; ЗАЦИКЛИВАНИЕ
0000	END		

Программа НМАН2 проверяет условие нормализации мантииссы, представленной в дополнительном коде (в этом случае наличие единицы в старшем цифровом разряде мантииссы еще не свидетельствует о ее нормализованности), и при нормализации влево контролирует порядок суммы на антипереполнение, устанавливая признак переноса  $CY=1$  при антипереполнении (порядок суммы равен нулю). В заключение сложения выполняются симметричное округление суммы, устранение, если необходимо, переполнения мантииссы после округления и ее запись в память на место первого слагаемого. Заметим, что симметричное округление по первой отбрасываемой цифре денормализованной мантииссы (в случае, если не было нормализации вправо), реализованное в программе, эффективно лишь при сложении чисел с одинаковыми знаками, поскольку только в этом случае знак отбрасываемой цифры совпадает со знаком суммы, и, следовательно, округление выполняется верно, иначе ошибка округления превысит ошибку несимметричного округления. Если такая ситуация нежелательна с точки зрения использования программы в системе, можно либо усложнить процесс симметричного округления в программе, введя анализ знака ошибки, либо упростить программу, исключив это округление.

Набор положительных и отрицательных чисел, представленных в формате (8,16), приведен в табл. 2.3. Эти числа можно использовать для тестовой проверки программы СДПЗЗ (в рамках представленного множества).

Табл. 2.3. Числа с плавающей запятой в формате (8, 16)

Положительные числа		Отрицательные числа	
$A_{16}$	$A_{10}$	$A_{10}$	$A_{16}$
408000	0,5	—0,5	C08000
418000	1	—1	C18000
41C000	1,5	—1,5	C14000
428000	2	—2	C28000
42A000	2,5	—2,5	C26000
42C000	3	—3	C24000
42E000	3,5	—3,5	C22000
438000	4	—4	C38000
439000	4,5	—4,5	C37000
43A000	5	—5	C36000

**2.2.3. ФОРМАТ (8,24) + (8,24) = (8,24)**

Программа СДПЗ4 выполняет сложение чисел, представленных в формате (8,24) (см. рис. 2.3, б).

1110		ORG	1110H
0098	КОМ4	SET	98H
00C7	ПМ4	SET	0C7H
00F2	ОБМ4	SET	0F2H
11B0	ДМАНЗ	SET	11B0H
11D0	НМАНЗ	SET	11D0H
1200	ПМАНЗ	SET	1200H

**СДПЗ4:**

```

;*****
;ПОДПРОГРАММА СЛОЖЕНИЯ 4-БАЙТНЫХ ДВОИЧНЫХ ЧИСЕЛ С ПЛА-
;ВЮЩЕЙ ЗАПЯТОЙ В ДОПОЛНИТЕЛЬНОМ КОДЕ ФОРМАТА (8,24)=
;=(ПОР,МАН),ГДЕ БАЙТ ПОРЯДКА СОДЕРЖИТ ЦЕЛОЧИСЛЕННЫЙ ДВО-
;ИЧНЫЙ ПОРЯДОК СО СМЕЩЕНИЕМ +ВОН,А 3 БАЙТА МАНТИССЫ-СТБ,
;СРБ,МЛБ-БИТ ЗНАКА МАНТИССЫ И ДВОИЧНОЕ ДРОБНОЕ НОРМАЛИ-
;ЗОВАННОЕ ЧИСЛО В ДОПОЛНИТЕЛЬНОМ КОДЕ.
;ВХОДНЫЕ ПАРАМЕТРЫ: (В,С)-АДРЕС СЛАГАЕМОГО 1, (Н,Л)-АДРЕС
;СЛАГАЕМОГО 2.ВЫХОДНЫЕ ПАРАМЕТРЫ: (В,С)-АДРЕС СУММЫ (СУМ-
;МА НА МЕСТЕ СЛАГАЕМОГО 1),СУ=1-ПРИЗНАК ПЕРЕПОЛНЕНИЯ ИЛИ
;АНТИПЕРЕПОЛНЕНИЯ ПОРЯДКА СУММЫ.ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИ-
;СТРЫ,СОХРАНЯЮТСЯ (В,С), (Н,Л).ГЛУБИНА СТЕКА-14.ИСПОЛЬЗУ-
;ЮТСЯ ПОДПРОГРАММЫ:*КОМ4*,*ПМ4*,*ОБМ4*,*ДМАНЗ*,*НМАНЗ*,
;*ПМАНЗ*.
;ОЦЕНКА:ДЛИНА-152 БАЙТ (+145 БАЙТ ПОДПРОГРАММ),ВРЕМЯ-
;НЕ БОЛЕЕ 3920 ТАКТОВ (С УЧЕТОМ ПОДПРОГРАММ)
;*****

```

1110	CD9800	CALL	КОМ4	;Z=1,ЕСЛИ СЛ2=0
1113	C8	RZ		;ЕСЛИ СЛ2=0, (СУ=0)
		;ПРОВЕРКА СЛАГАЕМОГО 1 (СЛ1) НА НОЛЬ		
1114	C5	PUSH	В	;СОХРАНЕНИЕ АДРЕСА СЛ1
1115	50	MOV	Д,В	

1116 59	MOV	E,C	; (D,E)-АДРЕС СЛ1
1117 EB	XCHG		
1118 CD9800	CALL	KOM4	; Z=1, ЕСЛИ СЛ1=0
111B EB	XCHG		
111C C22411	JNZ	ПЕР1	; ЕСЛИ СЛ1 НЕ НОЛЬ
	; ПЕРЕМЕЩЕНИЕ СЛ2 НА МЕСТО СЛ1 (СЛ1=0, СЛ2 НЕ 0)		
111F CDC700	CALL	ПМ4	; СЛ2 НА МЕСТО СЛ1
1122 C1	POP	B	; ВОССТАНОВЛЕНИЕ АДРЕСА СЛ1
1123 C9	RET		; ЕСЛИ СЛ1=0, (CY=0)
	; ОБА СЛАГАЕМЫЕ НЕ 0. ОПРЕДЕЛЕНИЕ РАЗНОСТИ ПОРЯДКОВ		
1124 E5	ПЕР1: PUSH	H	; СОХРАНЕНИЕ АДРЕСА СЛ2
1125 1A	LDAX	D	; (A)-ПОР1
1126 96	SUB	M	; (A)=ПОР1-ПОР2
1127 CA4411	JZ	ПЕР2	; ЕСЛИ ПОР1=ПОР2
112A D23211	JNC	ПЕР3	; ЕСЛИ ПОР1>ПОР2
112D 2F	CMA		; ДОПОЛНЕНИЕ ПРИ ПОР2>ПОР1
112E 3C	INR	A	; (A)=/ПОР1-ПОР2/
112F CDF200	CALL	ОВМ4	; ОБМЕН СЛАГАЕМЫМИ: СЛ1 НА СЛ2
	; ПРОВЕРКА ВЕЛИЧИНЫ РАЗНОСТИ ПОРЯДКОВ: <23?		
1132 FE17	ПЕР3: CPI	23	
1134 FA3B11	JM	ПЕР4	; ЕСЛИ РАЗНОСТЬ <23
1137 AF	XRA	A	; CY=0
1138 E1	POP	H	; ВОССТАНОВЛЕНИЕ АДРЕСА СЛ2
1139 C1	POP	B	; ВОССТАНОВЛЕНИЕ АДРЕСА СЛ1
113A C9	RET		; ЕСЛИ РАЗНОСТЬ ПОРЯДКОВ >22
	; ЗАГРУЗКА СЛ2 В РЕГИСТРЫ, ДЕНОРМАЛИЗАЦИЯ МАНТИССЫ СЛ2		
113B D5	ПЕР4: PUSH	D	; СОХРАНЕНИЕ АДРЕСА СЛ1
113C 23	INX	H	; АДРЕС СЛ2 МАН2
113D CDB011	CALL	ДМАН3	; (B,C,D,E)-МАНТИССА
1140 E1	POP	H	; (H,L)-АДРЕС СЛ1
1141 C34E11	JMP	ПЕР5	
	; ЗАГРУЗКА СЛ2 В РЕГИСТРЫ (B,C,D) И (E)=0		
1144 D5	ПЕР2: PUSH	D	; СОХРАНЕНИЕ АДРЕСА СЛ1
1145 23	INX	H	
1146 46	MOV	B,M	; (B)-СЛБ МАН2
1147 23	INX	H	
1148 4E	MOV	C,M	; (C)-СЛБ МАН2
1149 23	INX	H	
114A 56	MOV	D,M	; (D)-МЛБ МАН2
114B 1E00	MVI	E,0	
114D E1	POP	H	; (H,L)-АДРЕС СЛ1
	; СЛОЖЕНИЕ МАНТИСС: (M)+(B,C,D)=(B,C,D)		
114E 23	ПЕР5: INX	H	
114F 23	INX	H	
1150 23	INX	H	; АДРЕС МЛБ МАН1
1151 7E	MOV	A,M	
1152 82	ADD	D	
1153 57	MOV	D,A	; (D)-МЛБ МАН СУММЫ
1154 2B	DCX	H	
1155 7E	MOV	A,M	
1156 89	ADC	C	
1157 4F	MOV	C,A	; (C)-СЛБ МАН СУММЫ
1158 2B	DCX	H	
1159 7E	MOV	A,M	
115A 8B	ADC	B	



```

;ПРОВЕРКА ЗНАКОВ СЛАГАЕМЫХ НА СОВПАДЕНИЕ
115B F5      PUSH    PSW
115C D5      PUSH    D
115D 5F      MOV     E,A      ;(E)-СОХРАНЕНИЕ СТВ СУММЫ
115E 78      MOV     A,B      ;(A)-СТВ СЛ2
115F 43      MOV     B,E      ;(B)-СТВ МАИ СУММЫ
1160 AE      XRA     M        ;(M)-СТВ СЛ1
1161 FA8111  JM      PER6     ;ЕСЛИ ЗНАКИ СЛАГАЕМЫХ РАЗНЫЕ

;ПРОВЕРКА ПЕРЕПОЛНЕНИЯ СУММЫ
1164 7E      MOV     A,M      ;(A)-СТВ СЛ1
1165 A8      XRA     B        ;(B)-СТВ СУММЫ
1166 FA7611  JM      PER7     ;ЕСЛИ ПЕРЕПОЛНЕНИЕ

;ПРОВЕРКА ОСОБОГО СЛУЧАЯ ПЕРЕПОЛНЕНИЯ
1169 7E      MOV     A,M
116A A0      ANA     B
116B F28111  JP      PER6     ;ЕСЛИ СУММА ПОЛОЖИТЕЛЬНА
116E 78      MOV     A,B
116F E67F    ANI     7FH      ;ИСКЛЮЧЕНИЕ ЗНАКА
1171 B1      ORA     C
1172 B2      ORA     D
1173 C28111  JNZ     PER6     ;ЕСЛИ МАНТИССА НЕ 0

;ПЕРЕПОЛНЕНИЕ:НОРМАЛИЗАЦИЯ МАНТИССЫ ВПРАВО
PER7: POP     D
      POP     PSW
1176 D1      CALL    ПМАНЗ     ;(B,C,D)-МАНТИССА
1177 F1      JNC     PER8     ;НЕТ ПЕРЕПОЛНЕНИЯ ПОРЯДКА
1178 CD0012  JMF     PER9     ;ПЕРЕПОЛНЕНИЕ ПОРЯДКА
117B D28C11  JMF     PER9
117E C3A011  ;НОРМАЛИЗАЦИЯ МАНТИССЫ ВЛЕВО

1181 D1      PER6: POP     D
1182 F1      POP     PSW
1183 CD0011  CALL    НМАНЗ     ;(B,C,D)-МАНТИССА
1186 D28C11  JNC     PER8     ;НЕТ АНТИПЕРЕПОЛНЕНИЯ ПОРЯДКА
1189 C3A011  JMF     PER9     ;АНТИПЕРЕПОЛНЕНИЕ ПОРЯДКА

;ОКРУГЛЕНИЕ РЕЗУЛЬТАТА,НОРМАЛИЗАЦИЯ МАНТИССЫ
118C AF      PER8: XRA     A
118D 83      ADD     E
118E F2A011  JP      PER9     ;ЕСЛИ КОРРЕКЦИЯ=0
1191 7A      MOV     A,D
1192 C601    ADI     01
1194 57      MOV     D,A
1195 79      MOV     A,C
1196 CE00    ACI     00
1198 4F      MOV     C,A

1199 78      MOV     A,B
119A CE00    ACI     00
119C 47      MOV     B,A
119D DC0012  CS     ПМАНЗ     ;(B,C,D)-НОРМАЛИЗОВАННАЯ МАНТИССА

;ЗАПИСЬ РЕЗУЛЬТАТА В ПАМЯТЬ
PER9: MOV     M,B
11A0 70      INX     H
11A1 23      MOV     M,C
11A2 71      INX     H
11A3 23      MOV     M,D
11A4 72      INX     H
11A5 E1      POP     H        ;ВОССТАНОВЛЕНИЕ АДРЕСА СЛ2
11A6 C1      POP     B        ;ВОССТАНОВЛЕНИЕ АДРЕСА СЛ1
11A7 C9      RET          ;CY=1,ЕСЛИ ОШИБКА ПОРЯДКА
0000      END

```

Структура ее аналогична структуре программы СДПЗЗ, за исключением того, что переполнение мантиссы суммы проводится без применения модифицированных кодов. Такое решение для микропроцессора более экономично, требует меньше затрат памяти. Программа обращается к вспомогательным подпрограммам контроля слагаемых на нуль КОМ4 и их перемещения ПМ4, приведенным выше, к подпрограммам обмена слагаемых ОБМ4, денормализации мантиссы ДМАНЗ, устранения переполнения мантиссы суммы ПМАНЗ и ее нормализации влево НМАНЗ:

```

00F2                                ORG      0F2H
                                ОБМ4:
                                ;*****
                                ;ПОДПРОГРАММА ОБМЕНА СОДЕРЖИМЫМ ДВУХ ЧЕТЫРЕХБАЙТНЫХ МАС-
                                ;СИВОВ, РАЗМЕШЕННЫХ В НЕПЕРЕСЕКАЮЩИХСЯ ОБЛАСТЯХ ПАМЯТИ.
                                ;ВХОДНЫЕ ПАРАМЕТРЫ: (D,E) - АДРЕС МАССИВА 1, (H,L) - АДРЕС
                                ;МАССИВА 2. ИСПОЛЗУЮТСЯ И СОХРАНЯЮТСЯ ВСЕ РЕГИСТРЫ, ГЛУ-
                                ;БИНА СТЕКА - 8.
                                ;ОЦЕНКА: ДЛИНА - 22 БАЙТА, ВРЕМЯ - 333 ТАКТА.
                                ;*****
                                ;СОХРАНЕНИЕ РЕГИСТРОВ
00F2 E5                PUSH    H
00F3 D5                PUSH    D
00F4 C5                PUSH    B
00F5 F5                PUSH    PSW
00F6 0E04             MVI      C, 4      ; СЧЕТЧИК ЦИКЛОВ
                                ;ОБМЕН ОЧЕРЕДНЫМИ БАЙТАМИ
00F8 46              ЦИКЛ: MOV     B, M
00F9 1A              LDAX     D
00FA 77              MOV     M, A
00FB 78              MOV     A, B
00FC 12              STAX     D
00FD 13              INX      D
00FE 23              INX      H
                                ;ПРОВЕРКА КОНЦА ЦИКЛА
00FF 0D              DCR      C
0100 C2F800          JNZ      ЦИКЛ      ;ЗАЦИКЛИВАНИЕ
                                ;ВОССТАНОВЛЕНИЕ РЕГИСТРОВ
0103 F1              POP      PSW
0104 C1              POP      B
0105 D1              POP      D
0106 E1              POP      H
0107 C9              RET
0000                END

11B0                                ORG      11B0H
                                ДМАНЗ:
                                ;*****
                                ;ПОДПРОГРАММА ДЕНОРМАЛИЗАЦИИ ТРЕХБАЙТНОЙ МАНТИССЫ ЧИСЛА
                                ;С ПЛАВАЮЩЕЙ ЗАПЯТОЙ.
                                ;ВХОДНЫЕ ПАРАМЕТРЫ: (A) - РАЗНОСТЬ ПОРЯДКОВ N, ОПРЕДЕЛЯЮЩАЯ
                                ;ВЕЛИЧИНУ ДЕНОРМАЛИЗАЦИИ (N<23), (H,L) - АДРЕС ЧИСЛА С ПЛА-
```



```

1209 7B      MOV      A,E
120A 1F      RAR
120B 5F      MOV      E,A
;КОНТРОЛЬ ПОРЯДКА НА ПЕРЕПОЛНЕНИЕ И ЕГО КОРРЕКЦИЯ
120C 2B      DCX      H      ;АДРЕС ПОРЯДКА
120D 7E      MOV      A,M
120E FEFF    CPI      OFFH    ;FF-МАКСИМУМ ПОРЯДКА
1210 37      STC
;CY=1
1211 CA1612  JZ      ПЕР      ;ЕСЛИ ПОРЯДОК МАКСИМАЛЕН
1214 34      INR      M
1215 AF      XRA      A      ;CY=0
1216 23      ПЕР:    INX      H
1217 C9      RET
0000        END

```

```

11D0        ORG      11D0H

```

НМАНЗ:

```

;*****
;ПОДПРОГРАММА НОРМАЛИЗАЦИИ ТРЕХБАЙТНОЙ МАНТИССЫ ЧИСЛА
;С ПЛВАЮЩЕЙ ЗАПЯТОЙ.
;ВХОДНЫЕ ПАРАМЕТРЫ: (B,C,D)-НОРМАЛИЗУЕМАЯ МАНТИССА, (E)-
;МЛАДШИЕ РАЗРЯДЫ ДЕНОРМАЛИЗОВАННОЙ МАНТИССЫ, (H,L)-АДРЕС
;СТБ МАНТИССЫ В ПАМЯТИ. ВЫХОДНЫЕ ПАРАМЕТРЫ: (B,C,D)-НОРМА-
;ЛИЗОВАННАЯ МАНТИССА, (E)-ДОПОЛНИТЕЛЬНЫЕ МЛАДШИЕ РАЗРЯДЫ
;МАНТИССЫ, (H,L)-АДРЕС СТБ МАНТИССЫ В ПАМЯТИ, CY=1-ПРИЗНАК
;АНТИПЕРЕПОЛНЕНИЯ ПОРЯДКА. ИСПОЛЗУЮТСЯ ВСЕ РЕГИСТРЫ, СО-
;ХРАНЯЕТСЯ (H,L).
;ОЦЕНКА: ДЛИНА-46 БАЙТ, ВРЕМЯ-НЕ БОЛЕЕ (77+141(N-1)) ТАКТОВ
;N < 23.
;*****
;ПРОВЕРКА МАНТИССЫ НА НОЛЬ

```

```

11D0 7B      MOV      A,B      ; (A)-СТБ МАНТИССЫ
11D1 E67F    ANI      7FH      ;УДАЛЕНИЕ ЗНАКА
11D3 B1      ORA      C
11D4 B2      ORA      D
11D5 C2DD11  JNZ      ЦИКЛ      ;ЕСЛИ НЕ НОЛЬ
11D8 2B      DCX      H
11D9 77      MOV      M,A      ;ОБНУЛЕНИЕ ПОРЯДКА
11DA 23      INX      H
11DB 47      MOV      B,A      ;ОБНУЛЕНИЕ СТБ МАНТИССЫ
11DC C9      RET

```

;ПРОВЕРКА ЯВНО НОРМАЛИЗОВАННОЙ МАНТИССЫ: ЗНАКОВЫЙ И  
;СТАРШИЙ ЗНАЧАЩИЙ РАЗРЯД РАЗЛИЧНЫ ?

```

11DD 7B      ЦИКЛ:    MOV      A,B
11DE E6C0    ANI      0C0H      ;ВЫДЕЛЕНИЕ РАЗРЯДОВ
11E0 E0      RPO          ;ЕСЛИ. ЯВНАЯ НОРМАЛИЗАЦИЯ, (CY=0)

```

;ПРОВЕРКА НЕЯВНО НОРМАЛИЗОВАННОЙ МАНТИССЫ

```

11E1 7B      MOV      A,B
11E2 E63F    ANI      3FH      ;ИСКЛЮЧЕНИЕ СТАРШИХ РАЗРЯДОВ
11E4 B1      ORA      C
11E5 B2      ORA      D
11E6 C8      RZ          ;ЕСЛИ НЕЯВНАЯ НОРМАЛИЗАЦИЯ, (CY=0)
;НОРМАЛИЗАЦИЯ МАНТИССЫ: СДВИГ ВЛЕВО В (B,C,D,E)

```

```

11E7 EB      XCHG
11E8 29      DAD      H
11E9 EB      XCHG
11EA 79      MOV      A,C

```

11EB 17	RAL		
11EC 4F	MOV	C, A	
11ED 78	MOV	A, B	
11EE 17	RAL		
11EF 47	MOV	B, A	
; КОНТРОЛЬ НА АНТИПЕРЕПОЛНЕНИЕ ПОРЯДКА, КОРРЕКЦИЯ ПОРЯДКА			
11F0 2B	DCX	H	
11F1 7E	MOV	A, M	
11F2 B7	ORA	A	
11F3 37	STC		; CY=1
11F4 AFC11	JZ	ПЕР	; ЕСЛИ ПОРЯДОК МИНИМАЛЕН
11F7 35	DCR	M	
11F8 23	INX	H	
11F9 C3DD11	JMP	ЦИКЛ	; ЗАЦИКЛИВАНИЕ
11FC 23	PER: INX	H	
11FD C9	RET		; CY=1-АНТИПЕРЕПОЛНЕНИЕ ПОРЯДКА
0000	END		

Структура этих программ аналогична структуре программ, рассмотренных выше.

Набор положительных и отрицательных чисел, представленных в формате (8, 24), приведен в табл. 2.4.

Табл. 2.4. Числа с плавающей запятой в формате (8, 24)

Положительные числа		Отрицательные числа	
$A_{16}$	$A_{10}$	$A_{10}$	$A_{16}$
80400000	0,5	—0,5	80C00000
81400000	1	—1	81C00000
81600000	1,5	—1,5	81A00000
82400000	2	—2	82C00000
82500000	2,5	—2,5	82B00000
82600000	3	—3	82A00000
82700000	3,5	—3,5	82900000
83400000	4	—4	83C00000
83480000	4,5	—4,5	83B80000
83500000	5	—5	83B00000

## 2.3. УМНОЖЕНИЕ ДВОИЧНЫХ ЧИСЕЛ

### 2.3.1. МЕТОДИКА УМНОЖЕНИЯ

Умножение  $z = x \cdot y$  двоичных нормализованных чисел с плавающей запятой  $x = 2^{m_x} \cdot X$ ,  $y = 2^{m_y} \cdot Y$  определяется выражениями:

$$z = 2^{m_x} \cdot X \cdot 2^{m_y} \cdot Y = 2^{m'_z} \cdot Z' = 2^{m_z} \cdot Z; \quad (2.15)$$

$$Z, m_z = \begin{cases} Z', m'_z, & \text{если } Z' \geq 2^{-1}; \\ Z' \cdot 2^1, m'_z - 1, & \text{если } Z' < 2^{-1}, \end{cases} \quad (2.16)$$

где  $Z' = X \cdot Y$ ;  $m'_z = m_x + m_y$ .

В соответствии с формулами (2.15), (2.16) умножение выполняется в три этапа: 1) определяется порядок произведения путем алгебраического сложения порядков сомножителей; 2) находится мантисса произведения путем перемножения мантисс сомножителей по правилам арифметики с фиксированной запятой; 3) производится, если необходимо, нормализация произведения влево. Поскольку минимальные значения нормализованных мантисс сомножителей равны  $2^{-1}$ , минимальное значение произведения составляет  $(2^{-1} \cdot 2^{-1}) = 2^{-2}$ , и нормализация требует не более одного сдвига произведения влево, что отражает формула (2.16). При алгебраическом суммировании порядков сомножителей и нормализации влево возможны *переполнение* и *антипереполнение порядка произведения*, что необходимо выявлять программными средствами. При вычислении порядка произведения в случае смещенных порядков сомножителей  $m_{x\text{ см}}$  и  $m_{y\text{ см}}$  сумма последних равна  $m_{x\text{ см}} + m_{y\text{ см}} = m_x + 2^k + m_y + 2^k = (m_x + m_y + 2^k) + 2^k$  и отличается от правильного смещенного порядка произведения на величину  $2^k$ . Поэтому вычисление смещенного порядка произведения выполняется по формуле  $m_{x\text{ см}} + m_{y\text{ см}} - 2^k = m'_z\text{ см}$ .

Граничная относительная ошибка произведения чисел с плавающей запятой определяется так же, как и для чисел с фиксированной запятой, по выражению (1.16) при условии отсутствия переполнения и антипереполнения порядка произведения.

Далее будут приведены программы умножения чисел с плавающей запятой УДПЗЗ и УДПЗ4, реализующие рассмотренный алгоритм умножения для короткого и длинного форматов чисел.

### 2.3.2. Формат $(8,16) \cdot (8,16) = (8,16)$

Программа УДПЗЗ выполняет умножение чисел в формате  $(8,16)$ :

1220		ORG	1220H
0090	КОМЗ	SET	90H
00A3	ОВНЗ	SET	0A3H
0790	УДФ17	SET	790H
10D0	НМАН2	SET	10D0H

УДПЗЗ:

```

;*****
;ПОДПРОГРАММА УМНОЖЕНИЯ 3-БАЙТНЫХ ДВОИЧНЫХ ЧИСЕЛ С ПЛА-
;ВАЮЩЕЙ ЗАПЯТОЙ В ДОПОЛНИТЕЛЬНОМ КОДЕ ФОРМАТА (8,16)=
; (ПОР,МАН),ГДЕ БАЙТ ПОРЯДКА СОДЕРЖИТ БИТ ЗНАКА МАНТИССЫ

```

;И ЦЕЛОЧИСЛЕННЫЙ ДВОИЧНЫЙ ПОРЯДОК СО СМЕЩЕНИЕМ +40H, A  
 ;ДВА БАЙТА МАНТИССЫ-СТБ И МЛБ-ДРОБНОЕ НОРМАЛИЗОВАННОЕ  
 ;ЧИСЛО В ДОПОЛНИТЕЛЬНОМ КОДЕ.  
 ;ВХОДНЫЕ ПАРАМЕТРЫ: (B,C)-АДРЕС МНОЖИМОГО, (H,L)-АДРЕС  
 ;МНОЖИТЕЛЯ. ВЫХОДНЫЕ ПАРАМЕТРЫ: (B,C)-АДРЕС ПРОИЗВЕДЕНИЯ  
 ;ПР (ПР НА МЕСТЕ МНОЖИМОГО), CY=1-ПРИЗНАК ПЕРЕПОЛНЕНИЯ  
 ;ИЛИ АНТИПЕРЕПОЛНЕНИЯ ПОРЯДКА ПР. ИСПОЛЗУЮТСЯ ВСЕ РЕ-  
 ;ГИСТРЫ, СОХРАНЯЮТСЯ (B,C), (H,L). ГЛУБИНА СТЕКА-14. ИСПОЛЬ-  
 ;ЗУЮТСЯ ПОДПРОГРАММЫ: \*КОМЗ\*, \*ОВНЗ\*, \*УДФ17\*, \*НМАН2\*.  
 ;ОЦЕНКА: ДЛИНА-61 БАЙТ (+167 БАЙТ ПОДПРОГРАММ), ВРЕМЯ-НЕ  
 ;БОЛЕЕ 2054 ТАКОВ (С УЧЕТОМ ПОДПРОГРАММ).  
 ;\*\*\*\*\*  
 ;ПРОВЕРКА МНОЖИМОГО ММ НА НОЛЬ

1220	50	MOV	D,B	
1221	59	MOV	E,C	; (D,E)-АДРЕС ММ
1222	EB	XCHG		
1223	CD9000	CALL	КОМЗ	; Z=1, ЕСЛИ ММ=0
1226	EB	XCHG		
1227	C8	RZ		; ЕСЛИ ММ=0, (CY=0)
				; ПРОВЕРКА МНОЖИТЕЛЯ МН НА НОЛЬ
1228	CD9000	CALL	КОМЗ	; Z=1, ЕСЛИ МН=0
1228	C23412	JNZ	ПЕР1	; ЕСЛИ МН НЕ 0
122E	EB	XCHG		
122F	CDA300	CALL	ОВНЗ	; ОБНУЛЕНИЕ ПРОИЗВЕДЕНИЯ
1232	EB	XCHG		
1233	C9	RET		; ЕСЛИ МН=0, (CY=0)
				; ОБА СОМНОЖИТЕЛЯ НЕ 0. ОПРЕДЕЛЕНИЕ ПОРЯДКА И ЗНАКА ПР
1234	C5	ПЕР1: PUSH	B	; СОХРАНЕНИЕ АДРЕСА ММ
1235	E5	PUSH	H	; СОХРАНЕНИЕ АДРЕСА МН
1236	1A	LDAX	D	; (A)-БАЙТ ПОР ММ
1237	AE	XRA	M	; (M)-БАЙТ ПОР МН
1238	E680	ANI	80H	; (A7)-ЗНАК ПРОИЗВЕДЕНИЯ
123A	4F	MOV	C,A	; СОХРАНЕНИЕ ЗНАКА В (C)
123B	1A	LDAX	D	
123C	E67F	ANI	7FH	; ИСКЛЮЧЕНИЕ ЗНАКА ММ
123E	47	MOV	B,A	; (B)-ПОР ММ
123F	7E	MOV	A,M	
1240	E67F	ANI	7FH	; ИСКЛЮЧЕНИЕ ЗНАКА МН
1242	D640	SUI	40H	; ИСКЛЮЧЕНИЕ СМЕЩЕНИЯ ДЛЯ МН
1244	80	ADD	B	; (A)=(ПОР ММ + ПОР МН)
1245	37	STC		; CY=1
1246	FA5A12	JM	ПЕР2	; ЕСЛИ ПЕРЕПОЛНЕНИЕ ПОРЯДКА
1249	B1	ORA	C	; ВСТАВКА ЗНАКА В БАЙТ ПОР
124A	F5	PUSH	PSW	; СОХРАНЕНИЕ БАЙТА ПОРЯДКА
				; ВЫЧИСЛЕНИЕ МАНТИССЫ ПРОИЗВЕДЕНИЯ И ЕЕ НОРМАЛИЗАЦИЯ
124B	CD9007	CALL	УДФ17	; (B,C)-МАНТИССА ПР
124E	F1	POP	PSW	; ВОССТАНОВЛЕНИЕ БАЙТА ПОРЯДКА
124F	12	STAX	D	; ЗАПОМИНАНИЕ ПОРЯДКА
1250	EB	XCHG		; (H,L)-АДРЕС ПОРЯДКА ПР
1251	1600	MVI	D,0	
1253	CD0010	CALL	НМАН2	; (B,C)-МАНТИССА
				; ЗАПИСЬ РЕЗУЛЬТАТА В ПАМЯТЬ
1256	23	INX	H	
1257	70	MOV	M,B	
1258	23	INX	H	

1259 71		MOV	M,C	
125A E1	PER2:	POP	H	
125B C1		POP	B	
125C C9		RET		;CY=1,ЕСЛИ ОШИБКА ПОРЯДКА
0000		END		

Программа осуществляет контроль множителей на нуль с помощью подпрограммы КОМЗ и обнуление произведения в случае нулевого сомножителя с помощью подпрограммы ОБНЗ:

00A3		ORG	0A3H
------	--	-----	------

ОБНЗ:

```

;*****
;ПОДПРОГРАММА ОБНУЛЕНИЯ 3-БАЙТНОГО МАССИВА В ЗАДАННОЙ
;ОБЛАСТИ ПАМЯТИ.
;ВХОДНОЙ ПАРАМЕТР: (H,L)-НАЧАЛЬНЫЙ АДРЕС ОБНУЛЯЕМОГО МАС-
;СИВА.ИСПОЛЬЗУЕТСЯ РЕГИСТР А,СОХРАНЯЕТСЯ (H,L).
;ОЦЕНКА:ДЛИНА-9 БАЙТ,ВРЕМЯ-55 ТАКТОВ.
;*****
00A3 AF      XRA      A      ; (A)=0
00A4 77      MOV      M,A
00A5 23      INX      H
00A6 77      MOV      M,A
00A7 23      INX      H
00A8 77      MOV      M,A
00A9 2B      DCX      H
00AA 2B      DCX      H
00AB C9      RET

```

ОБН4:

```

;*****
;ПОДПРОГРАММА ОБНУЛЕНИЯ 4-БАЙТНОГО МАССИВА В ЗАДАННОЙ
;ОБЛАСТИ ПАМЯТИ.
;ВХОДНОЙ ПАРАМЕТР: (H,L)-НАЧАЛЬНЫЙ АДРЕС ОБНУЛЯЕМОГО МАС-
;СИВА.ИСПОЛЬЗУЕТСЯ РЕГИСТР А,СОХРАНЯЕТСЯ (H,L).
;ОЦЕНКА:ДЛИНА-12 БАЙТ,ВРЕМЯ-72 ТАКТА.
;*****
00AC AF      XRA      A      ; (A)=0
00AD 77      MOV      M,A
00AE 23      INX      H
00AF 77      MOV      M,A
00B0 23      INX      H
00B1 77      MOV      M,A
00B2 23      INX      H
00B3 77      MOV      M,A
00B4 2B      DCX      H
00B5 2B      DCX      H
00B6 2B      DCX      H
00B7 C9      RET
0000      END

```

Аналогичная программа ОБН4 используется в программе УДПЗ4 для обнуления четырехбайтного произведения. Применение программ КОМЗ и ОБНЗ сокращает среднее время выполнения программы УДПЗЗ, когда в по-



токе сомножителей часто встречаются нулевые числа. При ненулевых сомножителях программа вычисляет порядок и мантиссу произведения и, если необходимо, выполняет нормализацию числа с помощью подпрограммы НМАН2, описанной выше. Заканчивается умножение записью результата в память. При переполнении или антипереполнении порядка произведения устанавливается признак переноса  $CY=1$ , сигнализирующий о некорректности вычислений. При вычислении мантиссы произведения программа обращается к подпрограмме умножения УДФ17. Эта программа использует симметричный способ округления произведения, что определяет и соответствующую ему относительную ошибку.

Для тестирования программы УДПЗ3 можно использовать данные табл. 2.3.

### 2.3.3. ФОРМАТ $(8,24) \cdot (8,24) = (8,24)$

Программа УДПЗ4 выполняет умножение чисел в формате  $(8,24)$ :

1270		ORG	1270H
0098	КОМ4	SET	98H
00AC	ОБН4	SET	0ACH
0810	УДФ24	SET	810H
11D0	НМАНЗ	SET	11D0H

#### УДПЗ4:

```

;*****
;ПОДПРОГРАММА УМНОЖЕНИЯ 4-БАЙТНЫХ ДВОИЧНЫХ ЧИСЕЛ С ПЛА-
;ВЯЮЩЕЙ ЗАПЯТОЙ В ДОПОЛНИТЕЛЬНОМ КОДЕ ФОРМАТА (8,24)=
;=(ПОР,МАН),ГДЕ БАЙТ ПОРЯДКА СОДЕРЖИТ ЦЕЛОЧИСЛЕННЫЙ ДВО-
;ИЧНЫЙ ПОРЯДОК СО СМЕЩЕНИЕМ +80H,А 3 БАЙТА МАНТИССЫ-СТБ,
;СРБ,МЛБ-БИТ ЗНАКА МАНТИССЫ И ДВОИЧНОЕ ДРОБНОЕ НОРМАЛИ-
;ЗОВАННОЕ ЧИСЛО В ДОПОЛНИТЕЛЬНОМ КОДЕ.
;ВХОДНЫЕ ПАРАМЕТРЫ: (В,С)-АДРЕС МНОЖИМОГО, (Н,Л)-АДРЕС МНО-
;ЖИТЕЛЯ.ВЫХОДНЫЕ ПАРАМЕТРЫ: (В,С)-АДРЕС ПРОИЗВЕДЕНИЯ ПР
;(ПР НА МЕСТЕ МНОЖИМОГО),CY=1-ПРИЗНАК ПЕРЕПОЛНЕНИЯ ИЛИ
;АНТИПЕРЕПОЛНЕНИЯ ПОРЯДКА ПР.ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ,
;СОХРАНЯЮТСЯ (В,С), (Н,Л).ГЛУБИНА СТЕКА-XX.ИСПОЛЬЗУЮТСЯ
;ПОДПРОГРАММЫ: *КОМ4*, *ОБН4*, *УДФ24*, *НМАНЗ*.
;ОЦЕНКА: ДЛИНА- 52 БАЙТ (+245 БАЙТ ПОДПРОГРАММ),ВРЕМЯ-
;НЕ БОЛЕЕ 3277 ТАКОВ (С УЧЕТОМ ПОДПРОГРАММ).
;*****
;ПРОВЕРКА МНОЖИМОГО МН НА НОЛЬ

```

1270 50	MOV	D,B	
1271 59	MOV	E,C	; (D,E)-АДРЕС МН
1272 E8	XCHG		
1273 CD9800	CALL	КОМ4	;Z=1,ЕСЛИ МН=0
1276 E8	XCHG		
1277 C8	RZ		;ЕСЛИ МН=0,(CY=0)
			;ПРОВЕРКА МНОЖИТЕЛЯ МН НА НОЛЬ
1278 CD9800	CALL	КОМ4	;Z=1,ЕСЛИ МН=0

127B C28412	JNZ	ПЕР1	;ЕСЛИ МН НЕ 0
127E EB	XCHG		
127F CDAC00	CALL	ОБН4	;ОБНУЛЕНИЕ ПРОИЗВЕДЕНИЯ
1282 EB	XCHG		
1283 C9	RET		;ЕСЛИ МН=0, (CY=0)
;ОБА СМНОЖИТЕЛЯ НЕ 0.ОПРЕДЕЛЕНИЕ ПОРЯДКА ПРОИЗВЕДЕНИЯ			
1284 C5	ПЕР1: PUSH	B	;СОХРАНЕНИЕ АДРЕСА МН
1285 E5	PUSH	H	;СОХРАНЕНИЕ АДРЕСА МН
1286 1A	LDAX	D	; (A) - ПОР МН
1287 D680	SUI	80H	;ИСКЛЮЧЕНИЕ СМЕЩЕНИЯ
1289 86	ADD	M	; (A) = (ПОР МН + ПОР МН)
128A DAA112	JC	ПЕР2	;ЕСЛИ ПЕРЕПОЛНЕНИЕ ПОРЯДКА
128D 12	STAX	D	;ЗАПОМИНАНИЕ ПОРЯДКА
;ВЫЧИСЛЕНИЕ МАНТИССЫ ПРОИЗВЕДЕНИЯ, ЕЕ НОРМАЛИЗАЦИЯ			
128E 13	INX	D	
128F 23	INX	H	
1290 CD100B	CALL	УДФ24	; (A, B, C) - МАНТИССА ПР
1293 EB	XCHG		; (H, L) - АДРЕС ПОРЯДКА ПР
1294 51	MOV	D, C	
1295 48	MOV	C, B	
1296 47	MOV	B, A	; (B, C, D) - МАНТИССА ПР
1297 1E00	MVI	E, 0	
1299 CDD011	CALL	НМАНЗ	; (B, C, D) - МАНТИССА ПР
;ЗАПИСЬ МАНТИССЫ В ПАМЯТЬ			
129C 70	MOV	M, B	
129D 23	INX	H	
129E 71	MOV	M, C	
129F 23	INX	H	
12A0 72	MOV	M, D	
12A1 E1	ПЕР2: POP	H	;ВОССТАНОВЛЕНИЕ АДРЕСА МН
12A2 C1	POP	B	;ВОССТАНОВЛЕНИЕ АДРЕСА МН
12A3 C9	RET		;CY=1, ЕСЛИ ОШИБКА ПОРЯДКА
0000	END		

Структура этой программы аналогична структуре УДПЗЗ. Программа обращается к рассмотренным выше подпрограммам КОМ4 и ОБН4 и программам НМАНЗ и УДФ24. Для тестирования можно использовать данные табл. 2.4.

## 2.4. ДЕЛЕНИЕ ДВОИЧНЫХ ЧИСЕЛ

Деление  $z:y = x$  двоичных нормализованных чисел с плавающей запятой  $z = 2^{m_z} \cdot Z$ ,  $y = 2^{m_y} \cdot Y$  определяется выражениями:

$$x = 2^{m_z} \cdot Z : 2^{m_y} \cdot Y = 2^{m'_x} \cdot X' = 2^{m_x} \cdot X; \quad (2.17)$$

$$X, m_x = \begin{cases} X', m'_x, & \text{если } X' < 1; \\ X' \cdot 2^{-1}, m'_x + 1, & \text{если } X' \geq 1, \end{cases} \quad (2.18)$$

где  $m'_x = m_z - m_y$ ;  $X' = Z/Y$ .

В соответствии с формулами (2.17), (2.18) деление выполняется в три этапа: 1) определяется порядок частного путем вычитания порядка делителя из порядка делимого; 2) находится мантисса частного путем деления мантиссы делимого на мантиссу делителя по правилам арифметики с фиксированной запятой; 3) производится, если необходимо, нормализация частного вправо. Диапазон мантиссы частного имеет вид

$$2^{-1} < \frac{2^{-1}}{1-2^{-n}} = \frac{|Z|_{\min}}{|Y|_{\max}} \leq |X| \leq \frac{|Z|_{\max}}{|Y|_{\min}} = \frac{1-2^{-n}}{2^{-1}} < 2.$$

Следовательно, при делении возможны *переполнение мантиссы частного* (но не потеря значности) и ее нормализация вправо в соответствии с формулой (2.18).

Как и в случае умножения чисел, при делении возможны *переполнение* и *антипереполнение порядка частного*, которые необходимо выявлять. Кроме того, следует фиксировать условие деления на нулевой делитель. Вычисление смещенного порядка частного производится по формуле  $m_{z\text{ см}} - m_{y\text{ см}} + 2^k = m'_{x\text{ см}}$ .

Граничная относительная ошибка деления чисел с плавающей запятой определяется, как и для чисел с фиксированной запятой, выражением (1.18) при условии отсутствия переполнения и антипереполнения порядка частного.

Программа ДДПЗЗ выполняет деление чисел с плавающей запятой в формате (8,16):

12C0		ORG	12C0H
0090	КОМЗ	SET	90H
0C20	ДДФ17	SET	0C20H
10FA	ПМА2	SET	10FAH

ДДПЗЗ:

```
*****
;ПОДПРОГРАММА ДЕЛЕНИЯ 3-БАЙТНЫХ ДВОИЧНЫХ ЧИСЕЛ С ПЛА-
;ВАЮЩЕЙ ЗАПЯТОЙ В ДОПОЛНИТЕЛЬНОМ КОДЕ ФОРМАТА (8,16)=
;=(ПОР,МАН),ГДЕ БАЙТ ПОРЯДКА СОДЕРЖИТ БИТ ЗНАКА МАН-
;ТИССЫ И ЦЕЛОЧИСЛЕННЫЙ ДВОИЧНЫЙ ПОРЯДОК СО СМЕЩЕНИЕМ
;+40Н,А 2 БАЙТА МАНТИССЫ-СТБ И МЛБ-ДРОБНОЕ НОРМАЛИЗО-
;ВАННОЕ ЧИСЛО В ДОПОЛНИТЕЛЬНОМ КОДЕ.
;ВХОДНЫЕ ПАРАМЕТРЫ:(В,С)-АДРЕС ДЕЛИМОГО,(Н,Л)-АДРЕС
;ДЕЛИТЕЛЯ.ВЫХОДНЫЕ ПАРАМЕТРЫ:(В,С)-АДРЕС ЧАСТНОГО
;ЧАСТНОЕ НА МЕСТЕ ДЕЛИМОГО,СУ=1-ПРИЗНАК ПЕРЕПОЛНЕНИЯ
;ИЛИ АНТИПЕРЕПОЛНЕНИЯ ПОРЯДКА ЧАСТНОГО,ПРИЗНАК НУЛЕ-
;ВОГО ДЕЛИТЕЛЯ.ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ,СОХРАНЯЮТСЯ
;(В,С),(Н,Л).ГЛУБИНА СТЕКА-8.ИСПОЛЬЗУЮТСЯ ПОДПРОГРАММЫ:
;*КОМЗ*,*ДДФ17*,*ПМА2*.
```

```

;ОЦЕНКА:ДЛИНА=59 БАЙТ (+119 БАЙТ ПОДПРОГРАММ),ВРЕМЯ=НЕ
;БОЛЕЕ 2780 ТАКОВ (С УЧЕТОМ ПОДПРОГРАММ).
;*****
;ПРОВЕРКА ДЕЛИТЕЛЯ ДЛ НА НОЛЬ
12C0 CD9000      CALL    КОМЗ      ;Z=1,ЕСЛИ ДЛ=0
12C3 37          STC          ;CY=1
12C4 C8          RZ          ;ЕСЛИ ДЕЛИТЕЛЬ=0,CY=1
;ПРОВЕРКА ДЕЛИМОГО ДМ НА НОЛЬ
12C5 EB          XCHG
12C6 CD9000      CALL    КОМЗ      ;Z=1,ЕСЛИ ДМ=0
12C9 EB          XCHG
12CA C8          RZ          ;ЕСЛИ ДЕЛИМОЕ=0,CY=0
;ДЛ И ДМ НЕ НОЛЬ.ОПРЕДЕЛЕНИЕ ПОРЯДКА И ЗНАКА ЧАСТНОГО
12CB 50          MOV        D,B
12CC 59          MOV        E,C      ;(D,E)-АДРЕС ДМ
12CD C5          PUSH       B      ;СОХРАНЕНИЕ АДРЕСА ДМ
12CE E5          PUSH       H      ;СОХРАНЕНИЕ АДРЕСА ДЛ
12CF 1A          LDAX       D      ;(A)-БАЙТ ПОРЯДКА ДМ
12D0 AE          XRA        M      ;(M)-БАЙТ ПОРЯДКА ДЛ
12D1 E680        ANI        80H    ;(A7)-ЗНАК ЧАСТНОГО
12D3 4F          MOV        C,A      ;СОХРАНЕНИЕ ЗНАКА В (C)
12D4 7E          MOV        A,M
12D5 E67F        ANI        7FH    ;ИСКЛЮЧЕНИЕ ЗНАКА ДЛ
12D7 47          MOV        B,A      ;(B)-ПОРЯДОК ДЛ
12D8 1A          LDAX       D
12D9 E67F        ANI        7FH    ;ИСКЛЮЧЕНИЕ ЗНАКА ДМ
12DB C640        ADI        40H    ;УЧЕТ СМЕШЕНИЯ ПОРЯДКА
12DD 90          SUB        B      ;(A)-ПОРЯДОК ЧАСТНОГО
12DE 37          STC          ;CY=1
12DF FAF912      JM         PER1    ;ЕСЛИ ПЕРЕПОЛНЕНИЕ ПОРЯДКА
12E2 B1          ORA        C      ;ВСТАВКА ЗНАКА В БАЙТ ПОР
12E3 47          MOV        B,A
12E4 C5          PUSH       B      ;СОХРАНЕНИЕ БАЙТА ПОР
;ВЫЧИСЛЕНИЕ МАНТИССЫ ЧАСТНОГО
12E5 CD200C      CALL    ДД017    ;(B,C)-ЧАСТНОЕ
12E8 E1          POP        H      ;ВОССТАНОВЛЕНИЕ БАЙТА ПОР
12E9 7C          MOV        A,H
12EA 12          STAX       D      ;ЗАПОМИНАНИЕ БАЙТА ПОР
12EB EB          XCHG         ;(H,L)-АДРЕС ДМ
12EC D2F512      JNC        PER2    ;ЕСЛИ НЕТ ПЕРЕПОЛНЕНИЯ
;КОРРЕКЦИЯ ПОРЯДКА ЧАСТНОГО
12EF CDF010      CALL    ПМА2      ;CY=1,ЕСЛИ ПЕРЕПОЛНЕНИЕ ПОРЯДКА
12F2 DAF912      JC         PER1    ;ЕСЛИ ПЕРЕПОЛНЕНИЕ
;ЗАПИСЬ МАНТИССЫ ЧАСТНОГО В ПАМЯТЬ
12F5 23          PER2: INX        H
12F6 70          MOV        M,B

12F7 23          INX        H
12F8 71          MOV        M,C
12F9 E1          PER1: POP        H      ;ВОССТАНОВЛЕНИЕ АДРЕСА ДЛ
12FA C1          POP        B      ;ВОССТАНОВЛЕНИЕ АДРЕСА ДМ
12FB C9          RET          ;CY=1,ЕСЛИ ОШИБКА ПОРЯДКА
0000            END

```

Программа использует обращение к подпрограмме КОМЗ (см. п. 2.2.1) для проверки делимого и делителя

на нуль. Далее при ненулевых числах программа определяет порядок и знак частного и возможность переполнения порядка. Если переполнения порядка нет, программа с помощью подпрограммы деления ДДФ17 (см. п. 1.5.3) находит мантиссу частного, устраняет, если необходимо, ее переполнение с помощью подпрограммы ПМА2 (дополнительный вход в программу ПМАН2, см. п. 2.2.1) и записывает результат в память на место делимого. В случае переполнения, антипереполнения порядка и деления на нуль устанавливается признак переноса  $CU=1$ , который для программы более высокого уровня является сигналом некорректности деления. Для тестирования программы ДДПЗ3 можно использовать данные табл. 2.3.

## 2.5. УМНОЖЕНИЕ ЦЕЛОГО ЧИСЛА НА ЧИСЛО С ПЛАВАЮЩЕЙ ЗАПЯТОЙ

Применение арифметики с плавающей запятой в микропроцессорных информационно-измерительных системах требует расширения стандартного набора арифметических действий операцией умножения числа с плавающей запятой на  $N$ -байтное целое число. Дело в том, что в таких системах многие измерительные величины накапливаются в целочисленном виде, например в виде количества импульсов, поступающих на вход системы от объекта измерения. Обычно каждый импульс несет определенную информацию об объекте, например сообщает о некотором кванте энергии, выработанной или потребленной объектом измерения, т. е. обладает некоторым заранее известным числовым весом. Умножая весовой коэффициент, представляемый в форме с плавающей запятой, на количество импульсов, поступивших в систему за определенное время, можно определить интегральные характеристики объекта измерения.

Программа УЦПЗ выполняет умножение трехбайтного числа с плавающей запятой на  $N$ -байтное ( $N < 256$ ) целое число, размещенное в памяти в порядке возрастания адресов от МЛБ к СТБ этого числа:

1310		ORG	1310H
0058	ДОПА	SET	58H
0050	ДОПВ	SET	50H
0078	ПОСН	SET	78H
0090	КОМЗ	SET	90H
00A3	ОВНЗ	SET	0A3H

УЦПЗ:

```

;*****
;ПОДПРОГРАММА УМНОЖЕНИЯ ЦЕЛОГО БЕЗЗНАКОВОГО ДВОИЧНОГО
;ЧИСЛА ФОРМАТА 8*N НА 3-БАЙТНОЕ ДВОИЧНОЕ ЧИСЛО С ПЛА-
;ВАЮЩЕЙ ЗАПЯТОЙ В ДОПОЛНИТЕЛЬНОМ КОДЕ ФОРМАТА (8,16)=
;(ПОР.МАН),ГДЕ БАЙТ ПОРЯДКА СОДЕРЖИТ БИТ ЗНАКА МАНТИССЫ
;И ЦЕЛОЧИСЛЕННЫЙ ДВОИЧНЫЙ ПОРЯДОК СО СМЕЩЕНИЕМ +40Н,А 2
;БАЙТА МАНТИССЫ-СТБ,МЛБ-ДРОБНОЕ НОРМАЛИЗОВАННОЕ ЧИСЛО.
;ВХОДНЫЕ ПАРАМЕТРЫ:(D,E)-АДРЕС СОМНОЖИТЕЛЯ СМ1 С ПЛАВА-
;ЮЩЕЙ ЗАПЯТОЙ,(H,L)-АДРЕС ЦЕЛОЧИСЛЕННОГО СОМНОЖИТЕЛЯ,(C)
;КОЛИЧЕСТВО N БАЙТ ЦЕЛОГО ЧИСЛА.ВЫХОДНЫЕ ПАРАМЕТРЫ:ПРОИЗ-
;ВЕДЕНИЕ-3-БАЙТНОЕ ЧИСЛО С ПЛАВАЮЩЕЙ ЗАПЯТОЙ В ОБЛАСТИ
;ПАМЯТИ "БУФЕР",CY=-ПРИЗНАК ПЕРЕПОЛНЕНИЯ ПОРЯДКА ПРОИЗВЕ-
;ДЕНИЯ.ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ,СОХРАНЯЮТСЯ (D,E),(H,L)
;ГЛУБИНА СТЕКА-8.ИСПОЛЬЗУЮТСЯ ПОДПРОГРАММЫ:*КОМЗ*,*ОВНЗ*
;*ДОПД*,*ДОПВ*,*ПОСН*.
;ОЦЕНКА:ДЛИНА-171 БАЙТ (+41 БАЙТ ПОДПРОГРАММ),ВРЕМЯ -НЕ
;БОЛЕЕ (712+1443*N) ТАКТОВ (С УЧЕТОМ ПОДПРОГРАММ)
;*****

```

```

1310 E5      PUSH    H      ;СОХРАНЕНИЕ АДРЕСА СМ2
1311 D5      PUSH    D      ;СОХРАНЕНИЕ АДРЕСА СМ1
;ПРОВЕРКА СОМНОЖИТЕЛЯ СМ1 НА НОЛЬ
1312 EB      XCHG
1313 CD9000   CALL    КОМЗ    ;Z=1,ЕСЛИ СМ1=0
1316 EB      XCHG
1317 C22313   JNZ     ПЕР1    ;ЕСЛИ СМ1 НЕ 0
;ОБНУЛЕНИЕ ПРОИЗВЕДЕНИЯ
131A 21BA13   ПЕР2: LXI     H,БУФЕР ; (H,L)-АДРЕС ПРОИЗВЕДЕНИЯ
131D CDA300   CALL    ОВНЗ
1320 D1      POP     D      ;ВОССТАНОВЛЕНИЕ АДРЕСА СМ1
1321 E1      POP     H      ;ВОССТАНОВЛЕНИЕ АДРЕСА СМ2
1322 C9      RET         ;ЕСЛИ СМ=0,(CY=0)
;ПРОВЕРКА СОМНОЖИТЕЛЯ СМ2 НА НОЛЬ,ОПРЕДЕЛЕНИЕ НОМЕРА N*
;СТАРШЕГО НЕНУЛЕВОГО БАЙТА СМ2
1323 AF      ПЕР1: XRA     A
1324 81      ADD     C
1325 CA1A13   JZ      ПЕР2    ;ЕСЛИ N=0
1328 E5      PUSH    H      ;СОХРАНЕНИЕ АДРЕСА СМ2
1329 0600    MVI     B,0
132B 09      DAD     B
132C 2B      DCX     H      ; (H,L)-АДРЕС СТБ СМ2
132D AF      XRA     A      ; (A)=0
132E B6      ЦИКЛ1: ORA     M      ;ПРОВЕРКА БАЙТА НА 0
132F C23B13   JNZ     ПЕР3    ;ЕСЛИ БАЙТ НЕ 0
1332 2B      DCX     H
1333 0D      DCR     C      ;УМЕНЬШЕНИЕ N*=N-1
1334 C22E13   JNZ     ЦИКЛ1   ;ЗАЦИКЛИВАНИЕ 1
1337 E1      POP     H      ;БАЛАНС СТЕКА
1338 C31A13   JMP     ПЕР2    ;СМ2=0
133B E1      ПЕР3: POP     H      ;ВОССТАНОВЛЕНИЕ АДРЕСА СМ2
;СОМНОЖИТЕЛИ НЕ 0.ЗАГРУЗКА МАНТИССЫ СМ1 В РЕГИСТРЫ
133C C5      PUSH    B      ;СОХРАНЕНИЕ N*
133D EB      XCHG      ; (H,L)-АДРЕС СМ1
133E 7E      MOV     A,M
133F E680    ANI     80H     ;ВЫДЕЛЕНИЕ ЗНАКА СМ1
1341 23      INX     H
1342 46      MOV     B,M
1343 23      INX     H
1344 4E      MOV     C,M

```

1345 EВ	XCHG		; (H,L) - АДРЕС СМ2
1346 50	MOV	D,B	
1347 59	MOV	E,C	; (D,E) - МАНТИССА
1348 FC5800	CM	ДОПД	; ДОПОЛНЕНИЕ, ЕСЛИ ЗНАК "-"
134B C1	POP	B	; ВОССТАНОВЛЕНИЕ N*
; ПОДГОТОВКА РЕГИСТРОВ К ЦИКЛУ УМНОЖЕНИЯ			
134C E5	PUSH	H	; СОХРАНЕНИЕ АДРЕСА СМ2
134D 21BA13	LXI	H, БУФЕР	
1350 3608	MVI	M, 8	; (БУФЕР) = 8 - СЧЕТ СДВИГОВ ЦИКЛА
1352 23	INX	H	
1353 AF	XRA	A	
1354 77	MOV	M, A	; (БУФЕР+1) = 0 - ОБЩИЙ СЧЕТ СДВИГОВ
1355 67	MOV	H, A	
1356 6F	MOV	L, A	; (H,L) = 0 - СУММА ЧП
1357 E3	XTHL		; (H,L) - ТЕКУЩИЙ АДРЕС СМ2
; ЦИКЛ УМНОЖЕНИЯ НА ТЕКУЩИЙ БАЙТ ЦЕЛОГО ЧИСЛА			
1358 46	MOV	B, M	; (B) - БАЙТ МНОЖИТЕЛЯ
1359 E3	XTHL		; (H,L) - ТЕКУЩАЯ СУММА ЧП
; СДВИГ МНОЖИТЕЛЯ ВПРАВО			
135A AF	XRA	A	; CY=0
135B 78	MOV	A, B	
135C 1F	RAR		
135D 47	MOV	B, A	
135E D26513	JNC	ПЕР4	; ЕСЛИ РАЗРЯД МНОЖИТЕЛЯ=0
; СЛОЖЕНИЕ МАНТИССЫ-МНОЖИМОГО С СУММОЙ ЧП			
1361 19	DAD	D	
1362 DA6E13	JC	ПЕР5	; ЕСЛИ ПЕРЕПОЛНЕНИЕ СУММЫ
; ПРОВЕРКА: ПОСЛЕДНИЙ НЕНУЛЕВОЙ БАЙТ МНОЖИТЕЛЯ ?			
1365 79	MOV	A, C	
1366 3D	DCR	A	
1367 C26E13	JNZ	ПЕР5	; ЕСЛИ N* > 1
136A 80	ADD	B	; (A) - ОСТАТОК МНОЖИТЕЛЯ ПРИ N* = 1
136B CA9013	JZ	ПЕР6	; ЕСЛИ ОСТАТОК = 0
; СДВИГ СУММЫ ЧП ВПРАВО, ПРОВЕРКА КОНЦА ЦИКЛА 3			
136E CD7800	CALL	ПОСЧ	; СДВИГ ВПРАВО (H,L)
1371 3ABA13	LDA	БУФЕР	; (A) - СЧЕТ СДВИГОВ ЦИКЛА
1374 3D	DCR	A	
1375 32BA13	STA	БУФЕР	; ЗАПОМИНАНИЕ СЧЕТЧИКА
1378 C25A13	JNZ	ЦИКЛ3	; ЗАЦИКЛИВАНИЕ 3
; ПОДГОТОВКА СЧЕТЧИКОВ К ПРОДОЛЖЕНИЮ ЦИКЛА 2			
137B E5	PUSH	H	
137C 21BA13	LXI	H, БУФЕР	
137F 3608	MVI	M, 8	; (БУФЕР) = 8
1381 23	INX	H	
1382 7E	MOV	A, M	
1383 C608	ADI	8	; УВЕЛИЧЕНИЕ ОБЩЕЙ СУММЫ СДВИГОВ
1385 77	MOV	M, A	
1386 E1	POP	H	
; ПРОВЕРКА КОНЦА ЦИКЛА УМНОЖЕНИЯ			
1387 0D	DCR	C	; УМЕНЬШЕНИЕ N*
1388 CA9013	JZ	ПЕР6	; ЕСЛИ N* = 0
138B E3	XTHL		; (H,L) - ТЕКУЩИЙ АДРЕС СМ2
138C 23	INX	H	; (H,L) - АДРЕС СЛЕДУЮЩЕГО БАЙТА
138D C35813	JMP	ЦИКЛ2	; ЗАЦИКЛИВАНИЕ 2
; ОКОНЧАНИЕ ЦИКЛА УМНОЖЕНИЯ			
1390 E5	PUSH	H	; СОХРАНЕНИЕ ПРОИЗВЕДЕНИЯ
1391 21BA13	LXI	H, БУФЕР	
1394 3E08	MVI	A, 8	
1396 96	SUB	M	; (A) - ПОСЛЕДНЕЕ ЧИСЛО СДВИГОВ

1397 23	INX	H	
1398 86	ADD	M	; (A) - СУММАРНОЕ ЧИСЛО СБИГОВ
1399 2B	DCX	H	
139A 77	MOV	M, A	; (БУФЕР) - СУММАРНОЕ ЧИСЛО СБИГОВ
	; ДОПОЛНЕНИЕ ОТРИЦАТЕЛЬНОГО ПРОИЗВЕДЕНИЯ		
139B C1	POP	B	; (B, C) - ПРОИЗВЕДЕНИЕ
139C D1	POP	D	; БАЛАНС СТЕКА
139D D1	POP	D	; ВОССТАНОВЛЕНИЕ АДРЕСА СМ1
139E 1A	LDAX	D	; (A) - БАЙТ ПОРЯДКА СМ1
139F D5	PUSH	D	; СОХРАНЕНИЕ АДРЕСА СМ1
13A0 5F	MOV	E, A	; (E) - БАЙТ ПОРЯДКА СМ1
13A1 E680	ANI	80H	; ВЫДЕЛЕНИЕ ЗНАКА СМ1
13A3 57	MOV	D, A	; СОХРАНЕНИЕ ЗНАКА В (D)
13A4 FC5000	CM	ДОТВ	; ДОПОЛНЕНИЕ ПРОИЗВЕДЕНИЯ, ЕСЛИ "-"
13A7 7B	MOV	A, E	; (A) - БАЙТ ПОРЯДКА СМ1
13A8 E67F	ANI	7FH	; ИСКЛЮЧЕНИЕ ЗНАКА СМ1
13AA 86	ADD	M	; КОРРЕКЦИЯ ПОРЯДКА ПРОИЗВЕДЕНИЯ
13AB F5	PUSH	PSW	; СОХРАНЕНИЕ ПРИЗНАКОВ
13AC B2	ORA	D	; ВСТАВКА ЗНАКА ПРОИЗВЕДЕНИЯ
	; ЗАПИСЬ ПРОИЗВЕДЕНИЯ В БУФЕР		
13AD 77	MOV	M, A	; ЗАПИСЬ ПОРЯДКА ПРОИЗВЕДЕНИЯ
13AE 23	INX	H	
13AF 70	MOV	M, B	; ЗАПИСЬ СТБ ПРОИЗВЕДЕНИЯ
13B0 23	INX	H	
13B1 71	MOV	M, C	; ЗАПИСЬ МЛБ ПРОИЗВЕДЕНИЯ
13B2 F1	POP	PSW	; ВОССТАНОВЛЕНИЕ ПРИЗНАКОВ
13B3 D1	POP	D	; ВОССТАНОВЛЕНИЕ АДРЕСА СМ1
13B4 E1	POP	H	; ВОССТАНОВЛЕНИЕ АДРЕСА СМ2
13B5 D8	RC		; ЕСЛИ ПЕРЕПОЛНЕНИЕ ПОРЯДКА, CY=1
13B6 37	STC		; CY=1
13B7 F8	RM		; ЕСЛИ ПЕРЕПОЛНЕНИЕ ПОРЯДКА
13B8 AF	XRA	A	; CY=0
13B9 C9	RET		; ПЕРЕПОЛНЕНИЯ НЕТ
	; ОБЛАСТЬ ХРАНЕНИЯ РЕЗУЛЬТАТА		
13BA	БУФЕР: DS	3	; 3 БАЙТА
0000	END		

Результат размещается в области памяти БУФЕР. Программа обращается к ряду вспомогательных программ, рассмотренных ранее. Умножение непосредственно выполняется лишь после проверки обоих сомножителей на нуль, причем проверка *N*-байтного сомножителя осуществляется, начиная с его старших байтов. Тем самым в дальнейшем исключаются из процесса умножения старшие незначащие (нулевые) байты числа. Мантисса сомножителя с плавающей запятой загружается из памяти в регистры микропроцессора, преобразуется из дополнительного кода в прямой, и далее выполняются вложенные циклы умножения: внутренний ЦИКЛ3 и внешний ЦИКЛ2. Умножение текущего байта целого сомножителя на положительный сомножитель с плаваю-



щей запятой осуществляется по методике умножения чисел с фиксированной запятой согласно вычислительной схеме 1 (см. рис. 1.5, а) с коррекцией порядка произведения при каждом очередном переполнении СЧП. После окончания умножения на текущий байт происходит переход к следующему байту, и так до тех пор, пока не будет закончено умножение на все значащие байты целого сомножителя. После этого производятся, если необходимо, дополнение произведения и его запись в БУФЕР. В случае переполнения порядка произведения устанавливается признак переноса  $CY=1$ . Тестовые данные программы УЦПЗ приведены в табл. 2.5.

**Табл. 2.5. Тесты умножения для программы УЦПЗ**

Представление чисел	
шестнадцатеричное	десятичное
418000·FF=48FF00	1·255=255
418000·FFFF=50FFFF	1·65535=65535
418000·5555=4FAAAA	1·21845=21845
418000·FF00=50FF00	1·65280=65280
44A000·AAAA=53D554	10·43690=436900

## 3. ПРОГРАММЫ ПРЕОБРАЗОВАНИЯ ПРЕДСТАВЛЕНИЙ ЧИСЕЛ

### 3.1. ОБЩИЕ СВЕДЕНИЯ

Обработка числовых данных в современных микропроцессорных устройствах производится с минимальными затратами памяти и времени, т. е. наиболее экономичным образом, в случае представления этих данных в двоичной системе счисления. Однако ввод-вывод исходных данных и результатов необходимо осуществлять в привычном и удобном для пользователя десятичном виде, в связи с чем возникает проблема перевода чисел из одной системы счисления в другую. Эта проблема решается программным путем при определенных затратах памяти и времени на преобразования представлений чисел. Если оказывается, что эти затраты велики и, кроме того, процессы ввода-вывода существенно преобладают над вычислительными процессами, целесообразно использовать десятичную, точнее, смешанную двоично-десятичную систему счисления и в процессе обработки данных, снимая тем самым проблему перевода чисел и минимизируя общие затраты на ввод-вывод и вычисления. В общем же случае проблема перевода сохраняется и является типичной для микропроцессорных устройств.

В данной главе рассматриваются алгоритмы и программы преобразования различных форматов беззнаковых целых и дробных чисел с фиксированной запятой, а также чисел с плавающей запятой из  $R$ -ичной системы счисления в  $P$ -ичную (и наоборот), в частности преобразования между двоичной и двоично-десятичной системами. Изложение ограничено, за одним исключением, беззнаковыми числами, поскольку перевод любого числа со знаком сводится к переводу его абсолютной величины и присвоению соответствующего знака результату.

Рассмотрим методы перевода чисел с фиксированной запятой из  $R$ -ичной системы в  $P$ -ичную ( $R \rightarrow P$ ) и из  $P$ -ичной системы в  $R$ -ичную ( $P \rightarrow R$ ) [17, 47, 53, 64]. Для определенности примем, что  $R < P$ , в частности  $R = 2$ ,  $P = 10$ .

Отдельно представим случаи целых и дробных чисел, поскольку перевод любого числа можно свести к независимому переводу его целой и дробной частей.

Один из методов перевода  $R \rightarrow P$  целых и дробных чисел заключается в непосредственном вычислении  $P$ -ичного значения полинома  $R$ -ичного числа, в котором  $R$ -ичные цифры и основание  $10_R$  записываются  $P$ -ичными цифрами, а операции сложения и умножения выполняются по правилам  $P$ -ичной арифметики:

$$\left. \begin{aligned} A_R &= \sum_{i=0}^{n-1} a_{Ri} \cdot 10_R^i \rightarrow A_P = \sum_{i=0}^{n-1} a_{Pi} \cdot R_P^i; \\ A_{R\Phi} &= \sum_{i=1}^n a_{Ri} \cdot 10_R^{-i} \rightarrow A_{P\Phi} = \sum_{i=0}^n a_{Pi} \cdot R_P^{-i}, \end{aligned} \right\} \quad (3.1)$$

где  $a_{Pi}$  и  $R_P$  — соответственно эквивалентные цифры  $a_{Ri}$  и основание  $10_R$ , записанные в  $P$ -ичной системе. Например, перевод  $2 \rightarrow 10$  для чисел 1101 и 1100 имеет вид ( $10_2 = 2_{10}$ ):  $1101 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 8 + 4 + 1 = 13$  и  $1100 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 8 + 4 = 12$ . Заметим, что, поскольку  $R < P$ , цифры в  $R$ -ичной и  $P$ -ичной записях полинома (3.1) совпадают, что упрощает перевод. Данный метод в равной мере применим и для обратного перевода  $P \rightarrow R$ . Например, перевод  $10 \rightarrow 2$  для чисел 13 и 0,75 имеет вид ( $10_{10} = 1010_2$ ):  $13_{10} = 0001 \cdot (1010)^1 + 0011 = 1101_2$  и  $0,75 = 0111 \times (1010)^{-1} + 0101 \cdot (1010)^{-2} = (0,101100...) + (0,000011...) = 0,101111...$  Пример показывает, что перевод чисел в систему с меньшим основанием ( $R < P$ ) по методу (3.1) достаточно сложен, особенно для дробных чисел.

На практике метод (3.1) используется в основном для переводов целых чисел  $R \rightarrow P$ , причем программирование метода упрощается, если полином вычислять по схеме Горнера:

$$A_P = (...((0 + a_{n-1}) \cdot R + a_{n-2})R + ... + a_1)R + a_0. \quad (3.2)$$

В этом случае рассмотренный ранее пример перевода  $2 \rightarrow 10$  числа  $1101_2$  имеет вид  $A_{10} = (((0 + 1) \cdot 2 + 1) \times 2 + 0)2 + 1 = 13$ .

В основе второго метода перевода целых чисел  $P \rightarrow R$  лежит идея представления фактического значения  $P$ -ичного числа эквивалентным  $R$ -ичным полиномом:

$$A_P = a_{R(n-1)} \cdot R^{n-1} + ... + a_{R1} \cdot R^1 + a_{R0}, \quad (3.3)$$

где  $A_P$  — фактическое значение  $P$ -ичного числа  $a_{Ri}$  — подлежащие определению цифры эквивалентного  $R$ -ичного числа. При делении обеих частей этого равенства на основание  $R$  системы, в которую совершается перевод, получаем целочисленное частное  $A_P/R = a_{R(n-1)} \cdot R^{n-2} + \dots + a_{R1}$  и остаток  $a_{R0}$ , который является искомой младшей цифрой эквивалентного  $R$ -ичного числа  $A_R$ . Продолжая деление частного на основание  $R$ , можно найти очередную цифру  $a_{R1}$  числа  $A_R$  и т. д. Метод определяется рекуррентным выражением

$$a_{Ri} = A_{Pi} - A_{P(i+1)} \cdot R, \quad (3.4)$$

где  $A_{P(i+1)} = A_{Pi}/R$ ;  $A_{P0} = A_P$ ,  $i = 0, 1, \dots, n-1$ .

Процесс деления в формуле (3.4) заканчивается, как только  $A_{P(i+1)}$  станет равным нулю. Операции деления и вычитания выполняются по правилам исходной  $P$ -ичной арифметики, и в этой же системе формируются искомые коэффициенты  $a_{Ri}$ . Например, перевод  $10 \rightarrow 2$  числа  $13$  имеет вид:  $a_0 = 13 - (13/2) \cdot 2 = 1$ ;  $a_1 = 6 - (6/2) \cdot 2 = 0$ ;  $a_2 = 3 - (3/2) \cdot 2 = 1$ ;  $a_3 = 1 - (1/2) \cdot 2 = 1$  и, следовательно,  $13_{10} = 1101_2$ . Данный метод применим и для обратного перевода  $R \rightarrow P$ . Например, перевод  $2 \rightarrow 10$  для числа  $1101_2$  имеет вид:  $a_0 = 1101 - (1101/1010) \cdot 1010 = 1101 - 1 \cdot 1010 = 0011$ ;  $a_1 = 0001 - (0001/1010) \cdot 1010 = 0001 - 0 \cdot 1010 = 0001$ . Для окончательной записи числа необходимо коэффициенты  $a_0, a_1$  изобразить в десятичной системе:  $1101_2 = 13_{10}$ .

В основе второго метода перевода дробных чисел  $P \rightarrow R$  лежит та же идея представления  $P$ -ичного числа эквивалентным  $R$ -ичным полиномом, что и для целых чисел:

$$A_{P\phi} = a_{R1} \cdot R^{-1} + a_{R2} \cdot R^{-2} + \dots + a_{Rn} \cdot R^{-n},$$

где  $A_{P\phi}$  — значение дроби в  $P$ -ичной системе;  $a_{Ri}$  — коэффициенты  $R$ -ичного представления, подлежащие определению. Очевидно, что эти коэффициенты можно получить путем последовательного умножения на  $R$  сначала исходного числа  $A_{P\phi}$ , а затем дробной части очередного произведения, причем целая часть произведения соответствует очередному  $R$ -ичному коэффициенту, записанному в  $P$ -ичной системе. Метод определяется рекуррентными выражениями:

$$a_{Ri} = [A_{P(i-1)} \cdot R]; A_{Pi} = \{A_{P(i-1)} \cdot R\}; A_{P0} = A_{P\Phi}, i = 1, 2, \dots, n, \quad (3.5)$$

где скобки [ ] и { } обозначают соответственно целую и дробную части числа. Умножение выполняется по правилам  $P$ -ичной арифметики и заканчивается, как только  $A_{Pi}$  станет равным нулю или будет достигнута требуемая точность  $R$ -ичного изображения числа  $A_{P\Phi}$ , определяемая количеством разрядов  $n$ . Например, перевод  $10 \rightarrow 2$  дроби 0,75 имеет вид:  $a_1 = [0,75 \cdot 2] = [1,5] = 1$ ;  $a_2 = [0,5 \cdot 2] = 1$  и, следовательно,  $0,75_{10} = ,11_2$ . Данный метод, как и предыдущие, применим и для обратного перевода  $R \rightarrow P$ . Например, перевод  $2 \rightarrow 10$  дроби  $,11_2$  имеет вид:  $a_1 = [,11 \times \times 1010] = [111,1] = 0111$ ;  $a_2 = [,1 \cdot 1010] = [101,0] = 0101$ . Преобразуя двоичные значения  $a_1$  и  $a_2$  в эквивалентные десятичные, получим  $,11_2 = 0,75_{10}$ .

Заметим, что процесс перевода дробей в отличие от процесса перевода целых чисел, который всегда оканчивается через конечное число шагов, может быть бесконечным, т. е. представление  $P$ -ичной дроби в  $R$ -ичной системе (или наоборот) может иметь бесконечное количество цифр. Поэтому переводы дробей в общем случае выполнимы лишь приближенно. Число цифр в представлении  $R$ -ичного числа необходимо определять по условию соответствия его точности исходному  $P$ -ичному числу:  $P^{-n_P} = R^{-n_R}$ , откуда

$$n_R = n_P / \log_P R, \quad (3.6)$$

где  $n_P$ ,  $n_R$  — количество цифр в изображении  $P$ -ичного и  $R$ -ичного чисел. На практике используют целую часть выражения (3.6), увеличенную на единицу. Например, если точность исходного десятичного числа с четырьмя значащими цифрами равна  $\delta_r = 0,5 \cdot 1000^{-1} = 0,05\%$  (при симметричном способе округления), то его двоичное представление той же точности должно содержать  $n = \lceil 4 \cdot (\log_{10} 2)^{-1} \rceil + 1 = [13,3] + 1 = 14$  двоичных разрядов. Для уменьшения погрешности перевода целесообразно вычислить также цифру следующего, 15-го разряда, но не сохранять ее, а использовать для округления двоичного числа.

Остановимся на методах перевода чисел с плавающей запятой. Пусть  $A_R = R^{m_R} \cdot A_{R\Phi}$  и  $A_P = P^{m_P} \cdot A_{P\Phi}$  — эквивалентные числа с плавающей запятой, представленные соответственно в  $R$ -ичной и  $P$ -ичной системах счисления. Один из методов перевода  $P \rightarrow R$  заключается в пред-

варительном преобразовании числа с плавающей запятой  $A_P$  в число с фиксированной запятой, выделении целой и дробной частей этого числа, переводе их в соответствии с вышерассмотренными методами в  $R$ -ичную систему и переходе к числу с плавающей запятой  $A_P$ . Так, преобразование  $10 \rightarrow 2$  для числа  $A_{10} = 10^2 \cdot 0,1375$  имеет вид:  $A_{10} = 13,75$ ;  $13_{10} = 1101_2$ ;  $0,75_{10} = 0,11_2$ ;  $A_2 = 1101,11$  и, наконец,  $A_2 = 2^4 \cdot 0,110111$ . Данный метод применим лишь для значений порядков, соизмеримых с разрядностью мантииссы, в противном случае резко возрастают затраты памяти и времени на хранение и обработку нулей.

Общий метод перевода заключается в том, что при переводе  $P \rightarrow R$  числа  $A_P$  отдельно преобразуют мантииссу  $A_{Pф}$  и характеристику этого числа  $P^{m_P}$ , причем каждое из этих чисел в  $R$ -ичной системе формируется как число с плавающей запятой, а затем выполняются их умножение и нормализация с окончательным получением мантииссы  $A_{Rф}$  и характеристики  $R^{m_R}$ , т. е. значения числа  $A_R$ . Так, преобразование  $10 \rightarrow 2$  для числа  $A_{10} = 10^2 \cdot 0,1375$  имеет вид:

$$10^2_{10} = 2^7 \cdot 0,110010_2; \quad 0,1375_{10} = 0,00100011... = \\ = (0,100011...) \cdot 2^{-2}; \quad A_2 = 2^7 \cdot 0,110010 \cdot 2^{-2} \times \\ \times (0,100011...) = 2^5 \cdot (0,011010...) = 2^4 \cdot (0,11010...). \quad \text{Заме-} \\ \text{тим, что в данном примере для получения } A_2 \text{ с точностью} \\ \text{числа } A_{10} \text{ надо брать 14 разрядов двоичной мантииссы.} \\ \text{Подробности метода перевода поясняются далее при рас-} \\ \text{смотрении конкретных программ.}$$

### 3.2. ПРЕОБРАЗОВАНИЯ ЦЕЛЫХ ДЕСЯТИЧНЫХ ЧИСЕЛ В ДВОИЧНЫЕ

Программа П108 преобразует двухразрядное десятичное число  $A_{10} \in [00,99]$  в эквивалентное однобайтное двоичное число  $A_2 \in [00,63H]$  по методу (3.1):

0E00

ORG

0E00H

П108:

```

; *****
; ПОДПРОГРАММА ПРЕОБРАЗОВАНИЯ ДВОИЧНО-ДЕСЯТИЧНОГО (КОД
; 8421 ) ЦЕЛОГО БЕЗЗНАКОВОГО ЧИСЛА ФОРМАТА 2*4 В ДВОИЧ-
; НОЕ ЧИСЛО ФОРМАТА 8.
; ВХОДНОЙ ПАРАМЕТР: (А) - ДВОИЧНО-ДЕСЯТИЧНОЕ ЧИСЛО. ВЫХОДНОЙ
; ПАРАМЕТР: (А) - ЭКВИВАЛЕНТНОЕ ДВОИЧНОЕ ЧИСЛО. ИСПОЛЪЗУЮТСЯ
; РЕГИСТРЫ (В,С).
; ОЦЕНКА: ДЛИНА-14 БАЙТ, ВРЕМЯ-64 ТАКТА.
; *****
; ВЫДЕЛЕНИЕ СТАРШЕЙ СТЦ И МЛАДШЕЙ МЛЦ ЦИФР ЧИСЛА

```

0E00 47

MOV

B,A

0E01 E60F	ANI	0FH	;МАСКА НА МЛЦ
0E03 4F	MOV	C,A	; (C) -МЛЦ
0E04 78	MOV	A,B	
0E05 E6F0	ANI	0F0H	;МАСКА НА СЦ
0E07 0F	RRC		
0E08 47	MOV	B,A	; (B) -СЦ*8
	;ДВОИЧНОЕ СЛОЖЕНИЕ:СЦ*(8+2)+МЛЦ		
0E09 0F	RRC		
0E0A 0F	RRC		; (A) -СЦ*2
0E0B 80	ADD	B	; (A) -СЦ*(8+2)
0E0C 81	ADD	C	; (A) -СЦ*10+МЛЦ
0E0D C9	RET		
0000	END		

Программа распаковывает исходное двоично-десятичное число отдельно на его *младшую* (МЛЦ) и *старшую* (СЦ) *десятичные цифры*, а затем умножает СЦ на 10 и суммирует произведение с МЛЦ. После распаковки СЦ она имеет в рамках двоичного байта значение  $СЦ \times 2^4$ . Умножение этой цифры на 10 производится с помощью последовательности сдвигов и сложений в двоичной системе:  $СЦ \cdot 10 = СЦ \cdot (2^3 + 2^1)$ .

Программа П1016 преобразует четырехразрядное десятичное число  $A_{10} \in [0000,9999]$  в эквивалентное двухбайтное двоичное число  $A_2 \in [0000,270FH]$  по методам (3.1), (3.2):

0E10		ORG	0E10H
0E00	П108	SET	0E00H

П1016:

\*\*\*\*\*

;ПОДПРОГРАММА ПРЕОБРАЗОВАНИЯ ДВОИЧНО-ДЕСЯТИЧНОГО (КОД ;8421) БЕЗЗНАКОВОГО ЧИСЛА ФОРМАТА 4\*4 В ДВОИЧНОЕ ЧИСЛО ;ФОРМАТА 16.

;ВХОДНОЙ ПАРАМЕТР: (B,C) -ДВОИЧНО-ДЕСЯТИЧНОЕ ЧИСЛО (РЗР2 ;R1R0). ВЫХОДНОЙ ПАРАМЕТР: (H,L) -ЭКВИВАЛЕНТНОЕ ДВОИЧНОЕ ;ЧИСЛО. ИСПОЛЬЗУЮТСЯ РЕГИСТРЫ (D,E), (A). ГЛУБИНА СТЕКА-4.

;ИСПОЛЬЗУЕТСЯ ПОДПРОГРАММА \*П108\*.

;ОЦЕНКА: АЛИНА-40 БАЙТОВ (+14 БАЙТ ПОДПРОГРАММЫ), ВРЕМЯ- ;334 ТАКТОВ (С УЧЕТОМ ПОДПРОГРАММЫ).

\*\*\*\*\*

;ВЫЧИСЛЕНИЕ ВЫРАЖЕНИЯ Q1=(P3\*10+P2)

0E10 78	MOV	A,B	; (A) -ЧИСЛО (P3P2)
0E11 C5	PUSH	B	
0E12 CD000E	CALL	П108	; (A) -ДВОИЧНОЕ ЧИСЛО Q1
0E15 C1	POP	B	

;ВЫЧИСЛЕНИЕ ВЫРАЖЕНИЯ Q1\*10

0E16 2600	MVI	H,0	
0E18 6F	MOV	L,A	; (H,L) -Q1
0E19 29	DAD	H	; (H,L) -Q1*2
0E1A E5	PUSH	H	
0E1B D1	POP	D	; (D,E) -Q1*2
0E1C 29	DAD	H	

```

0E1D 29      DAD      H      ; (H,L)-Q1*8
0E1E 19      DAD      D      ; (H,L)-Q1*(8+2)
;ВЫДЕЛЕНИЕ ЦИФРЫ P1
0E1F 79      MOV      A,C
0E20 E6F0    ANI      OF0H    ;МАСКА НА ЦИФРУ
0E22 0F      RRC
0E23 0F      RRC
0E24 0F      RRC
0E25 0F      RRC      ; (A)-ЦИФРА P1
;ВЫЧИСЛЕНИЕ ВЫРАЖЕНИЯ Q2=(Q1*10+P1)
0E26 1600    MVI      D,0
0E28 5F      MOV      E,A
0E29 19      DAD      D      ; (H,L)-Q2
;ВЫЧИСЛЕНИЕ ВЫРАЖЕНИЯ Q2*10
0E2A 29      DAD      H      ; (H,L)-Q2*2
0E2B E5      PUSH     H
0E2C D1      POP      D      ; (D,E)-Q2*2
0E2D 29      DAD      H
0E2E 29      DAD      H      ; (H,L)-Q2*8
0E2F 19      DAD      D      ; (H,L)-Q2*10
;ВЫЧИСЛЕНИЕ ВЫРАЖЕНИЯ (Q2*10+P0)
0E30 79      MOV      A,C
0E31 E60F    ANI      OFH     ;МАСКА НА ЦИФРУ
0E33 1600    MVI      D,0
0E35 5F      MOV      E,A
0E36 19      DAD      D      ; (H,L)-(((P3*10+P2)*10+P1)*10+P0
0E37 C9      RET
0000      END

```

Исходное десятичное число представляется, согласно методу (3.2), в виде  $A_{10} = ((P_3 \cdot 10 + P_2) \cdot 10 + P_1) \cdot 10 + P_0$ , где  $P_i$  — десятичные цифры. Для вычисления выражения в первых скобках используется подпрограмма П108. При выполнении последующих вычислений производятся распаковка двух младших  $P_1 P_0$  цифр числа, сложение и умножение на 10 по правилу  $10 = 8 + 2$  с применением команд двухбайтного сложения. Для тестирования программ П108 и П1016 можно использовать данные таблиц, приведенных в прил. 2.

### 3.3. ПРЕОБРАЗОВАНИЯ ЦЕЛЫХ ДВОИЧНЫХ ЧИСЕЛ В ДЕСЯТИЧНЫЕ

Программа П810 преобразует однобайтное двоичное число  $A_2 \in [00, FFH]$  в эквивалентное трехразрядное десятичное число  $A_{10} \in [000, 255]$  по методам (3.1), (3.2):

```

0E40      ORG      0E40H
П810:
;*****
;ПОДПРОГРАММА ПРЕОБРАЗОВАНИЯ ДВОИЧНОГО ЦЕЛОГО БЕЗЗНА-
;КОВОГО ЧИСЛА ФОРМАТА 8 В ДВОИЧНО-ДЕСЯТИЧНОЕ (КОД

```



```

;8421) ЧИСЛО ФОРМАТА 3*4.
;ВХОДНОЙ ПАРАМЕТР: (С) – ДВОИЧНОЕ ЧИСЛО. ВЫХОДНОЙ ПАРАМЕТР:
;(Н, L) – ЭКВИВАЛЕНТНОЕ ДВОИЧНО-ДЕСЯТИЧНОЕ ЧИСЛО (3 ЦИФРЫ).
;ИСПОЛЬЗУЮТСЯ РЕГИСТРЫ В, А, СОХРАНЯЕТСЯ (С).
;ОЦЕНКА: ДЛИНА – 21 БАЙТ, ВРЕМЯ – 547 ТАКОВ.
;*****
0E40 210000      LXI    H, 0      ;ОБНУЛЕНИЕ ТЕКУЩЕЙ СУММЫ
0E43 0608        MVI    B, 8      ;СЧЕТЧИК ЦИКЛОВ
;СДВИГ ДВОИЧНОГО ЧИСЛА ВЛЕВО
0E45 79          ЦИКЛ:  MOV    A, C
0E46 07          RLC
0E47 4F          MOV    C, A
;ДВОИЧНО-ДЕСЯТИЧНОЕ УДВОЕНИЕ СУММЫ С УЧЕТОМ ПЕРЕНОСА
0E48 7D          MOV    A, L
0E49 8D          ADC    L
0E4A 27          DAA
0E4B 6F          MOV    L, A
0E4C 7C          MOV    A, H
0E4D 8C          ADC    H
0E4E 27          DAA
0E4F 67          MOV    H, A
;ПРОВЕРКА КОНЦА ЦИКЛА
0E50 05          DCR    B
0E51 C2450E      JNZ     ЦИКЛ      ;ЗАЦИКЛИВАНИЕ
0E54 C9          RET
0000            END

```

Программа вычисляет выражение (3.2) путем последовательного сдвига двоичного числа влево. При этом значение признака переноса CY совпадает со значением очередной выдвинутой старшей двоичной цифры:  $CY = a_{n-i}$ , где  $n = 8$ ,  $i = 1, 2, \dots, 8$ . Умножение на основание  $R = 2$  и сложение с очередной цифрой двоичного числа производятся методом двоично-десятичного сложения. Вычисление заканчивается через 8 циклов сдвигов и сложений.

Программа П1610, как и программа П810, преобразует двухбайтное двоичное число  $A_2 \in [0000, FFFFH]$  в эквивалентное пятиразрядное десятичное число  $A_{10} \in [00000, 65535]$  по методам (3.1), (3.2):

```

0E60                                ORG      0E60H
;*****
;ПОДПРОГРАММА ПРЕОБРАЗОВАНИЯ ДВОИЧНОГО ЦЕЛОГО БЕЗЗНА-
;КОВОГО ЧИСЛА ФОРМАТА 16 В ДВОИЧНО-ДЕСЯТИЧНОЕ (КОД
;8421) ЧИСЛО ФОРМАТА 5*4.
;ВХОДНОЙ ПАРАМЕТР: (В, С) – ДВОИЧНОЕ ЧИСЛО. ВЫХОДНЫЕ ПАРА-
;МЕТРЫ: (А, Н, L) – ЭКВИВАЛЕНТНОЕ ДВОИЧНО-ДЕСЯТИЧНОЕ ЧИСЛО
;(5 ЦИФР). ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ.
;ОЦЕНКА: ДЛИНА – 29 БАЙТ, ВРЕМЯ – 1443 ТАКТА.
;*****
;ОБНУЛЕНИЕ ТЕКУЩЕЙ СУММЫ (С, Н, L)
0E60 AF          XRA    A

```

0E61 67	MOV	H, A	
0E62 6F	MOV	L, A	
0E63 50	MOV	D, B	
0E64 59	MOV	E, C	; (D, E) - ИСХОДНОЕ ЧИСЛО
0E65 4F	MOV	C, A	
0E66 0610	MVI	B, 16	; СЧЕТЧИК ЦИКЛОВ
	; СДВИГ ДВОИЧНОГО ЧИСЛА ВЛЕВО		
0E68 EB	XCHG		
0E69 29	DAD	H	
0E6A EB	XCHG		
	; ДВОИЧНО-ДЕСЯТИЧНОЕ УДВОЕНИЕ СУММЫ С УЧЕТОМ ПЕРЕНОСА		
0E6B 7D	MOV	A, L	
0E6C 8D	ADC	L	
0E6D 27	DAA		
0E6E 6F	MOV	L, A	
0E6F 7C	MOV	A, H	
0E70 8C	ADC	H	
0E71 27	DAA		
0E72 67	MOV	H, A	
0E73 79	MOV	A, C	
0E74 89	ADC	C	
0E75 27	DAA		
0E76 4F	MOV	C, A	
	; ПРОВЕРКА КОНЦА ЦИКЛА		
0E77 05	DCR	B	
0E78 C2680E	JNZ	ЦИКЛ	; ЗАЦИКЛИВАНИЕ
0E7B 79	MOV	A, C	
0E7C C9	RET		
0000	END		

Программа заканчивает вычисления через 16 циклов сдвигов и сложений. Сопоставляя программы П810 и П1610, можно заметить, что увеличение в два раза разрядности обрабатываемых чисел приводит почти к трехкратному росту времени обработки. Время преобразования двухбайтных двоичных чисел можно немного сократить (увеличив затраты памяти), если вместо вычисления выражения (3.2) непосредственно использовать вычисление по формуле (3.3) с табличным заданием двоичных значений десятичных степеней:  $10^1_{10} = 000A_{16}$ ;  $10^2_{10} = 0064_{16}$ ;  $10^3_{10} = 03E8_{16}$ ;  $10^4_{10} = 2710_{16}$ . При этом, например, вычисление старшего коэффициента  $a_4$  сводится к последовательному вычитанию степени  $10^4$  из исходного двоичного числа  $A_2$  вплоть до получения отрицательного остатка. Количество выполненных вычитаний (за исключением последнего), очевидно, дает значение цифры  $a_4$ . Последний отрицательный остаток необходимо восстановить и использовать его аналогично ранее указанному для вычисления очередной цифры  $a_3$  и т. д. При программной реализации этого метода проще использовать вместо операции вычитания десятичных степеней суммирование

с дополнительными двоичными кодами этих степеней:  
 $[10^1_{10}]_d = \text{FFF6H}$ ;  $[10^2_{10}]_d = \text{FF9CH}$ ;  $[10^3_{10}]_d = \text{FC18H}$ ;  
 $[10^4_{10}]_d = \text{D8F0H}$ . Данный алгоритм реализует программа  
 П1610А:

0E80		ORG	0E80H
0E80	ц10	SET	0E80H

П1610А:  
 ;\*\*\*\*\*  
 ;ПОДПРОГРАММА ПРЕОБРАЗОВАНИЯ ДВОИЧНОГО ЦЕЛОГО БЕЗЗНА-  
 ;КОВОГО ЧИСЛА ФОРМАТА 16 В ДВОИЧНО-ДЕСЯТИЧНОЕ (КОД  
 ;8421) ЧИСЛО ФОРМАТА 5\*4.ВАРИАНТ А.  
 ;ВХОДНОЙ ПАРАМЕТР: (В,С)-ДВОИЧНОЕ ЧИСЛО.ВЫХОДНЫЕ ПА-  
 ;РАМЕТРЫ: (А,Н,Л)-ЭКВИВАЛЕНТНОЕ ДВОИЧНО-ДЕСЯТИЧНОЕ ЧИС-  
 ;ЛО (F4F3F2F1F0).ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ,ИСПОЛЬЗУ-  
 ;ЕТСЯ ПОДПРОГРАММА \*ц10\*.ГЛУБИНА СТЕКА-4.  
 ;ОЦЕНКА:ДЛИНА-43 БАЙТ (+16 БАЙТ ПОДПРОГРАММЫ),ВРЕМЯ-  
 ;НЕ БОЛЕЕ 1329 ТАКОВ (С УЧЕТОМ ПОДПРОГРАММЫ).  
 ;\*\*\*\*\*

0E80 60	MOV	H,B	
0E81 69	MOV	L,C	; (H,L)-ДВОИЧНОЕ ЧИСЛО
			;ВЫЧИСЛЕНИЕ ЦИФРЫ F4 В РАЗРЯДЕ Q4=10000
0E82 11F0DB	LXI	D,0D8F0H	;ДОПОЛНИТЕЛЬНЫЙ КОД Q4
0E85 CDB00E	CALL	ц10	; (A)-ЦИФРА F4, (H,L)-ОСТАТОК
0E88 F5	PUSH	PSW	;СОХРАНИТЬ ЦИФРУ
			;ВЫЧИСЛЕНИЕ ЦИФРЫ F3 В РАЗРЯДЕ Q3=1000
0E89 1118FC	LXI	D,0FC18H	;ДОПОЛНИТЕЛЬНЫЙ КОД Q3
0E8C CDB00E	CALL	ц10	; (A)-ЦИФРА F3, (H,L)-ОСТАТОК
			;УПАКОВКА ЦИФРЫ F3 В СТАРШУЮ ТЕТРАДУ (ВH)
0E8F 07	RLC		
0E90 07	RLC		
0E91 07	RLC		
0E92 07	RLC		
0E93 47	MOV	B,A	
			;ВЫЧИСЛЕНИЕ ЦИФРЫ F2 В РАЗРЯДЕ Q2=100
0E94 119CFF	LXI	D,0FF9CH	;ДОПОЛНИТЕЛЬНЫЙ КОД Q2
0E97 CDB00E	CALL	ц10	; (A)-ЦИФРА F2, (H,L)-ОСТАТОК
0E9A B0	ORA	B	
0E9B 47	MOV	B,A	; (B)-ЦИФРЫ (F3F2)
			;ВЫЧИСЛЕНИЕ ЦИФРЫ F1 В РАЗРЯДЕ Q1=10
0E9C 11F6FF	LXI	D,0FF6H	;ДОПОЛНИТЕЛЬНЫЙ КОД Q1
0E9F CDB00E	CALL	ц10	; (A)-ЦИФРА F1, (L)-ЦИФРА F0
			;УПАКОВКА ЦИФР F1F0 В (L)
0EA2 07	RLC		
0EA3 07	RLC		
0EA4 07	RLC		
0EA5 07	RLC		
0EA6 B5	ORA	L	
0EA7 6F	MOV	L,A	; (L)-ЦИФРЫ (F1F0)
0EA8 60	MOV	H,B	; (H)-ЦИФРЫ (F3F2)
0EA9 F1	POP	PSW	; (A)-ЦИФРА F4
0EAA C9	RET		
0000	END		

Программа последовательно вычисляет двоично-деся-  
 тичные цифры  $a_4 = P_4$ ,  $a_3 = P_3$ ,  $a_2 = P_2$ ,  $a_1 = P_1$ ,  $a_0 = P_0$

и упаковывает их в пятиразрядный формат двоично-десятичного числа. При этом для вычисления очередной цифры она обращается к подпрограмме Ц10:

ОЕВ0		ORG	ОЕВ0Н
	Ц10:		
	;*****		
	;ПОДПРОГРАММА ВЫЧИСЛЕНИЯ ДВОИЧНО-ДЕСЯТИЧНОЙ ЦИФРЫ В ЗА-		
	;ДАННОМ ДЕСЯТИЧНОМ РАЗРЯДЕ ЧИСЛА, ЭКВИВАЛЕНТНОГО ЦЕЛОМУ		
	;БЕЗЗНАКОВОМУ ДВОИЧНОМУ ЧИСЛУ ФОРМАТА 16.		
	;ВХОДНЫЕ ПАРАМЕТРЫ: (Н, L) - ДВОИЧНОЕ ЧИСЛО, (D, E) - КОНСТАНТА		
	; (ДОПОЛНИТЕЛЬНЫЙ КОД ДВОИЧНОГО ЧИСЛА, ЭКВИВАЛЕНТНОГО ЗА-		
	;ДАННОМУ ВЕСУ 10, 100, 1000, 10000 ДЕСЯТИЧНОГО РАЗРЯДА. ИС-		
	;ХОДНОЕ ЧИСЛО ДОЛЖНО БЫТЬ НЕ БОЛЕЕ 9-КРАТНОГО ЗНАЧЕНИЯ		
	;ПРЯМОГО КОДА ВЕСА). ВЫХОДНЫЕ ПАРАМЕТРЫ: (A) - ДВОИЧНО-ДЕ-		
	;СЯТИЧНАЯ ЦИФРА, (H, L) - ДВОИЧНЫЙ ОСТАТОК. ИСПОЛЬЗУЕТСЯ РЕ-		
	;ГИСТР C, СОХРАНЯЕТСЯ (D, E).		
	;ОЦЕНКА: ДЛИНА - 16 БАЙТ, ВРЕМЯ - НЕ БОЛЕЕ 280 ТАКТОВ.		
	;*****		
ОЕВ0 ОЕ00	MVI	C, 0	; ОБНУЛЕНИЕ ЦИФРЫ
	; ДВОИЧНОЕ СЛОЖЕНИЕ (ВЫЧИТАНИЕ ПРЯМОГО КОДА КОНСТАНТЫ)		
ОЕВ2 19	ЦИКЛ:	DAD	D
ОЕВ3 0C		INR	C
			; УВЕЛИЧЕНИЕ ЦИФРЫ
ОЕВ4 DAB20E		JC	ЦИКЛ
			; ЕСЛИ РАЗНОСТЬ > 0
	; КОРРЕКЦИЯ ЦИФРЫ И ВОССТАНОВЛЕНИЕ ОСТАТКА		
ОЕВ7 0D		DCR	C
			; УМЕНЬШЕНИЕ ЦИФРЫ
ОЕВ8 7D		MOV	A, L
ОЕВ9 93		SUB	E
ОЕВА 6F		MOV	L, A
ОЕВВ 7C		MOV	A, H
ОЕВС 9A		SBB	D
ОЕВД 67		MOV	H, A
			; (H, L) - ДВОИЧНЫЙ ОСТАТОК
ОЕВЕ 79		MOV	A, C
			; (A) - ЦИФРА
ОЕВF C9		RET	
0000		END	

Программа Ц10 выполняет вычитание десятичной степени из остатка двоичного числа, формирует очередную двоично-десятичную цифру и восстанавливает отрицательный остаток двоичного числа. Программа П1610А совместно с подпрограммой Ц10 осуществляет преобразование на 8 % быстрее, чем П1610, но требует на 100 % больше затрат памяти. Для тестирования программ П810, П1610 и П1610А можно использовать данные таблиц из прил. 2.

### 3.4. ПРЕОБРАЗОВАНИЯ ДРОБНЫХ ДЕСЯТИЧНЫХ ЧИСЕЛ В ДВОИЧНЫЕ

Для программной реализации перевода десятичных дробей в двоичные целесообразно использовать метод (3.5). Однако, поскольку ранее получены программы

аналогичного перевода для целых чисел, решение данной задачи можно упростить, используя готовые средства. Переход от целого числа к дробному осуществляется путем деления целого числа на некоторую степень. Поэтому преобразование десятичной дроби в двоичную можно представить как преобразование целого десятичного числа, изображенного в виде десятичной дроби, в эквивалентное целое двоичное число и деление последнего на соответствующую десятичную степень. Этот алгоритм реализуется программой ПФ10.

Программа ПФ10 преобразует четырехразрядную десятичную дробь  $A_{10} \in [0,0000; 0,9999]$  в эквивалентную двухбайтную двоичную дробь  $A_2 \in [0000; , FFFFH]$ :

ОЕСО		ORG	ОЕСОН
ОЕ10	П1016	SET	ОЕ10Н
ОВ20	Д216	SET	ОВ20Н

ПФ10:

```

;*****
;ПОДПРОГРАММА ПРЕОБРАЗОВАНИЯ ДВОИЧНО-ДЕСЯТИЧНОГО БЕЗЗНА-
;КОВОГО ЧИСЛА С ФИКСИРОВАННОЙ ПЕРЕД СТАРШИМ РАЗРЯДОМ ЗА-
;ПЯТОЙ ФОРМАТА 4*4 В ДВОИЧНОЕ ЧИСЛО С ФИКСИРОВАННОЙ ЗА-
;ПЯТОЙ ФОРМАТА 16.
;ВХОДНОЙ ПАРАМЕТР: (В,С) -ДВОИЧНО-ДЕСЯТИЧНОЕ ЧИСЛО (0,Р1Р2
;Р3Р4). ВЫХОДНОЙ ПАРАМЕТР: (Н,Л) -ЭКВИВАЛЕНТНОЕ ДВОИЧНОЕ
;ЧИСЛО. ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ, ГЛУБИНА СТЕКА -6. ИСПОЛЬ-
;ЗУЮТСЯ ПОДПРОГРАММЫ: *П108*, *П1016*, *Д216*, *ДОПВ*.
;ОЦЕНКА: ДЛИНА -11 БАЙТ (+102 БАЙТ ПОДПРОГРАММ), ВРЕМЯ -НЕ
;БОЛЕЕ 2224 ТАКТОВ (С УЧЕТОМ ПОДПРОГРАММ).
;*****
;ДВОИЧНОЕ ПРЕОБРАЗОВАНИЕ ЧИСЛА (0,Р1Р2Р3Р4)*10**4
ОЕСО СД100Е      CALL    П1016      ; (Н,Л) -ЦЕЛОЕ ДВОИЧНОЕ ЧИСЛО
;ДЕЛЕНИЕ ДВОИЧНОГО ЦЕЛОГО ЧИСЛА НА 10**4=2710Н
ОЕСЗ 011027      LXI      В,2710Н ;ЗАГРУЗКА КОНСТАНТЫ 10000
ОЕС6 СД200В      CALL    Д216      ; (Д,Е) -ЧАСТНОЕ
ОЕС9 ЕВ          XCHG          ; (Н,Л) -ЧАСТНОЕ, (Д,Е) -ОСТАТОК
ОЕСА С9          RET
0000             END

```

С помощью подпрограммы П1016 эта программа преобразует целое десятичное число в двоичное, а затем делит последнее на степень  $10^4_{10} = 2710_{16}$ . В соответствии с формулой (3.7) точность преобразования достаточна для сохранения точности исходной десятичной дроби. Тестовые данные для программы ПФ10 приведены в табл. 3.1.

Табл. 3.1. Дробные числа  $A_{10} \rightarrow A_{16}$ 

$A_{16}$	$A_{16}$	$A_{10}$	$A_{16}$
0,5000	,8000	0,6666	,AAA6
0,2500	,4000	0,9960	,FEF9
0,9999	,FFF9	0,0038	,00F9
0,5555	,8E35	0,1250	,2000
0,3333	,5553	0,0625	,1000

### 3.5. ПРЕОБРАЗОВАНИЯ ДРОБНЫХ ДВОИЧНЫХ ЧИСЕЛ В ДЕСЯТИЧНЫЕ

Программа ПФ16 преобразует двухбайтную двоичную дробь  $A_2 \in [0,0000; ,FFFFH]$  в эквивалентную пятиразрядную десятичную дробь  $A_{10} \in [0,00000; 0,99999]$  по формуле (3.5):

```

OED0      ORG      OED0H
OF00      SET      OF00H

ПФ16:
;*****
;ПОДПРОГРАММА ПРЕОБРАЗОВАНИЯ ДВОИЧНОГО БЕЗЗНАКОВОГО ЧИС-
;ЛА С ФИКСИРОВАННОЙ ПЕРЕД СТАРШИМ РАЗРЯДОМ ЗАПЯТОЙ ФОР-
;МАТА 16 В ДВОИЧНО-ДЕСЯТИЧНОЕ ЧИСЛО С ФИКСИРОВАННОЙ ЗА-
;ПЯТОЙ (,F1P2F3P4F5) ФОРМАТА 5*4.
;ВХОДНОЙ ПАРАМЕТР: (B,C) - ДВОИЧНОЕ ЧИСЛО. ВЫХОДНЫЕ ПАРАМЕТ-
;РЫ: (A,H,L) - ЭКВИВАЛЕНТНОЕ ДВОИЧНО-ДЕСЯТИЧНОЕ ЧИСЛО. ИС-
;ПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ. ГЛУБИНА СТЕКА - 10. ИСПОЛЬЗУЮТСЯ
;ПОДПРОГРАММЫ: *УФЦ10*.
;ОЦЕНКА: ДЛИНА - 38 БАЙТ (+30 БАЙТ ПОДПРОГРАММ), ВРЕМЯ - НЕ
;БОЛЕЕ 1099 ТАКТОВ.
;*****
;ПОСЛЕДОВАТЕЛЬНОЕ УМНОЖЕНИЕ ИСХОДНОГО ЧИСЛА НА 10

OED0 CD000F      CALL    УФЦ10
OED3 F5          PUSH    PSW      ; СОХРАНЕНИЕ ЦИФРЫ P1
OED4 CD000F      CALL    УФЦ10
OED7 F5          PUSH    PSW      ; СОХРАНЕНИЕ ЦИФРЫ P2
OED8 CD000F      CALL    УФЦ10
OEDB F5          PUSH    PSW      ; СОХРАНЕНИЕ ЦИФРЫ P3
OEDC CD000F      CALL    УФЦ10
OEDF F5          PUSH    PSW      ; СОХРАНЕНИЕ ЦИФРЫ P4
OEE0 CD000F      CALL    УФЦ10
OEE3 6F          MOV     L,A       ; (L) - ЦИФРА P5
;УПАКОВКА ЦИФР РЕЗУЛЬТАТА В РЕГИСТРЫ
OEE4 F1          POP     PSW      ; ВОССТАНОВЛЕНИЕ ЦИФРЫ P4
OEE5 07          RLC
OEE6 07          RLC
OEE7 07          RLC
OEE8 07          RLC
OEE9 B5          ORA     L
OEEA 6F          MOV     L,A       ; (L) - ЦИФРЫ P4,P5
OEEB F1          POP     PSW      ; ВОССТАНОВЛЕНИЕ ЦИФРЫ P3

```

0EEC 67	MOV	H,A	; (H)-ЦИФРА P3
0EED F1	POP	PSW	; ВОССТАНОВЛЕНИЕ ЦИФРЫ P2
0EEE 07	RLC		
0EEF 07	RLC		
0EF0 07	RLC		
0EF1 07	RLC		
0EF2 B4	ORA	H	
0EF3 67	MOV	H,A	; (H,L)-ЦИФРЫ P2P3P4P5
0EF4 F1	POP	PSW	; ВОССТАНОВЛЕНИЕ ЦИФРЫ P1
0EF5 C9	RET		; (A)-ЦИФРА P1
0000	END		

Программа выполняет последовательное умножение исходной двоичной дроби на основание десятичной системы и выделение очередной двоично-десятичной цифры с помощью подпрограммы УЦФ10:

0F00	ORG	0F00H	
	УЦФ10:		
	;*****		
	; ПОДПРОГРАММА УМНОЖЕНИЯ ДВОИЧНОГО БЕЗЗНАКОВОГО ЧИСЛА С		
	; ФИКСИРОВАННОЙ ПЕРЕД СТАРШИМ РАЗРЯДОМ ЗАПЯТОЙ ФОРМАТА 16		
	; НА ОСНОВАНИЕ ДЕСЯТИЧНОЙ СИСТЕМЫ: 10.		
	; ВХОДНОЙ ПАРАМЕТР: (B,C)-ЧИСЛО С ФЗ. ВЫХОДНЫЕ ПАРАМЕТРЫ:		
	; (A)-ЦЕЛАЯ ЧАСТЬ ПРОИЗВЕДЕНИЯ (ДВОИЧНО-ДЕСЯТИЧНАЯ ЦИФРА)		
	; (B,C)-ДРОБНАЯ ЧАСТЬ ПРОИЗВЕДЕНИЯ. ИСПОЛЮЮТСЯ ВСЕ РЕ-		
	; ГИСТРЫ.		
	; ОЦЕНКА: ДЛИНА-30 БАЙТ, ВРЕМЯ-172 ТАКТА.		
	;*****		
	; ПОДГОТОВКА РЕГИСТРОВ		
0F00 60	MOV	H,B	
0F01 69	MOV	L,C	; (H,L)-ЧИСЛО С ФЗ
0F02 50	MOV	D,B	
0F03 59	MOV	E,C	; (D,E)-ЧИСЛО С ФЗ
0F04 AF	XRA	A	; (A)=0
0F05 47	MOV	B,A	
0F06 4F	MOV	C,A	; (B,C)=0
	; СДВИГ ЧИСЛА С ФЗ ВЛЕВО НА 1 РАЗРЯД В (B,H,L)		
0F07 29	DAD	H	
0F08 88	ADC	B	
0F09 47	MOV	B,A	; (B,H,L)-ЧИСЛО С ФЗ*2
	; СДВИГ ЧИСЛА С ФЗ ВЛЕВО НА 3 РАЗРЯДА В (C,D,E)		
0F0A EB	XCHG		; (H,L)-ЧИСЛО С ФЗ
0F0B 29	DAD	H	; ПЕРВЫЙ СДВИГ
0F0C 79	MOV	A,C	
0F0D 17	RAL		
0F0E 4F	MOV	C,A	
0F0F 29	DAD	H	; ВТОРОЙ СДВИГ
0F10 79	MOV	A,C	
0F11 17	RAL		
0F12 4F	MOV	C,A	
0F13 29	DAD	H	; ТРЕТИЙ СДВИГ
0F14 79	MOV	A,C	
0F15 17	RAL		
0F16 4F	MOV	C,A	
0F17 EB	XCHG		; (C,D,E)-ЧИСЛО С ФЗ*8

# СЛОЖЕНИЕ СДВИНУТЫХ ЧИСЕЛ: УМНОЖЕНИЕ НА 10

0F18 19	DAD	D	
0F19 78	MOV	A, B	
0F1A 89	ADC	C	; (A, H, L) - ЧИСЛО С ФЗ*10
0F1B 44	MOV	B, H	
0F1C 4D	MOV	C, L	; (B, C) - ДРОБНАЯ ЧАСТЬ
0F1D C9	RET		; (A) - ЦЕЛАЯ ЧАСТЬ
0000	END		

Эта подпрограмма умножает дробь на 10 по правилу  $10 = 8 + 2$ , используя команды сдвига и сложения. Целая часть произведения, т. е. искомая цифра, запоминается в аккумуляторе. После выполнения очередного умножения в программе ПФ16 каждая очередная двоично-десятичная цифра сохраняется в стеке, а после завершения цикла из пяти умножений, т. е. по достижении требуемой точности преобразования, эти цифры извлекаются из стека и упаковываются в формат результата. Тестовые данные для программы ПФ16 приведены в табл. 3.2.

Табл. 3.2. Дробные числа  $A_{16} \rightarrow A_{10}$

$A_{16}$	$A_{10}$	$A_{16}$	$A_{10}$
,8000	0,50000	,0400	0,01562
,4000	0,25000	,0200	0,00781
,2000	0,12500	,0100	0,00390
,1000	0,06250	,0080	0,00195
,0800	0,03125	,0040	0,00097

## 3.6. ПРЕОБРАЗОВАНИЯ ДЕСЯТИЧНЫХ ЧИСЕЛ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ В ДВОИЧНЫЕ ЧИСЛА С ПЛАВАЮЩЕЙ ЗАПЯТОЙ

Программа ПП310 преобразует двоично-десятичное число с плавающей запятой  $A_{10} = 10^{m_{10}} \cdot A_{10ф}$ , содержащее двоично-десятичный порядок  $m_{10}$  в прямом коде со знаком  $S_n$  и четырехразрядную беззнаковую двоично-десятичную мантиссу  $A_{10ф} = 0, P_1 P_2 P_3 P_4$ , в эквивалентное двоичное число с плавающей запятой  $A_2 = 2^{m_2} \cdot A_{2ф}$ , содержащее двоичный смещенный порядок  $m_2$  (смещение  $= +40H$ ) и двухбайтную беззнаковую двоичную мантиссу  $A_{2ф}$  (рис. 3.1):



14A0		ORG	14A0H
00A3	ОБНЗ	SET	0A3H
0F20	ПФ10H	SET	0F20H
1450	ПС10A	SET	1450H
1220	УДПЗЗ	SET	1220H
12C0	ДДПЗЗ	SET	12C0H

ППЗ10:

```

;*****
;ПОДПРОГРАММА ПРЕОБРАЗОВАНИЯ 3-БАЙТНОГО ДВОИЧНО-ДЕСЯ-
;ТИЧНОГО БЕЗЗНАКОВОГО ЧИСЛА С ПЛАВАЮЩЕЙ ЗАПЯТОЙ ФОРМАТА
;((0,F1P2P3P4*10**(+N)) В 3-БАЙТНОЕ ДВОИЧНОЕ БЕЗЗНАКО-
;ВОЕ ЧИСЛО С ПЛАВАЮЩЕЙ ЗАПЯТОЙ ФОРМАТА (8,16)=(ПОР,МАН),
;ГДЕ БАЙТ ДЕСЯТИЧНОГО ПОРЯДКА СОДЕРЖИТ БИТ ЗНАКА ПОКАЗА-
;ТЕЛЯ И 2-ЗНАЧНЫЙ ПОКАЗАТЕЛЬ СТЕПЕНИ N В ПРЯМОМ ДВОИЧНО-
;ДЕСЯТИЧНОМ КОДЕ (N<19), А 2 БАЙТА ДЕСЯТИЧНОЙ МАНТИССЫ-
;4-ЗНАЧНОЕ ДРОБНОЕ НОРМАЛИЗОВАННОЕ ДВОИЧНО-ДЕСЯТИЧНОЕ
;ЧИСЛО; БАЙТ ДВОИЧНОГО ПОРЯДКА СОДЕРЖИТ ЦЕЛОЧИСЛЕННЫЙ ДВО-
;ИЧНЫЙ ПОРЯДОК СО СМЕЩЕНИЕМ +40H, А 2 БАЙТА МАНТИССЫ-ДВО-
;ИЧНОЕ ДРОБНОЕ НОРМАЛИЗОВАННОЕ ЧИСЛО.
;ВХОДНЫЕ ПАРАМЕТРЫ: (В,С)-АДРЕС ДЕСЯТИЧНОГО ЧИСЛА (АДРЕС1)
; (Н, L)-АДРЕС ДВОИЧНОГО ЧИСЛА (АДРЕС2). ВЫХОДНОЙ ПАРАМЕТР:
;СУ=1-ПРИЗНАК ПЕРЕПОЛНЕНИЯ РАЗРЯДНОЙ СЕТКИ. ИСПОЛЗУЮТСЯ
;ВСЕ РЕГИСТРЫ, СОХРАНЯЮТСЯ (В,С), (Н, L). ГЛУБИНА СТЕКА-20.
;ИСПОЛЗУЮТСЯ ПОДПРОГРАММЫ: *ПФ10*, *П1016*, *Д216*, *ДОПВ*,
;*ДОПД*, *ДОПН*, *ПОСВ*, *КОМЗ*, *ОБНЗ*, *ПС10A*, *П10В*,
;*УДПЗЗ*, *УДП17*, *ДДПЗЗ*, *НМАН2*, *ПМА2*, *УЗЗБ*, *У24A*,
;*ДДП17*, *Д216A*, *ПФ10H*.
;ОЦЕНКА: ДЛИНА-88 БАЙТ (+601 БАЙТ ПОДПРОГРАММ), ВРЕМЯ-
;НЕ БОЛЕЕ 7771 ТАКТА (С УЧЕТОМ ПОДПРОГРАММ).
;*****
;ПРОВЕРКА НА ПЕРЕПОЛНЕНИЕ РАЗРЯДНОЙ СЕТКИ

```

14A0 0A	LDA	B	; (A)-БАЙТ ДЕСЯТИЧНОГО ПОРЯДКА
14A1 E67F	ANI	7FH	;УДАЛЕНИЕ ЗНАКА
14A3 FE19	CPI	19H	
14A5 FAAA14	JM	PER1	;ЕСЛИ N<19
14A8 37	STC		;N>18, CY=1
14A9 C9	RET		;ПЕРЕПОЛНЕНИЕ СЕТКИ

;ПРОВЕРКА ДЕСЯТИЧНОЙ МАНТИССЫ НА НОЛЬ

14AA 03	PER1:	INX	B
14AB 0A		LDA	B
14AC 5F		MOV	E, A
14AD 03		INX	B
14AE 0A		LDA	B
14AF 0B		DCX	B
14B0 0B		DCX	B

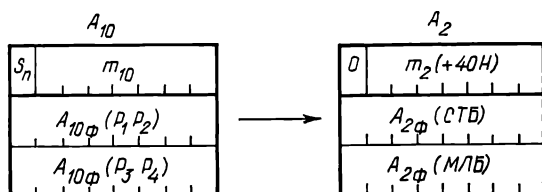


Рис. 3.1. Форматы десятичного  $A_{10}$  и двоичного  $A_2$  чисел в программе перевода ППЗ10

14B1 B3	ORA	E	
14B2 C2B914	JNZ	ПЕР2	ЕСЛИ МАНТИССА НЕ 0
14B5 CDA300	CALL	ОВНЗ	ОБНУЛЕНИЕ РЕЗУЛЬТАТА
14B8 C9	RET		
;ПРЕОБРАЗОВАНИЕ ДЕСЯТИЧНОЙ МАНТИССЫ В ДВОИЧНУЮ			
14B9 C5	ПЕР2: PUSH	B	;СОХРАНЕНИЕ АДРЕСА 1
14BA E5	PUSH	H	;СОХРАНЕНИЕ АДРЕСА 2
14BB 50	MOV	D,B	
14BC 59	MOV	E,C	; (D,E)-АДРЕС 1
14BD EB	XCHG		; (H,L)-АДРЕС 1
14BE 7E	MOV	A,M	; (A)-БАЙТ ПОКАЗАТЕЛЯ N
14BF F5	PUSH	PSW	;СОХРАНЕНИЕ БАЙТА N
14C0 23	INX	H	
14C1 46	MOV	B,M	
14C2 23	INX	H	
14C3 4E	MOV	C,M	; (B,C)-ДЕСЯТИЧНАЯ МАНТИССА
14C4 D5	PUSH	D	
14C5 CD200F	CALL	ПФ10H	; (H,L)-МАНТИССА, (A)-ПОРЯДОК
14C8 D1	POP	D	
;ЗАПИСЬ ДВОИЧНОЙ МАНТИССЫ И СМЕЩЕННОГО ПОРЯДКА В ПАМЯТЬ			
14C9 EB	XCHG		; (H,L)-АДРЕС 2
14CA 77	MOV	M,A	;ЗАПИСЬ ПОРЯДКА
14CB 23	INX	H	
14CC 72	MOV	M,D	;ЗАПИСЬ СТБ МАНТИССЫ
14CD 23	INX	H	
14CE 73	MOV	M,E	;ЗАПИСЬ МЛБ МАНТИССЫ
14CF 2B	DCX	H	
14D0 2B	DCX	H	
;ПРЕОБРАЗОВАНИЕ ДЕСЯТИЧНОЙ СТЕПЕНИ			
14D1 F1	POP	PSW	;ВОССТАНОВЛЕНИЕ БАЙТА N
14D2 F5	PUSH	PSW	;СОХРАНЕНИЕ БАЙТА N
14D3 E67F	ANI	7FH	;УДАЛЕНИЕ ЗНАКА
14D5 4F	MOV	C,A	; (C)-ПОКАЗАТЕЛЬ СТЕПЕНИ N
14D6 E5	PUSH	H	;СОХРАНЕНИЕ АДРЕСА 2
14D7 CD5014	CALL	ПС10A	; (C)-ПОРЯДОК, (H,L)-МАНТИССА
;ЗАПИСЬ ДВОИЧНОГО ЭКВИВАЛЕНТА СТЕПЕНИ В БУФЕР			
14DA EB	XCHG		; (D,E)-МАНТИССА
14DB 21F714	LXI	H,БУФЕР	
14DE 71	MOV	M,C	;ЗАПИСЬ ПОРЯДКА
14DF 23	INX	H	
14E0 72	MOV	M,D	;ЗАПИСЬ СТБ МАНТИССЫ
14E1 23	INX	H	
14E2 73	MOV	M,E	;ЗАПИСЬ МЛБ МАНТИССЫ
14E3 2B	DCX	H	
14E4 2B	DCX	H	
14E5 C1	POP	B	; (B,C)-АДРЕС 2
;ПРОВЕРКА ЗНАКА ПОКАЗАТЕЛЯ СТЕПЕНИ N			
14E6 F1	POP	PSW	;ВОССТАНОВЛЕНИЕ БАЙТА N
14E7 17	RAL		; (CY)-ЗНАК ПОКАЗАТЕЛЯ
14E8 DAF114	JC	ПЕР3	ЕСЛИ ЗНАК "-"
;ЗНАК "+": УМНОЖЕНИЕ ЧИСЕЛ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ			
14EB CD2012	CALL	УАПЗЗ	;РЕЗУЛЬТАТ ПО АДРЕСУ 2
14EE E1	POP	H	;ВОССТАНОВЛЕНИЕ АДРЕСА 2
14EF C1	POP	B	;ВОССТАНОВЛЕНИЕ АДРЕСА 1
14F0 C9	RET		
;ЗНАК "-": ДЕЛЕНИЕ ЧИСЕЛ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ			
14F1 CDC012	ПЕР3: CALL	ДАПЗЗ	;РЕЗУЛЬТАТ ПО АДРЕСУ 2

14F4 E1	POP	H	;ВОССТАНОВЛЕНИЕ АДРЕСА 2
14F5 C1	POP	B	;ВОССТАНОВЛЕНИЕ АДРЕСА 1
14F6 C9	RET		
	;ОБЛАСТЬ ПАМЯТИ ПРОМЕЖУТОЧНОГО РЕЗУЛЬТАТА		
14F7	БУФЕР:	DS	3 ;3 БАЙТА
0000	END		

Максимальное десятичное число, представленное в формате двоичного числа с плавающей запятой, равно  $A_{10} = 0,92 \cdot 10^{19}$  (см. § 2.1). Поэтому программа ПП310 прежде всего проверяет модуль десятичного порядка и переходит к дальнейшим преобразованиям в случае, если  $m_{10} \leq 18$  (тем самым максимальное десятичное число ограничивается величиной  $A_{10\max} = 10^{18} \cdot 0,9999$ ), иначе фиксируется переполнение разрядной сетки и устанавливается признак  $CY = 1$ . На следующем шаге проверяется значение десятичной мантиссы, и в случае равенства ее нулю выполнение программы заканчивается и результат преобразования обнуляется с помощью подпрограммы ОБНЗ. Если значение мантиссы  $A_{10ф}$  не равно нулю, осуществляется перевод ее в двоичную систему счисления посредством подпрограммы ПФ10Н:

0F20		ORG	0F20H
0EC0	ПФ10	SET	0EC0H
0B20	D216	SET	0B20H

ПФ10Н:

```

;*****
;ПОДПРОГРАММА ПРЕОБРАЗОВАНИЯ ДВОИЧНО-ДЕСЯТИЧНОГО БЕЗЗНА-
;КОВОГО ЧИСЛА С ФИКСИРОВАННОЙ ПЕРЕД СТАРШИМ РАЗРЯДОМ ЗА-
;ПЯТОЙ ФОРМАТА (0,P1P2P3P4) В ДВОИЧНОЕ ЧИСЛО С ПЛАВАЮЩЕЙ
;ЗАПЯТОЙ ФОРМАТА (8,16)=(ПОР,МАН),ГДЕ БАЙТ ПОРЯДКА СО-
;ДЕРЖИТ ЦЕЛОЧИСЛЕННЫЙ ДВОИЧНЫЙ ПОРЯДОК СО СМЕЩЕНИЕМ +40H
;А ДВА БАЙТА МАНТИССЫ-ДРОБНОЕ НОРМАЛИЗОВАННОЕ БЕЗЗНАКО-
;ВОЕ ЧИСЛО.
;ВХОДНОЙ ПАРАМЕТР:(В,С)-ДЕСЯТИЧНАЯ МАНТИССА.ВЫХОДНЫЕ ПА-
;РАМЕТРЫ:(Н,L)-ДВОИЧНАЯ МАНТИССА,(А)-ДВОИЧНЫЙ ПОРЯДОК.
;ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ,ГЛУБИНА СТЕКА-8.ИСПОЛЬЗУЮТСЯ
;ПОДПРОГРАММЫ:П108*,П1016*,ПФ10*,ДДОПВ*,ДД216*.
;ОЦЕНКА:ДЛИНА- 35 БАЙТ (+113 БАЙТ ПОДПРОГРАММ),ВРЕМЯ-НЕ
;БОЛЕЕ 4352 ТАКТОВ (С УЧЕТОМ ПОДПРОГРАММ).
;*****
;ДВОИЧНОЕ ПРЕОБРАЗОВАНИЕ ЧИСЛА 0,P1P2P3P4
0F20 CDC00E      CALL    ПФ10      ; (Н,L)-ЧАСТНОЕ,(D,E)-ОСТАТОК
;ПРОВЕРКА:ДВОИЧНАЯ МАНТИССА НОРМАЛИЗОВАНА ?
0F23 7C          MOV     А,Н      ; (А)-СТВ МАНТИССЫ
0F24 17          RAL          ;
0F25 3E40        MVI     А,40H    ; (А)-СМЕЩЕННЫЙ НУЛЕВОЙ ПОРЯДОК
0F27 D8          RC          ;ЕСЛИ ЧИСЛО НОРМАЛИЗОВАНО
;ДОПОЛНИТЕЛЬНОЕ ДЕЛЕНИЕ ОСТАТКА НА 10**4=2710H
0F28 E5          PUSH    Н        ;СОХРАНЕНИЕ ЧАСТНОГО 1
0F29 EB          XCHG         ; (Н,L)-ОСТАТОК (ДЕЛИМОЕ)

```

0F2A 011027	LXI	B,2710H	; (B,C) - ДЕЛИТЕЛЬ=10000
0F2D CD200B	CALL	D216	; (D,E) - ЧАСТНОЕ 2
0F30 E1	POP	H	; (H,L) - ЧАСТНОЕ 1
	;НОРМАЛИЗАЦИЯ МАНТИССЫ:СДВИГ ВЛЕВО ЧАСТНОГО 1,ЧАСТНОГО 2		
0F31 0640	MVI	B,40H	; (B) - СМЕШЕННЫЙ НУЛЕВОЙ ПОРЯДОК
0F33 29	цикл: DAD	H	
0F34 EB	XCHG		
0F35 29	DAD	H	
0F36 EB	XCHG		
0F37 D23B0F	JNC	ПЕР	
0F3A 23	INX	H	;УЧЕТ ПЕРЕНОСА
0F3B 05	ПЕР: DCR	B	;УМЕНЬШЕНИЕ ПОРЯДКА
	;ПРОВЕРКА КОНЦА ЦИКЛА		
0F3C 7C	MOV	A,H	; (A) - СТБ МАНТИССЫ
0F3D 17	RAL		
0F3E D2330F	JNC	ЦИКЛ	;ЗАЦИКЛИВАНИЕ
0F41 78	MOV	A,B	; (A) - ПОРЯДОК
0F42 C9	RET		
0000	END		

Программа ПФ10Н преобразует четырехразрядную десятичную мантиссу в двоичную шестнадцатиразрядную мантиссу с нулевым смещенным двоичным порядком. Если двоичная мантисса оказывается ненормализованной, программа выполняет ее нормализацию влево (предварительно вычислив дополнительные разряды денормализованной мантиссы) и соответствующую коррекцию двоичного порядка. В результате формируется число  $A'_2 = 2^{m'_2} \cdot A_{2ф}$ , которое программа ППЗ10 записывает в память, а затем переходит к двоичному преобразованию десятичной степени  $10^{m_{10}}$ .

Вычисление двоичного значения десятичной степени выполняет программа ПС10:

13D0		ORG	13D0H
05D0	Y24A	SET	5D0H
0E00	P108	SET	0E00H

ПС10:

```

;*****
;ПОДПРОГРАММА ПРЕОБРАЗОВАНИЯ ДЕСЯТИЧНОЙ СТЕПЕНИ 10**N
; (N<58) В ДВОИЧНОЕ ЧИСЛО С ПЛАВАЮЩЕЙ ЗАПЯТОЙ ФОРМАТА
; (8,16)=(ПОР,МАН),ГДЕ БАЙТ ПОРЯДКА СОДЕРЖИТ ЦЕЛОЧИСЛЕН-
; НЫЙ ДВОИЧНЫЙ ПОРЯДОК СО СМЕЩЕНИЕМ +40H,А 2 БАЙТА МАН-
; ТИССЫ-СТБ,МЛБ-ДВОИЧНОЕ ДРОБНОЕ НОРМАЛИЗОВАННОЕ ЧИСЛО В
; ПРЯМОМ КОДЕ.
;ВХОДНОЙ ПАРАМЕТР: (C)-ПРЯМОЙ ДВОИЧНО-ДЕСЯТИЧНЫЙ КОД ПО-
; КАЗАТЕЛЯ СТЕПЕНИ N.ВЫХОДНЫЕ ПАРАМЕТРЫ: (C)-ДВОИЧНЫЙ ПО-
; РЯДОК, (H,L)-ДВОИЧНАЯ МАНТИССА.ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ
; ГЛУБИНА СТЕКА-4.ИСПОЛЬЗУЮТСЯ ПОДПРОГРАММЫ: *Y24A*, *P108*
;ОЦЕНКА:ДЛИНА-125 БАЙТ (+32 БАЙТА ПОДПРОГРАММ),ВРЕМЯ-НЕ
; БОЛЕЕ 5520 ТАКОВ ПРИ N=18 (С УЧЕТОМ ПОДПРОГРАММ).
;*****

```

```

;ПРОВЕРКА:ПОКАЗАТЕЛЬ СТЕПЕНИ N>4 ?
13D0 79      MOV     A,C
13D1 FE05    CPI     05
13D3 F20714  JP      PER1      ;ЕСЛИ N>4
;ПРОВЕРКА:ПОКАЗАТЕЛЬ СТЕПЕНИ N=0 ?
13D6 B7      ORA     A
13D7 C2E013  JNZ     PER2      ;ЕСЛИ N>0
;N=0.ЗАНЕСЕНИЕ В РЕЗУЛЬТАТ КОНСТАНТЫ 1=(41H,8000H)
13DA 210080  LXI     H,8000H ;МАНТИССА=0,50000
13DD 0E41    MVI     C,41H   ;ПОРЯДОК=1
13DF C9      RET
;ПРОВЕРКА:ПОКАЗАТЕЛЬ СТЕПЕНИ N=1 ?
13E0 FE01    PER2: CPI     01
13E2 C2EB13  JNZ     PER3      ;ЕСЛИ N>1
;N=1.ЗАНЕСЕНИЕ В РЕЗУЛЬТАТ КОНСТАНТЫ 10=(44H,A000H)
13E5 2100A0  LXI     H,0A000H ;МАНТИССА=0,62500
13E8 0E44    MVI     C,44H   ;ПОРЯДОК=4
13EA C9      RET
;ПРОВЕРКА:ПОКАЗАТЕЛЬ СТЕПЕНИ N=2 ?
13EB FE02    PER3: CPI     2
13ED C2F613  JNZ     PER4      ;ЕСЛИ N>2
;N=2.ЗАНЕСЕНИЕ В РЕЗУЛЬТАТ КОНСТАНТЫ 100=(47H,C800H)
13F0 2100C8  LXI     H,0C800H ;МАНТИССА=0,78125
13F3 0E47    MVI     C,47H   ;ПОРЯДОК=7
13F5 C9      RET
;ПРОВЕРКА:ПОКАЗАТЕЛЬ СТЕПЕНИ N=3 ?
13F6 FE03    PER4: CPI     3
13F8 C20114  JNZ     PER5      ;ЕСЛИ N=4
;N=3.ЗАНЕСЕНИЕ В РЕЗУЛЬТАТ КОНСТАНТЫ 1000=(4AH,FA00H)
13FB 2100FA  LXI     H,0FA00H ;МАНТИССА=0,97656
13FE 0E4A    MVI     C,4AH   ;ПОРЯДОК=10
1400 C9      RET
;N=4.ЗАНЕСЕНИЕ В РЕЗУЛЬТАТ КОНСТАНТЫ 10000=(4EH,9C40H)
1401 21409C  PER5: LXI     H,9C40H ;МАНТИССА=0,61035
1404 0E4E    MVI     C,4EH   ;ПОРЯДОК=14
1406 C9      RET
;ПРЕОБРАЗОВАНИЕ ПОКАЗАТЕЛЯ СТЕПЕНИ N В ДВОИЧНЫЙ КОД,
;ПОДГОТОВКА РЕГИСТРОВ К ЦИКЛУ ВОЗВЕДЕНИЯ В СТЕПЕНЬ
1407 CD000E  PER1: CALL    P108      ;(A)-РЕЗУЛЬТАТ ПРЕОБРАЗОВАНИЯ
140A D605    SUI     5
140C 4F      MOV     C,A      ;(C)-ПОКАЗАТЕЛЬ (N-5)
140D C21514  JNZ     PER6      ;ЕСЛИ (N-5)>0
1410 3E0A    MVI     A,0AH   ;(A)-МНОЖИТЕЛЬ=10
1412 C31814  JMP      PER7
1415 0D      PER6: DCR     C      ;(C)-ПОКАЗАТЕЛЬ (N-6)
1416 3E64    MVI     A,64H   ;(A)-МНОЖИТЕЛЬ=100
1418 111027  PER7: LXI     D,2710H ;(D,E)-МНОЖИМОЕ=10000
141B 0650    MVI     B,50H   ;(B)-СМЕЩЕННЫЙ ПОРЯДОК (40H+10H)

;ЦИКЛ ВОЗВЕДЕНИЯ В СТЕПЕНЬ УМНОЖЕНИЕМ НА 10 ИЛИ 100
141D C5      ЦИКЛ1: PUSH    B
141E CDD005  CALL    Y24A      ;(A,H,L)=ПРОИЗВЕДЕНИЕ
1421 C1      POP     B
1422 17      ЦИКЛ2: RAL
1423 DA2D14  JC      PER8      ;ЕСЛИ ЧИСЛО НОРМАЛИЗОВАНО
1426 1F      RAR

```

	;НОРМАЛИЗАЦИЯ:СДВИГ ПРОИЗВЕДЕНИЯ ВЛЕВО		
1427 29	DAD	H	;СДВИГ (H,L)
1428 17	RAL		;СДВИГ (A)
1429 05	DCR	B	;УМЕНЬШЕНИЕ ПОРЯДКА
142A C32214	JMP	ЦИКЛ2	;ЗАЦИКЛИВАНИЕ 2
	;ПРОВЕРКА ОКОНЧАНИЯ ЦИКЛА 1		
142D 1F	PER8: RAR		
142E 57	MOV	D,A	
142F 5C	MOV	E,H	; (D,E)-СТЕ,СРБ ПРОИЗВЕДЕНИЯ
1430 78	MOV	A,B	
1431 C608	ADI	08	;УВЕЛИЧЕНИЕ ПОРЯДКА НА 8
1433 47	MOV	B,A	
1434 79	MOV	A,C	
1435 B7	ORA	A	
1436 CA4514	JZ	PER9	;ЕСЛИ ЦИКЛ ОКОНЧЕН
	;ПОДГОТОВКА РЕГИСТРОВ К ПРОДОЛЖЕНИЮ ЦИКЛА 1		
1439 0D	DCR	C	;УМЕНЬШЕНИЕ ПОКАЗАТЕЛЯ N
143A 3E0A	MVI	A,0AH	; (A)-МНОЖИТЕЛЬ=10
143C CA1D14	JZ	ЦИКЛ1	;ЗАЦИКЛИВАНИЕ 1
143F 0D	DCR	C	;УМЕНЬШЕНИЕ ПОКАЗАТЕЛЯ N
1440 3E64	MVI	A,64H	; (A)-МНОЖИТЕЛЬ=100
1442 C31D14	JMP	ЦИКЛ1	;ЗАЦИКЛИВАНИЕ 1
	;ЗАПИСЬ РЕЗУЛЬТАТА В РЕГИСТРЫ		
1445 48	PER9: MOV	C,B	; (C)-СМЕЩЕННЫЙ ПОРЯДОК
1446 45	MOV	B,L	; (B)-МЛБ ПРОИЗВЕДЕНИЯ
1447 6B	MOV	L,E	; (L)-СРБ ПРОИЗВЕДЕНИЯ
1448 62	MOV	H,D	; (H)-СТЕ ПРОИЗВЕДЕНИЯ
	;ОКРУГЛЕНИЕ РЕЗУЛЬТАТА ПО МЛБ ПРОИЗВЕДЕНИЯ		
1449 78	MOV	A,B	; (A)-МЛБ ПРОИЗВЕДЕНИЯ
144A 17	RAL		
144B D0	RNC		;ЕСЛИ СТАРШИЙ РАЗРЯД=0
144C 23	INX	H	
144D C9	RET		
0000	END		

Программа преобразует величину  $10^{m_{10}}$  в двоичное число с плавающей запятой вида  $A'_2 = 2^{m'_2} \cdot A_{2ф}$ . Преобразование выполняется, во-первых, переводом двоично-десятичного порядка  $m_{10}$  в двоичный посредством подпрограммы П108 и, во-вторых, последовательным умножением двоичного значения исходной десятичной степени на двоичное значение десятичного основания  $10_{10} = 0A_{16}$  или его квадрата  $10^2_{10} = 100_{10} = 64_{16}$  посредством подпрограммы У24А. С целью минимизации общего количества умножений и, следовательно, уменьшения времени работы программы ПС10 использован табличный способ преобразования степени для  $m_{10} < 5$ . При  $m > 4$  двоичное представление степени  $10^4_{10} = 2710_{16}$  используется в качестве исходного в цикле вычисления более высокой степени. Ускорение вычисления степени достигается и за счет умножения промежуточной степени на квадрат основания. В процессе возведения в степень выполняются нормали-

зация двоичной мантиссы и формирование соответствующего смещенного двоичного порядка.

Программа ПС10 требует больших затрат времени на преобразование степени. Ее целесообразно использовать один раз для предварительного расчета двоичных эквивалентов десятичных степеней, а в программе ПП310 применять уже готовые табличные данные. Такой табличный выбор производит подпрограмма ПС10А:

1450		ORG	1450H
0E00	П108	SET	0E00H

ПС10А:  
 ;\*\*\*\*\*  
 ;ПОДПРОГРАММА ПРЕОБРАЗОВАНИЯ ДЕСЯТИЧНОЙ СТЕПЕНИ 10\*\*N  
 ; (N(19) В ДВОИЧНОЕ ЧИСЛО С ПЛАВАЮЩЕЙ ЗАПЯТОЙ ФОРМАТА  
 ; (8,16)=(ПОР,МАН), ГДЕ БАЙТ ПОРЯДКА СОДЕРЖИТ ЦЕЛОЧИСЛЕН-  
 ; НЫЙ ДВОИЧНЫЙ ПОРЯДОК СО СМЕЩЕНИЕМ +40H, А 2 БАЙТА МАН-  
 ; ТИССЫ-ДВОИЧНОЕ ДРОБНОЕ НОРМАЛИЗОВАННОЕ ЧИСЛО В ПРЯМОМ  
 ; КОДЕ.  
 ; ВХОДНОЙ ПАРАМЕТР: (С)-ПРЯМОЙ ДВОИЧНО-ДЕСЯТИЧНЫЙ КОД ПОКА-  
 ; ЗАТЕЛЯ СТЕПЕНИ N. ВЫХОДНЫЕ ПАРАМЕТРЫ: (С)-ДВОИЧНЫЙ ПОРЯ-  
 ; ДОК, (H,L)-ДВОИЧНАЯ МАНТИССА. ИСПОЛЗУЮТСЯ ВСЕ РЕГИСТРЫ,  
 ; ГЛУБИНА СТЕКА-2. ИСПОЛЗУЕТСЯ ПОДПРОГРАММА: \*П108\*.  
 ; ОЦЕНКА: ДЛИНА-20 БАЙТ (+14 БАЙТ ПОДПРОГРАММЫ, +57 БАЙТ  
 ; ТАБЛИЦЫ), ВРЕМЯ-183 ТАКТА.  
 ;\*\*\*\*\*  
 ;ПРЕОБРАЗОВАНИЕ ПОКАЗАТЕЛЯ СТЕПЕНИ N В ДВОИЧНЫЙ КОД

1450 79	MOV	A, C	
1451 CD000E	CALL	П108	; (A)-РЕЗУЛЬТАТ ПРЕОБРАЗОВАНИЯ
1454 4F	MOV	C, A	

;АДРЕСАЦИЯ РЕЗУЛЬТАТА В ТАБЛИЦЕ

1455 0600	MVI	B, 0	
1457 216414	LXI	H, ТАБЛ	
145A 09	DAD	B	
145B 09	DAD	B	
145C 09	DAD	B	; (H,L)-АДРЕС ПОРЯДКА

;ЗАНЕСЕНИЕ РЕЗУЛЬТАТА В РЕГИСТРЫ

145D 4E	MOV	C, M	; (C)-ПОРЯДОК
145E 23	INX	H	
145F 56	MOV	D, M	; (D)-СТЕ МАНТИССЫ
1460 23	INX	H	
1461 5E	MOV	E, M	; (E)-МЛБ МАНТИССЫ
1462 EB	XCHG		; (H,L)-СТЕ, МЛБ МАНТИССЫ
1463 C9	RET		

1464 41800044	ТАБЛ: DB	41H, 80H, 00H, 44H, 0A0H, 00H
1468 A000		
146A 47C8004A	DB	47H, 0C8H, 00H, 4AH, 0F4H, 00H
146E FA00		
1470 4E9C4051	DB	4EH, 9CH, 40H, 51H, 0C3H, 50H
1474 C350		
1476 54F42458	DB	54H, 0F4H, 24H, 58H, 98H, 97H
147A 9897		
147C 5B8EBC5E	DB	5BH, 0BEH, 0BCH, 5EH, 0EEH, 6BH
1480 EE6B		

1482 62950365	DB	62H, 95H, 03H, 65H, 0BAH, 43H
1486 BA43		
1488 68E8D36C	DB	68H, 0E8H, 0D3H, 6CH, 91H, 84H
148C 9184		
148E 6FB5E572	DB	6FH, 0B5H, 0E5H, 72H, 0E3H, 5DH
1492 E35D		
1494 768E1A79	DB	76H, 8EH, 1AH, 79H, 0B1H, 0A1H
1498 B1A1		
149A 7CDE09	DB	7CH, 0DEH, 09H
0000	END	

В массиве ТАБЛ программы записаны двоичные эквиваленты ( $m_2''$ ,  $A_2''$ ) десятичных степеней  $10^0, \dots, 10^{18}$ .

После вычисления чисел  $A_2'$  и  $A_2''$  программа ППЗ10 проверяет знак  $S_n$  десятичного показателя степени и в зависимости от знака выполняет либо умножение  $A_2 = A_2' \cdot A_2''$  (знак «+»), либо деление  $A_2 = A_2' : A_2''$  (знак «-») промежуточных чисел, формируя окончательный результат преобразования. Тестовые данные для программы ППЗ10 приведены в табл. 3.3.

Табл. 3.3. Числа с плавающей запятой  $A_{10} \rightarrow A_{16}$

$A_{10}$	$A_{16}$	$A_{10}$	$A_{16}$
$0,1000 \cdot 10^0$	3DCCCC	$0,1000 \cdot 10^3$	47C7FF
$0,1200 \cdot 10^0$	3DF5C2	$0,1000 \cdot 10^{-5}$	2D8637
$0,1800 \cdot 10^0$	3EB852	$0,1000 \cdot 10^{-14}$	0F901D
$0,1200 \cdot 10^1$	419999	$0,1000 \cdot 10^{-18}$	01EC1F
$0,1000 \cdot 10^2$	449FFF	$0,1000 \cdot 10^{18}$	79B1A0

### 3.7. ПРЕОБРАЗОВАНИЯ ДВОИЧНЫХ ЧИСЕЛ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ В ДЕСЯТИЧНЫЕ ЧИСЛА С ПЛАВАЮЩЕЙ ЗАПЯТОЙ

Программа ПДПЗ2 выполняет обратное преобразование по отношению к программе ППЗ10, т. е. преобразует двоичное число с плавающей запятой (но уже знаковое число, представленное в дополнительном коде)  $A_2 = \pm 2^{m_2} \cdot A_{2ф}$ , в эквивалентное двоично-десятичное число с плавающей запятой  $A_{10} = \pm 10^{m_{10}} \cdot A_{10ф}$  (рис. 3.2):



1500		ORG	1500H
0090	КОМЗ	SET	90H
00AC	ОБН4	SET	0ACH
0ED0	ПФ16	SET	0ED0H
0050	ДОПВ	SET	50H
0120	Л4СВН	SET	120H
1580	УФ102	SET	1580H
15E0	ДФ102	SET	15E0H
15A0	ПМАН6	SET	15A0H

# ПАПЗ2:

\*\*\*\*\*  
 ПОДПРОГРАММА ПРЕОБРАЗОВАНИЯ 3-БАЙТНОГО ДВОИЧНОГО ЧИСЛА  
 В ДОПОЛНИТЕЛЬНОМ КОДЕ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ ФОРМАТА (8,  
 16)=(ПОР,МАН) В 3-БАЙТНОЕ ДВОИЧНО-ДЕСЯТИЧНОЕ ЧИСЛО С  
 ПЛАВАЮЩЕЙ ЗАПЯТОЙ В ПРЯМОМ КОДЕ ФОРМАТА ((+0,Р1Р2Р3Р4\*  
 \*10\*\*(+N)),ГДЕ БАЙТ ДВОИЧНОГО ПОРЯДКА СОДЕРЖИТ БИТ ЗНА-  
 КА МАНТИССЫ И ЦЕЛОЧИСЛЕННЫЙ ДВОИЧНЫЙ ПОРЯДОК СО СМЕЩЕ-  
 НИЕМ +40H,А 2 БАЙТА МАНТИССЫ-ДРОБНОЕ ДВОИЧНОЕ НОРМАЛИ-  
 ЗОВАННОЕ ЧИСЛО В ДОПОЛНИТЕЛЬНОМ КОДЕ;ДВОИЧНО-ДЕСЯТИЧНОЕ  
 ЧИСЛО СОДЕРЖИТ БАЙТ ЗНАКА ЧИСЛА,2 БАЙТА ДВОИЧНО-ДЕСЯТИЧ-  
 НОЙ МАНТИССЫ,БАЙТ ЗНАКА ПОРЯДКА И БАЙТ ДВУХЗНАЧНОГО  
 ПОРЯДКА N.  
 ВХОДНОЙ ПАРАМЕТР:(Н,L)-АДРЕС ДВОИЧНОГО ЧИСЛА.РЕЗУЛЬТАТ  
 РАЗМЕЩАЕТСЯ В ОБЛАСТИ ПАМЯТИ "БУФЕР".ИСПОЛЬЗУЮТСЯ ВСЕ  
 РЕГИСТРЫ,СОХРАНЯЮТСЯ (Н,L).ГЛУБИНА СТЕКА-10.ИСПОЛЬЗУЮТ-  
 СЯ ПОДПРОГРАММЫ:\*КОМЗ\*,\*ДОПВ\*,\*ПФ16\*,\*Л4СВН\*,\*УФ102\*,  
 \*ДФ102\*,\*ПМАН6\*,\*УФЦ10\*.  
 ОЦЕНКА:ДЛИНА-122 БАЙТ (+254 БАЙТ ПОДПРОГРАММ),ВРЕМЯ-НЕ  
 БОЛЕЕ (2105+1679\*K) ТАКОВ,ГДЕ К-ДВОИЧНЫЙ ПОРЯДОК.  
 \*\*\*\*\*  
 ПРОВЕРКА ИСХОДНОГО ЧИСЛА НА НОЛЬ

1500 CD9000	CALL	КОМЗ	:Z=1,ЕСЛИ ЧИСЛО=0
1503 C21215	JNZ	ПЕР1	:ЕСЛИ ЧИСЛО НЕ 0
	:ОБНУЛЕНИЕ РЕЗУЛЬТАТА В БУФЕРЕ		
1506 E5	PUSH	H	:СОХРАНЕНИЕ АДРЕСА
1507 217A15	LXI	H,БУФЕР	
150A 3600	MVI	M,0	:ОБНУЛЕНИЕ 1-ГО БАЙТА
150C 23	INX	H	
150D CDAC00	CALL	ОБН4	:ОБНУЛЕНИЕ 4 БАЙТ

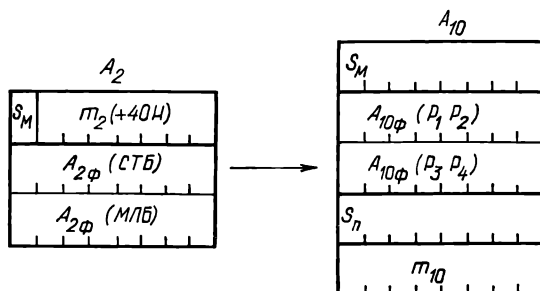


Рис. 3.2. Форматы двоичного  $A_2$  и десятичного  $A_{10}$  чисел в программе перевода ПДПЗ2

1510 E1	POP	H	;ВОССТАНОВЛЕНИЕ АДРЕСА
1511 C9	RET		;РЕЗУЛЬТАТ=0
;ВЫДЕЛЕНИЕ ЗНАКА МАНТИССЫ, ПЕРЕВОД ЕЕ В ПРЯМОЙ КОД			
1512 E5	PER1: PUSH	H	;СОХРАНЕНИЕ АДРЕСА
1513 7E	MOV	A, M	; (A) - БАЙТ ПОРЯДКА
1514 5F	MOV	E, A	;СОХРАНЕНИЕ БАЙТА ПОРЯДКА
1515 E680	ANI	80H	;ВЫДЕЛЕНИЕ ЗНАКА МАНТИССЫ
1517 327A15	STA	БУФЕР	;ЗАПОМИНАНИЕ ЗНАКА МАНТИССЫ
151A 23	INX	H	
151B 46	MOV	B, M	
151C 23	INX	H	
151D 4E	MOV	C, M	; (B, C) - МАНТИССА
151E F22415	JP	PER2	;ЕСЛИ ЗНАК МАНТИССЫ "+"
1521 CD5000	CALL	ДОПВ	; (B, C) - ДОПОЛНЕНИЕ МАНТИССЫ
;ПРЕОБРАЗОВАНИЕ СМЕЩЕННОГО ПОРЯДКА, ВЫДЕЛЕНИЕ ЕГО ЗНАКА			
1524 7B	PER2: MOV	A, E	; (A) - БАЙТ ПОРЯДКА
1525 E67F	ANI	7FH	;УДАЛЕНИЕ ЗНАКА МАНТИССЫ
1527 D640	SUI	40H	;УДАЛЕНИЕ СМЕЩЕНИЯ ПОРЯДКА
1529 FA3215	JM	PER3	;ЕСЛИ ПОРЯДОК < 0
152C 5F	MOV	E, A	; (E) - ДВОИЧНЫЙ ПОРЯДОК
152D 1600	MVI	D, 0	; (D) = 0 - ЗНАК ПОРЯДКА
152F C33715	JMP	PER4	
;ДОПОЛНЕНИЕ ОТРИЦАТЕЛЬНОГО ПОРЯДКА			
1532 2F	PER3: CMA		
1533 3C	INR	A	
1534 5F	MOV	E, A	; (E) - ДВОИЧНЫЙ ПОРЯДОК
1535 1680	MVI	D, 80H	; (D) = 80 - ЗНАК ПОРЯДКА
;ПРЕОБРАЗОВАНИЕ ДВОИЧНОЙ МАНТИССЫ В ДЕСЯТИЧНУЮ			
;САЙГ МАНТИССЫ НА 1 ДЕСЯТИЧНЫЙ РАЗРЯД ВЛЕВО			
1537 D5	PER4: PUSH	D	
1538 CDD00E	CALL	ПФ16	; (A, H, L) - ДЕСЯТИЧНАЯ МАНТИССА
153B D1	POP	D	
153C CD2101	CALL	Л4СВН+1	; (B, H, L) - ДЕСЯТИЧНАЯ МАНТИССА
;ПРОВЕРКА ВЕЛИЧИНЫ (НЕ 0 ?) И ЗНАКА ДВОИЧНОГО ПОРЯДКА			
153F AF	XRA	A	
1540 83	ADD	E	
1541 CA5415	JZ	PER5	;ЕСЛИ ПОРЯДОК=0
1544 7A	MOV	A, D	
1545 17	RAL		
1546 DA4F15	JC	PER6	;ЕСЛИ ЗНАК "-"
;УМНОЖЕНИЕ ДЕСЯТИЧНОЙ МАНТИССЫ НА ДВОИЧНЫЙ ПОРЯДОК			
1549 CD8015	CALL	УФ102	; (B, H, L) - ДЕСЯТИЧНАЯ МАНТИССА
154C C35415	JMP	PER5	; (D) - ДЕСЯТИЧНЫЙ ПОРЯДОК
;ДЕЛЕНИЕ ДЕСЯТИЧНОЙ МАНТИССЫ НА ДВОИЧНЫЙ ПОРЯДОК			
154F 1600	PER6: MVI	D, 0	
1551 CDE015	CALL	ДФ102	; (B, H, L) - МАНТИССА, (D) - ПОРЯДОК
;ОКРУГЛЕНИЕ 6-ЗНАЧНОЙ МАНТИССЫ ДО 4-ЗНАЧНОЙ			
1554 7D	PER5: MOV	A, L	; (A) - ОКРУГЛЯТОР
1555 FE50	CPI	50H	
1557 FA6715	JM	PER7	;ЕСЛИ ОКРУГЛЯТОР < 50H
155A 7C	MOV	A, H	
155B C601	ADI	01	
155D 27	DAA		
155E 67	MOV	H, A	
155F 78	MOV	A, B	

```

1560 CE00          ACI      00
1562 27           DAA
1563 47           MOV      B,A
1564 DCA015        CC      PMAN6      ; (B,H) - ДЕСЯТИЧНАЯ МАНТИССА
; ЗАПИСЬ РЕЗУЛЬТАТА В ПАМЯТЬ
1567 4C          ПЕР7: MOV      C,H      ; СОХРАНЕНИЕ (H)
1568 217B15       LXI      H,БУФЕР+1
156B 70           MOV      M,B      ; ЗАПИСЬ ЦИФР R1R2
156C 23           INX      H
156D 71           MOV      M,C      ; ЗАПИСЬ ЦИФР R3R4
156E 23           INX      H
156F 7A           MOV      A,D      ; (A) - ДЕСЯТИЧНЫЙ ПОРЯДОК СО ЗНАКОМ
1570 E680         ANI      80H      ; ВЫДЕЛЕНИЕ ЗНАКА ПОРЯДКА
1572 77           MOV      M,A      ; ЗАПИСЬ ЗНАКА ПОРЯДКА
1573 23           INX      H
1574 7A           MOV      A,D      ; (A) - ДЕСЯТИЧНЫЙ ПОРЯДОК СО ЗНАКОМ
1575 E67F         ANI      7FH      ; УДАЛЕНИЕ ЗНАКА ПОРЯДКА
1577 77           MOV      M,A      ; ЗАПИСЬ ПОРЯДКА
1578 E1           POP      H      ; ВОССТАНОВЛЕНИЕ АДРЕСА
1579 C9           RET
; ОБЛАСТЬ ЗАПОМИНАНИЯ РЕЗУЛЬТАТА
157A          БУФЕР: DS      5      ; 5 БАЙТ
0000          END

```

Программа прежде всего проверяет значение преобразуемого числа и в случае равенства его нулю обнуляет область результата и заканчивает работу. Если исходное число отличается от нуля, программа переводит мантиссу в прямой код и определяет модуль и знак несмещенного двоичного порядка. Далее двоичная мантисса преобразуется с помощью подпрограммы ПФ16 в эквивалентную пятиразрядную двоично-десятичную мантиссу, которая сдвигается на один десятичный разряд влево подпрограммой Л4СВН:

```

0120                                ORG      120H
; Л4СВН:
; *****
; ПОДПРОГРАММА ЛЕВОГО СДВИГА ЧИСЛА В РЕГИСТРАХ (B,H,L) С
; ЗАПОЛНЕНИЕМ СВОБОДНЫХ РАЗРЯДОВ НУЛЯМИ.
; ВХОДНОЙ ПАРАМЕТР: (B,H,L) - ИСХОДНОЕ ЧИСЛО. ВЫХОДНОЙ ПАРА-
; МЕТР: (B,H,L) - ЧИСЛО ПОСЛЕ СДВИГОВ. ИСПОЛЬЗУЕТСЯ РЕГИСТР A
; ОЦЕНКА: ДЛИНА - 11 БАЙТ, ВРЕМЯ - 76 ТАКТОВ.
; *****
0120 78          MOV      A,B      ; (A,H,L) - ИСХОДНОЕ ЧИСЛО
0121 29          DAD      H      ; ПЕРВЫЙ СДВИГ
0122 17          RAL
0123 29          DAD      H      ; ВТОРОЙ СДВИГ
0124 17          RAL
0125 29          DAD      H      ; ТРЕТИЙ СДВИГ
0126 17          RAL
0127 29          DAD      H      ; ЧЕТВЕРТЫЙ СДВИГ
0128 17          RAL
0129 47          MOV      B,A      ; (B,H,L) - ЧИСЛО ПОСЛЕ СДВИГОВ
012A C9          RET
0000          END

```

В результате сдвига старший десятичный разряд мантиссы совмещается со старшей тетрадой байта. Если двоичный несмещенный порядок оказывается равным нулю, преобразование числа в целом окончено: оно округляется, нормализуется и записывается в память. При ненулевом порядке в зависимости от его знака выполняется либо умножение (знак «+») десятичной мантиссы на двоичную характеристику подпрограммой УФ102, либо деление (знак «—») подпрограммой ДФ102.

Программа УФ102 выполняет умножение шестизначной двоично-десятичной беззнаковой мантиссы на двоичную характеристику  $2^{|m_2-40H|}$  с формированием двоично-десятичной мантиссы и двоично-десятичного порядка:

1580		ORG	1580H
15A0	ПМАН6	SET	15A0H

УФ102:

```

;*****
;ПОДПРОГРАММА УМНОЖЕНИЯ 6-ЗНАЧНОГО БЕЗЗНАКОВОГО ДВОИЧНО-
;ДЕСЯТИЧНОГО ЧИСЛА С ФИКСИРОВАННОЙ ПЕРЕД СТАРШИМ РАЗРЯ-
;ДОМ ЗАПЯТОЙ НА ДВОИЧНУЮ СТЕПЕНЬ ФОРМАТА 0,P1P2P3P4P5P6*
;*(2**K)=0,R1R2R3R4R5R6*(10**M),ГДЕ P(I),R(I)-ДВОИЧНО-
;ДЕСЯТИЧНАЯ ЦИФРА (I=1,...,6),K-ДВОИЧНЫЙ ПОРЯДОК,M-ДВО-
;ИЧНО-ДЕСЯТИЧНЫЙ ПОРЯДОК.
;ВХОДНЫЕ ПАРАМЕТРЫ:(B,H,L)-ИСХОДНОЕ ДЕСЯТИЧНОЕ ЧИСЛО,
;(E)-ДВОИЧНЫЙ ПОРЯДОК K,(D)=0.ВХОДНЫЕ ПАРАМЕТРЫ:
;(B,H,L)-РЕЗУЛЬТИРУЮЩЕЕ ДЕСЯТИЧНОЕ ЧИСЛО,(D)-ДЕСЯТИЧ-
;НЫЙ ПОРЯДОК.ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ,ГЛУБИНА СТЕКА-2.
;ИСПОЛЬЗУЕТСЯ ПОДПРОГРАММА *ПМАН6*.
;ОЦЕНКА:ДЛИНА-20 БАЙТ (+49 БАЙТ ПОДПРОГРАММЫ),ВРЕМЯ-НЕ
;БОЛЕЕ (10+172*K) ТАКТОВ.
;*****
;УМНОЖЕНИЕ ИСХОДНОГО ЧИСЛА НА 2 МЕТОДОМ СЛОЖЕНИЯ
ЦИКЛ:  MOV    A,L      ;СЛОЖЕНИЕ МЛЧ
        ADD    L
        DAA
        MOV    L,A
        MOV    A,H      ;СЛОЖЕНИЕ СРЧ
        ADC    H
        DAA
        MOV    H,A
        MOV    A,B      ;СЛОЖЕНИЕ СТЧ
        ADC    B
        DAA
        MOV    B,A
;НОРМАЛИЗАЦИЯ ЧИСЛА,КОРРЕКЦИЯ ДЕСЯТИЧНОГО ПОРЯДКА
        CC      ПМАН6   ;ЕСЛИ ПЕРЕПОЛНЕНИЕ
;ПРОВЕРКА КОНЦА ЦИКЛА
        DCR     E
        JNZ     ЦИКЛ   ;ЗАЦИКЛИВАНИЕ
        RET
        END

```

1580	7D		
1581	85		
1582	27		
1583	6F		
1584	7C		
1585	8C		
1586	27		
1587	67		
1588	78		
1589	88		
158A	27		
158B	47		
158C	DCA015		
158F	1D		
1590	C28015		
1593	C9		
0000			

Умножение мантиссы осуществляется методом двоично-десятичного сложения с накоплением, что неизбежно ведет к значительным затратам времени на выполнение программы при больших значениях двоичного порядка. Формирование двоично-десятичного порядка и нормализация вправо двоично-десятичной мантиссы производятся подпрограммой ПМАН6:

15A0'	ORG	15A0H
	ПМАН6:	
	;*****	
	;ПОДПРОГРАММА УСТРАНЕНИЯ ПЕРЕПОЛНЕНИЯ 6-ЗНАЧНОЙ БЕЗЗНА-	
	;КОВОЙ ДВОИЧНО-ДЕСЯТИЧНОЙ МАНТИССЫ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ.	
	;ВХОДНЫЕ ПАРАМЕТРЫ: (В,Н,Л)-ДЕСЯТИЧНАЯ МАНТИССА, (D)-	
	;ДЕСЯТИЧНЫЙ ПОРЯДОК СО ЗНАКОМ (МОДУЛЬ ПОРЯДКА < 79), CY=1	
	;-ПРИЗНАК ПЕРЕПОЛНЕНИЯ МАНТИССЫ. ВЫХОДНЫЕ ПАРАМЕТРЫ:	
	; (В,Н,Л)-НОРМАЛИЗОВАННАЯ ДЕСЯТИЧНАЯ МАНТИССА, (D)-ДЕСЯ-	
	;ТИЧНЫЙ ПОРЯДОК СО ЗНАКОМ. ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ, КРО-	
	;МЕ В, Е.	
	;ОЦЕНКА: ДЛИНА-49 БАЙТ, ВРЕМЯ-НЕ БОЛЕЕ 366 ТАКТОВ.	
	;*****	
15A0 0E04	MVI	C, 4 ; (C)-СЧЕТЧИК ЦИКЛОВ СДВИГА
	;СДВИГ МАНТИССЫ ВПРАВО НА 4 ДВОИЧНЫХ РАЗРЯДА	
15A2 78	CHG	A, B
15A3 1F	RAR	
15A4 47	MOV	B, A
15A5 7C	MOV	A, H
15A6 1F	RAR	
15A7 67	MOV	H, A
15A8 7D	MOV	A, L
15A9 1F	RAR	
15AA 6F	MOV	L, A
15AB AF	XRA	A ; CY=0
	;ПРОВЕРКА КОНЦА ЦИКЛА	
15AC 0D	DCR	C
15AD C2A215	JNZ	ЦИКЛ ;ЗАЦИКЛИВАНИЕ
	;ПРОВЕРКА ЗНАКА ДЕСЯТИЧНОГО ПОРЯДКА	
15B0 AF	XRA	A
15B1 82	ADD	D
15B2 FABA15	JM	ПЕР ;ЕСЛИ ЗНАК "-"
	;КОРРЕКЦИЯ ПОЛОЖИТЕЛЬНОГО ДЕСЯТИЧНОГО ПОРЯДКА	
15B5 C601	ADI	01
15B7 27	DAA	
15B8 57	MOV	D, A
15B9 C9	RET	
	;КОРРЕКЦИЯ ОТРИЦАТЕЛЬНОГО ДЕСЯТИЧНОГО ПОРЯДКА	
15BA E67F	PER:	ANI 7FH ;УДАЛЕНИЕ ЗНАКА
15BC 57	MOV	D, A
15BD 3E99	MVI	A, 99H ;ДЕСЯТИЧНОЕ ДОПОЛНЕНИЕ ПОРЯДКА
15BF 92	SUB	D
15C0 C602	ADI	02 ;КОРРЕКЦИЯ ДОПОЛНЕНИЯ ПОРЯДКА
15C2 27	DAA	
15C3 57	MOV	D, A
15C4 3E99	MVI	A, 99H ;ДОПОЛНЕНИЕ ДОПОЛНЕНИЯ ПОРЯДКА
15C6 92	SUB	D
15C7 C601	ADI	01

15C9 27	DAA		
15CA 57	MOV	D, A	
15CB C8	RZ		; ЕСЛИ ДЕСЯТИЧНЫЙ ПОРЯДОК=0
15CC 0E80	MVI	C, 80H	; (C) -ЗНАК "-"
15CE B1	ORA	C	; ВСТАВКА ЗНАКА
15CF 57	MOV	D, A	
15D0 C9	RET		; ДЕСЯТИЧНЫЙ ПОРЯДОК < 0
0000	END		

Программа ПМАН6 в случае переполнения двоично-десятичной мантиисы при ее удвоении в программе УФ102 сдвигает мантиису на один десятичный разряд вправо и должным образом корректирует десятичный порядок.

Деление двоично-десятичной мантиисы на двоичную характеристику  $2^{l_{m2}-40H}$  выполняется программой ДФ102:

15E0		ORG	15E0H
0120	Л4СВН	SET	120H

**ДФ102:**

```

;*****
;ПОДПРОГРАММА ДЕЛЕНИЯ 6-ЗНАЧНОГО БЕЗЗНАКОВОГО ДВОИЧНО-
;ДЕСЯТИЧНОГО ЧИСЛА С ФИКСИРОВАННОЙ ПЕРЕД СТАРШИМ РАЗРЯ-
;ДОМ ЗАПЯТОЙ НА ДВОИЧНУЮ СТЕПЕНЬ ФОРМАТА 0,F1P2P3P4P5P6:
;: (2**K)=0,R1R2R3R4R5R6*10**(-M), ГДЕ P(I), R(I) -ДВОИЧНО-
;ДЕСЯТИЧНАЯ ЦИФРА, I=1,...,6; K-ДВОИЧНЫЙ ПОРЯДОК
; (0 < K < 256); M-ДЕСЯТИЧНЫЙ ПОРЯДОК В ПРЯМОМ ДВОИЧНО-ДЕ-
;СЯТИЧНОМ КОДЕ (M<78).
;ВХОДНЫЕ ПАРАМЕТРЫ: (B,H,L)-ИСХОДНОЕ ДЕСЯТИЧНОЕ ЧИСЛО,
; (E)-ДВОИЧНЫЙ ПОРЯДОК K, (D)=0. ВЫХОДНЫЕ ПАРАМЕТРЫ:
; (B,H,L)-ДЕСЯТИЧНОЕ ЧАСТНОЕ, (D)-ДЕСЯТИЧНЫЙ ПОРЯДОК. ИС-
;ПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ, ГЛУБИНА СТЕКА-6. ИСПОЛЬЗУЕТСЯ
;ПОДПРОГРАММА *Л4СВН*.
;ОЦЕНКА: ДЛИНА-79 БАЙТ (+11 БАЙТ ПОДПРОГРАММЫ), ВРЕМЯ-НЕ
;БОЛЕЕ (39+1679*K) ТАКТОВ.
;*****
;ДЕЛЕНИЕ ДЕСЯТИЧНОГО ДЕЛИМОГО НА 2
ЦИКЛ1: PUSH    D      ; СОХРАНИТЬ ПОРЯДКИ
          MVI     D, 06 ; (D) -СЧЕТЧИК ЦИФР
15E1 1606      ;
          XRA     A      ; (A) -ОСТАТОК Q(I-1)*10=0
15E3 AF        ЦИКЛ2: PUSH    D      ; СОХРАНЕНИЕ СЧЕТЧИКА
15E4 D5        MOV     E, A      ; СОХРАНЕНИЕ ОСТАТКА
15E5 5F        ; ВЫДЕЛЕНИЕ ЦИФРЫ P(I) ДЕЛИМОГО, СЛОЖЕНИЕ ЕЕ С Q(I)*10
          MOV     A, B
15E6 78        ANI     0F0H      ; МАСКА НА СЦ
15E7 E6F0      RRC          ; СДВИГ СЦ НА МЕСТО МЛЦ
15E9 0F        RRC
15EA 0F        RRC
15EB 0F        RRC
15EC 0F        RRC
15ED 83        ADD     E
15EE 5F        MOV     E, A      ; (E)=Q(I-1)*10+P(I)
15EF CD2001    CALL    Л4СВН      ; СДВИГ ДЕЛИМОГО ВЛЕВО (B,H,L)
;ОПРЕДЕЛЕНИЕ ДЕСЯТИЧНОЙ ЦИФРЫ ЧАСТНОГО R(I)
          XRA     A      ; CY=0
15F2 AF        MOV     A, E      ; (A)=Q(I-1)*10+P(I)
15F3 7B        RAR          ; (A)=(Q(I-1)*10+P(I)):2=R(I)
15F4 1F

```

15F5 F5	PUSH	PSW	;СОХРАНЕНИЕ ОСТАТКА (CY)
15F6 B5	ORA	L	
15F7 6F	MOV	L,A	;ЗАПИСЬ ЦИФРЫ ЧАСТНОГО
15F8 F1	POP	PSW	;ВОССТАНОВЛЕНИЕ ОСТАТКА (CY)
	;ПРОВЕРКА КОНЦА ЦИКЛА 2		
15F9 D1	POP	D	;ВОССТАНОВЛЕНИЕ СЧЕТЧИКА
15FA 15	DCR	D	
15FB CA0716	JZ	ПЕР1	;ЕСЛИ КОНЕЦ ЦИКЛА 2
	;УМНОЖЕНИЕ НА 10 ОСТАТКА (CY) ДЕЛЕНИЯ P(I):2		
15FE 3E0A	MVI	A,0AH	;ОСТАТОК=1*10
1600 DAE415	JC	ЦИКЛ2	;ЗАЦИКЛИВАНИЕ 2
1603 AF	XRA	A	;ОСТАТОК=0*10
1604 C3E415	JMP	ЦИКЛ2	;ЗАЦИКЛИВАНИЕ 2
	;НОРМАЛИЗАЦИЯ ДЕСЯТИЧНОГО ДЕЛИМОГО		
1607 D1	ПЕР1: POP	D	;ВОССТАНОВЛЕНИЕ ПОРЯДКОВ
1608 F5	PUSH	PSW	;СОХРАНЕНИЕ ОСТАТКА Q(6)
1609 78	MOV	A,B	
160A E6F0	ANI	0F0H	;МАСКА НА СЦ
160C C22216	JNZ	ПЕР2	;ЕСЛИ СЦ НЕ 0
160F CD2001	CALL	J4CBH	; (B,H,L) -НОРМАЛИЗОВАННОЕ ЧИСЛО
1612 F1	POP	PSW	;ВОССТАНОВЛЕНИЕ ОСТАТКА Q(6)
1613 D21A16	JNC	ПЕР3	;ЕСЛИ ОСТАТОК Q(6)=0
1616 3E05	MVI	A,05	; (A) -ДЕСЯТИЧНЫЙ ОСТАТОК
1618 B5	ORA	L	
1619 6F	MOV	L,A	;ЗАПИСЬ ОСТАТКА В ЧАСТНОЕ
161A 7A	ПЕР3: MOV	A,D	;КОРРЕКЦИЯ ПОРЯДКА
161B C601	ADI	01	
161D 27	DAA		
161E 57	MOV	D,A	
161F C32316	JMP	ПЕР2+1	
	;ПРОВЕРКА КОНЦА ЦИКЛА 1		
1622 F1	ПЕР2: POP	PSW	;БАЛАНС СТЕКА
1623 1D	DCR	E	
1624 C2E015	JNZ	ЦИКЛ1	;ЗАЦИКЛИВАНИЕ 1
	;ВСТАВКА ЗНАКА ДЕСЯТИЧНОГО ПОРЯДКА		
1627 AF	XRA	A	
1628 B2	ADD	D	; (A) -ПОРЯДОК
1629 C8	RZ		;ДЕСЯТИЧНЫЙ ПОРЯДОК=0
162A 3E80	MVI	A,80H	;ЗНАК "-"
162C B2	ORA	D	
162D 57	MOV	D,A	
162E C9	RET		;ДЕСЯТИЧНЫЙ ПОРЯДОК < 0
0000	END		

Программа осуществляет деление методом последовательного выделения десятичных цифр делимого, начиная со старшей цифры, и деления их на 2 путем сдвига выделенной цифры на один двоичный разряд вправо. При этом формируются десятичная цифра частного и остаток от деления, равный 0 или 1. Этот остаток умножается на  $10_{10} = 1010_2$  и в сумме со следующей цифрой делимого служит основой для получения после очередного деления

следующей цифры частного. За 6 циклов сдвига формируется шестиразрядное десятичное частное от деления мантиссы на 2 и одноразрядный десятичный остаток. Далее частное нормализуется влево и производится коррекция десятичного порядка. Этот процесс повторяется  $2^{|m_2 - 40|}$  раз, после чего деление заканчивается.

Программа ПДП32 после выполнения умножения или деления десятичной мантиссы на двоичную характеристику округляет шестизначную мантиссу до четырехзначной по симметричному способу, устраняет переполнение мантиссы, возможное после ее округления, и записывает результат в память. Тестовые данные для программы ПДП32 можно заимствовать из табл. 3.3.



## 4 ПРОГРАММЫ ВЫЧИСЛЕНИЯ ЭЛЕМЕНТАРНЫХ ФУНКЦИЙ

### 4.1. ОБЩИЕ СВЕДЕНИЯ

Эта глава посвящена применению арифметических программ, описанных в предыдущих трех главах, для решения более сложных задач численной обработки данных, к которым относятся, в частности, задачи вычисления некоторых основных элементарных функций и факториала. Существует много различных методов вычисления элементарных функций, использующих представления их в виде степенных рядов, цепных дробей, итерационных процессов, разложений по ортогональным многочленам, рациональных приближений, бесконечных произведений и др. [7, 9, 14, 27, 45, 48, 51, 67, 70, 72]. Применение того или иного метода зависит от ряда условий, в том числе от требуемой точности, допустимых затрат памяти и времени выполнения программ. Мы ограничимся основными вопросами вычисления элементарных функций в рамках их наиболее простого и универсального представления в виде степенных рядов Тейлора (исключением будет являться итерационный процесс вычисления квадратного корня).

Рассмотрим классификацию элементарных функций. Самые простые элементарные функции принадлежат к классу целых *рациональных* функций, который в общем виде можно представить полиномом (многочленом)  $n$ -й степени:

$$y = F(x) = \sum_{i=0}^n a_i x^i,$$

где  $n$  — натуральное число или нуль;  $a_i$ ,  $x$  — действительные числа. В рамках класса выделяют обычно функции: *постоянную*  $y = a$ ; *линейную*  $y = ax + b$ ; *квадратичную*  $y = ax^2 + bx + c$ , а также функции  $n$ -й степени. Важный частный случай  $y = x^n$  задает *степенную* функцию с целочисленным показателем. Данный класс обра-

зуется в результате выполнения конечного числа сложений (вычитаний) и умножений двух простых функций: постоянной  $y = a$  и линейной  $y = x$ .

Добавление к классу целых рациональных функций операции деления полиномов порождает класс *дробно-рациональных* функций:

$$y = \frac{F(x)}{E(x)} = \frac{a_n x^n + \dots + a_1 x^1 + a_0}{b_n x^n + \dots + b_1 x^1 + b_0}, \quad E(x) \neq 0.$$

В рамках класса выделяют функции: *обратной пропорциональности*  $y = a/x$ , *дробно-линейную*  $y = (ax + b)/(cx + d)$  и *нелинейную дробно-рациональную*  $y = a + b/x + c/x^2$ . Сложная функция, т. е. вида  $y = F[\psi(x)]$ , составленная из дробно-рациональных функций, также принадлежит к этому классу.

Добавление к классу целых или дробно-рациональных функций операции отыскания *обратной* функции, т. е.  $x = \Phi(y)$  для  $y = F(x)$ , порождает класс *иррациональных* функций. Типичными представителями этого класса являются функция *извлечения корня*  $y = x^{1/n} = \sqrt[n]{x}$  (обратная для степенной функции  $Y = x^n$ ) и *степенная* функция с *рациональным показателем*  $y = x^k$ , где  $k = m/n$  ( $m, n$  — взаимно простые числа). Рассмотренные три класса функций в целом образуют более широкий класс *алгебраических* функций.

Функции, не являющиеся алгебраическими, принадлежат к классу *трансцендентных*. К ним относятся: *показательная*  $y = a^x (a \geq 0)$ , *логарифмическая*  $y = \log_a x$ , *тригонометрические*, *гиперболические* и обратные к ним функции. Перечисленные алгебраические и трансцендентные функции образуют множество основных *элементарных* функций — аналитически заданных в виде единых формул функций, которые получены из основных элементарных с помощью конечного числа алгебраических операций и операций образования сложных функций, причем сами операции, их число и порядок выполнения не зависят от значений аргумента. Все остальные функции называют *неэлементарными*. Если функция неэлементарна, то число операций над аргументом, либо сами операции, либо те и другие изменяются в зависимости от значения аргумента. Примером неэлементарной функции является *факториал*:  $y = n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n$ . Здесь количество операций зависит от значения аргумента.

Вычисление рациональных и дробно-рациональных

алгебраических функций осуществляется на основе точных аналитических записей этих функций, определяющих количество, тип и последовательность выполнения арифметических операций. Для минимизации вычислительных ресурсов аналитическая запись функции может быть эквивалентно преобразована, но в любом случае она остается точной в том смысле, что если точно заданы значения аргумента, точно выполнены арифметические операции, то гарантированно получается точное значение функции в виде рационального числа. Аналогичный способ вычисления неприменим в случае трансцендентных функций, так как их аналитическая запись, например  $y = \sin x$ , не позволяет непосредственно вычислять значение функции по значению аргумента с помощью арифметических операций. Трансцендентные функции определяются, как правило, приближенно на основе таблиц соответствующей точности [8, 58]. Программная реализация табличного вычисления этих функций очень громоздка, поэтому основной способ их вычисления заключается в замене (аппроксимации) исходной трансцендентной функции некоторой другой, алгебраической функцией, близкой к исходной по своим значениям в определенном диапазоне изменения аргумента и вычисляемой достаточно экономичным способом с помощью арифметических операций.

Пусть  $F(x)$  — точная элементарная функция;  $S(x)$  — ее приближение (*аппроксимирующая функция*) на интервале  $x \in (a, b)$ . Заметим, что аппроксимация выполняется, как правило, не для всей области определения функции, а только для сравнительно небольшой ее части, что облегчает выбор подходящей функции  $S(x)$ . Для вычисления  $F(x)$  на ином интервале могут потребоваться либо другая аппроксимирующая функция, либо специальные формулы перерасчета значений функции с одного интервала ее задания на другой интервал. Замена функции  $F(x)$  на  $S(x)$  порождает *ошибку метода* вычисления  $r(x) = F(x) - S(x)$ . Для оценки точности такого приближения вводят некоторые числовые характеристики ошибки  $r(x)$  на  $(a, b)$  — *нормы функции  $r(x)$* . В качестве одной из таких норм используют *равномерную норму* — верхнюю границу, аналогичную граничной абсолютной ошибке  $r(x) \leq r$ , где  $r = \max |r(x)|$  на  $(a, b)$ . Относительная ошибка метода  $\delta(x) = r(x)/F(x)$ , а граничная  $\delta(x)_r = |r/S(x)|$ .

Элементарные функции принадлежат к классу функций, точно представимых *степенными рядами*:

$$F(x) = \sum_{i=0}^{\infty} a_i (x - x_0)^i, \quad (4.1)$$

где  $x_0$  — постоянная величина. Выражение (4.1) представляет собой разложение функции  $F(x)$  в степенной ряд по степеням  $(x - x_0)$ . При  $x_0 = 0$  ряд (4.1) преобразуется к более простому виду:

$$F(x) = \sum_{i=0}^{\infty} a_i x^i. \quad (4.2)$$

Ряд вида (4.2) всегда сходится при  $x = 0$ , т. е. имеет конечный предел своих частичных сумм. Возможны три случая сходимости этого ряда в других точках: 1) ряд расходится во всех точках, кроме  $x = 0$ ; 2) ряд сходится во всех точках; 3) ряд сходится в одних точках и расходится в других. Последний вариант типичен. В общем случае область сходимости степенного ряда есть некоторый промежуток — *интервал сходимости*  $(-R, R)$ , симметричный относительно точки  $x = 0$ , где  $R$  — радиус сходимости. Если ряд сходится во всех точках, то  $R = \infty$ ; если сходится только в точке  $x = 0$ , то  $R = 0$ . Для ряда (4.1) интервал сходимости имеет вид  $(x_0 - R, x_0 + R)$ . Элементарные функции представляются степенными рядами, у которых радиус сходимости отличается от нуля, а сумма тождественно равна данной функции всюду внутри интервала сходимости.

Согласно известной теореме [14], если функция разлагается в степенной ряд, то разложение (4.1) единственно и ряд совпадает с *рядом Тейлора*, расположенным по степеням  $(x - x_0)$ :

$$F(x) = \sum_{i=0}^{\infty} \frac{F^{(i)}(x_0)}{i!} (x - x_0)^i, \quad (4.3)$$

где  $F^{(i)}(x_0)$  — производная  $i$ -го порядка функции  $F(x)$  в точке  $x = x_0$ . В формуле (4.3) коэффициенты  $a_i$  степенного ряда (4.1) определены через значения функции  $F(x)$  и ее производных в начальной точке  $x = x_0$ . Обрывая ряд (4.3) на  $n$ -м члене, получаем полином  $S_n(x)$   $n$ -й степени:

$$S_n(x) = \sum_{i=0}^n \frac{F^{(i)}(x_0)}{i!} (x - x_0)^i, \quad (4.4)$$

который позволяет вычислять значения функции  $F(x)$  с определенной точностью, т. е.  $F(x) \approx S_n(x)$ . Ошибка метода  $r_n(x) = F(x) - S_n(x)$ , а  $r_n(x)$  можно рассматривать как дополнительный член ряда:  $F(x) = S_n(x) + r_n(x)$ .

Если функция  $F(x)$  имеет производную  $(n+1)$ -го порядка в некоторой промежуточной точке  $C \in (x, x_0)$ , то дополнительный член можно представить в виде остаточного члена в форме Лагранжа:

$$r_n(x) = \frac{F^{(n+1)}(C)}{(n+1)!} (x - x_0)^{n+1},$$

который позволяет оценивать *сходимость ряда* и погрешность метода.

Для сходимости ряда  $S_n(x)$  при  $R > 0$  в интервале  $(x_0 - R, x_0 + R)$  к функции  $F(x)$  необходимо и достаточно, чтобы  $\lim_{n \rightarrow \infty} r_n(x) = 0$  при  $n \rightarrow \infty$  в указанном интервале. При сходимости ряда ошибка  $r_n(x)$  неограниченно уменьшается по абсолютному значению с увеличением числа членов. Число членов, обеспечивающее требуемую точность, существенно зависит от того, как велико расстояние  $|x - x_0|$  от начальной точки  $x_0$  до точки  $x$ . Чем больше это расстояние, тем больше членов приходится брать.

В общем случае *полином Тейлора* (4.4) пригоден для вычисления  $F(x)$  лишь на ограниченном расстоянии от начальной точки. Поэтому при использовании  $S_n(x)$  для вычисления конкретной функции необходимо четко уяснить ответы на следующие вопросы: пригоден ли  $S_n(x)$  для вычисления  $F(x)$  на данном расстоянии  $|x - x_0|$  от начальной точки  $x_0$ , и, если пригоден, то сколько членов надо брать для достижения требуемой точности? Ответы на эти вопросы дает исследование остаточного члена в форме Лагранжа. Однако его далеко не всегда удается получить в аналитическом виде. Для знакопередающегося сходящегося ряда оценка остатка может быть выполнена по *признаку Лейбница*: остаток имеет знак своего первого члена, и абсолютная величина остатка меньше абсолютной величины первого его члена [45].

В качестве иллюстрации применения полинома Тейлора вычислим показательную функцию  $y = e^x$  [14]. Все ее производные совпадают с самой функцией:  $y^{(n)} = e^x$ . При-

мем точку  $x_0 = 0$  за начальную. Тогда  $e^0 = 1$  и  $y^{(n)} = 1$ . Следовательно, функция представляется полиномом Тейлора [с учетом выражения (4.4)] вида

$$e^x = \sum_{i=0}^n \frac{1}{i!} x^i = 1 + \frac{1}{1!} x + \frac{1}{2!} x^2 + \dots + \frac{1}{n!} x^n. \quad (4.5)$$

Остаточный член в форме Лагранжа имеет вид

$$r_n(x) = \frac{e^c}{(n+1)!} x^{n+1}, \quad (4.6)$$

где  $0 \leq c \leq x$ ;  $e^0 = 1 \leq e^c \leq e^x$ . Определим пригодность выражения (4.5) для вычисления  $y = e^x$  в точке  $x = 1/2$ . Так как  $2 < e < 3$ , то  $e^{1/2} < 2$  и  $e^c < 2$  ( $0 \leq c \leq 1/2$ ). С учетом этого формулу (4.6) можно записать так:

$$|r_n(x)| < r = \frac{2}{(n+1)!} \left(\frac{1}{2}\right)^{n+1} = \frac{1}{(n+1)! 2^n}. \quad (4.7)$$

Величина  $r \rightarrow 0$  при  $n \rightarrow \infty$ . Следовательно, многочлен (4.5) пригоден для вычисления  $y = e^{1/2}$  с любой точностью. Пусть необходимо обеспечить точность суммы до 4-го десятичного знака, т. е.  $\Delta = 0,5 \cdot 10^{-4}$ . Подставляя в неравенство (4.7) последовательно значения  $n = 1, 2, \dots$ , получим следующие значения  $r$ :  $r_1 = 1/2! \cdot 2 = 1/4 = 0,25$ ;  $r_2 = 1/3! \cdot 2^2 = 1/24 \approx 0,4 \cdot 10^{-1}$ ;  $r_3 = 1/4! \cdot 2^3 = 1/192 \approx 0,5 \cdot 10^{-2}$ ;  $r_4 = 1/5! \cdot 2^4 = 1/1920 \approx 0,5 \cdot 10^{-3}$ ;  $r_5 = 1/6! \cdot 2^5 = 1/23040 \approx 0,4 \cdot 10^{-4}$ . Для обеспечения заданной точности с запасом достаточно взять первые 6 членов выражения (4.5). Аналогично определим пригодность выражения (4.5) для вычисления  $y = e^x$  в точке  $x = 1$ . Поскольку  $1 = e^0 \leq e^c \leq e^1 < 3$ , то  $r(x) < r = 3/(n+1)!$  и  $r \rightarrow 0$  при  $n \rightarrow \infty$ . Значит, и в этой точке возможно вычисление функции  $y = e$  с любой точностью. Для обеспечения той же точности  $\Delta = 0,5 \cdot 10^{-4}$  необходимо взять уже 8 членов. На рис. 4.1 изображены графики функции  $y = e^x$  и ее многочленов Тейлора. Из рисунка видно, что с ростом числа членов аппроксимирующего полинома растет его «близость» к исходной функции на большем участке графика.

Полином Тейлора дает наилучшее приближение вычисляемой функции в малой окрестности точки  $x = x_0$ , но его погрешность быстро возрастает по мере удаления от

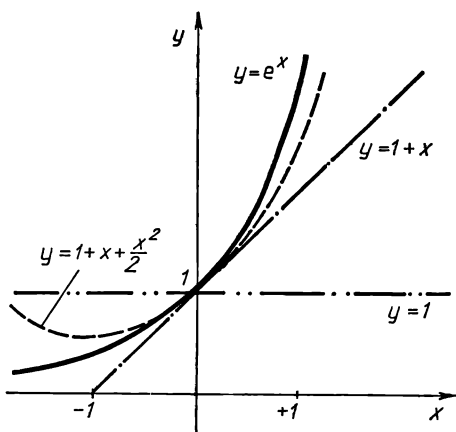


Рис. 4.1. Графики показательной функции и ее многочленов Тейлора

этой точки, что требует для обеспечения заданной точности существенного увеличения числа членов полинома и соответственно затрат памяти и времени на его программное вычисление. Поэтому данный метод аппроксимации элементарных функций применим при относительно невысоких требованиях к точности и диапазону аргументов вычисляемых функций. При более высоких требованиях необходимо использовать иные способы приближений функции, в частности наилучшие полиномиальные приближения (в смысле наименьшей нормы), обеспечивающие более быструю и равномерную сходимость на заданном интервале, чем полином Тейлора. Однако рассмотрение этих и других методов выходит за рамки книги.

Далее будут приведены конкретные алгоритмы и программы вычисления некоторых основных элементарных функций и факториала. Все входные, промежуточные и результирующие данные представлены в коротком формате двоичного числа с плавающей запятой (см. рис. 2.3, а). Программы используют подпрограммы арифметики с плавающей запятой, рассмотренные в гл. 2. Двоичные значения входных и десятичные значения выходных тестовых данных получены с помощью подпрограмм ППЗ10 и ПДПЗ2, описанных в гл. 3.

## 4.2. ОБРАТНАЯ ПРОПОРЦИОНАЛЬНОСТЬ

Дробно-рациональная функция обратной пропорциональности  $y = a/x$  ( $a \neq 0$ ), в частности  $y = 1/x$ , определена на множестве всех действительных ненулевых чисел. Поскольку функция является нечетной, ее график симметричен относительно начала координат и содержит

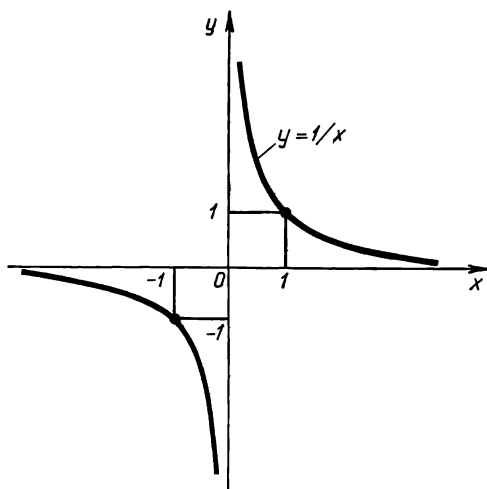


Рис. 4.2. График функции обратной пропорциональности

две ветви, расположенные в первом и третьем квадрантах декартовых координат, если  $a > 0$ , и во втором и четвертом квадрантах, если  $a < 0$  (рис. 4.2). График функции представляет собой гиперболу, асимптотами которой являются оси координат  $x$  и  $y$ .

Программа ОБРАТ вычисляет функцию  $y = 1/x$  путем деления единицы в формате числа с плавающей запятой  $1_{10} = 418000_{16}$  на значение аргумента:

1800		ORG	1800H
12C0	ДПЗЗ	SET	12C0H
0180	ЕДПЗ	SET	180H

ОБРАТ:

```

;*****
;ПОДПРОГРАММА ВЫЧИСЛЕНИЯ ОБРАТНОЙ ВЕЛИЧИНЫ Y(X)=1/X.
;ВХОДНЫЕ ПАРАМЕТРЫ: (H,L)-АДРЕС АРГУМЕНТА X, ПРЕДСТАВЛЕН-
;НОГО В ФОРМАТЕ 3-БАЙТНОГО ДВОИЧНОГО ЧИСЛА В ДОПОЛНИ-
;ТЕЛЬНОМ КОДЕ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ, (B,C)-АДРЕС ФУНКЦИИ
;Y(X), ПРЕДСТАВЛЯЕМОЙ В АНАЛОГИЧНОМ ФОРМАТЕ. Выходной па-

```



```

;РАМЕТР:СУ=1-ПРИЗНАК ПЕРЕПОЛНЕНИЯ ИЛИ АНТИПЕРЕПОЛНЕНИЯ
;ПОРЯДКА РЕЗУЛЬТАТА,ПРИЗНАК ДЕЛЕНИЯ НА АРГУМЕНТ Х=0.ИС-
;ПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ,СОХРАНЯЮТСЯ (Н,Л), (В,С),ГЛУБИ-
;НА СТЕКА-10.ИСПОЛЬЗУЮТСЯ ПОДПРОГРАММЫ:*ДАПЗЗ*,*КОМЗ*,
;*ДДФ17*,*ПМА2*,*ЕДПЗ*.
;ОЦЕНКА:ДЛИНА-7 БАЙТ (+178 БАЙТ ПОДПРОГРАММ),ВРЕМЯ-НЕ
;БОЛЕЕ 2883 ТАКТОВ (С УЧЕТОМ ПОДПРОГРАММ).
;*****
;ЗАНЕСЕНИЕ "1" В ФОРМЕ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ В РЕЗУЛЬТАТ
1800 CD8001      CALL    ЕДПЗ
;ДЕЛЕНИЕ "1" НА АРГУМЕНТ Х
1803 CDC012      CALL    ДАПЗЗ      ;(В,С)-АДРЕС ФУНКЦИИ
1806 C9          RET              ;СУ=1,ЕСЛИ ОШИБКА
0000            END

```

Формирование единицы с плавающей запятой в заданной области памяти выполняет программа ЕДПЗ:

```

0180                                ORG      180H
ЕДПЗ:
;*****
;ПОДПРОГРАММА ЗАПИСИ ЕДИНИЦЫ В ФОРМАТЕ С ПЛАВАЮЩЕЙ ЗАПЯ-
;ТОЙ В УКАЗАННУЮ ОБЛАСТЬ ПАМЯТИ.
;ВХОДНОЙ ПАРАМЕТР: (В,С)-АДРЕС ОБЛАСТИ ПАМЯТИ.ИСПОЛЬЗУЮТ-
;СЯ РЕГИСТРЫ (D,E), (H,L),СОХРАНЯЮТСЯ (H,L), (В,С).
;ОЦЕНКА:ДЛИНА-13 БАЙТ,ВРЕМЯ-59 ТАКТОВ.
;*****
0180 50          MOV      D,B
0181 59          MOV      E,C      ;(D,E)-АДРЕС ОБЛАСТИ ПАМЯТИ
0182 EB          XCHG      ;(H,L)-АДРЕС ОБЛАСТИ ПАМЯТИ
0183 3641        MVI      M,41H    ;ЗАПИСЬ ПОРЯДКА
0185 23          INX      H
0186 3680        MVI      M,80H    ;ЗАПИСЬ СТБ МАНТИССЫ
0188 23          INX      H
0189 3600        MVI      M,00H    ;ЗАПИСЬ МЛБ МАНТИССЫ
018B EB          XCHG      ;ВОССТАНОВЛЕНИЕ (H,L)
018C C9          RET
0000            END

```

В случае переполнения или антипереполнения порядка результата, а также при попытке деления на нулевой аргумент в программе ОБРАТ устанавливается признак переноса  $СУ=1$ . Тестовые данные программы приведены в табл. 4.1.

Табл. 4.1. Тестовые данные  $y=1/x$

$x$		$y = 1/x$	
$A_{10}$	$A_{16}$	$A_{10}$	$A_{16}$
1	2	3	4
$0,2468 \cdot 10^{-10}$	1DD916	$0,4052 \cdot 10^{+11}$	6496F1
$0,8642 \cdot 10^{-5}$	3090FC	$0,1157 \cdot 10^{+6}$	51E202

1	2	3	4
$0,1379 \cdot 10^{-1}$	3AEIEE	$0,7252 \cdot 10^{+2}$	479109
$0,1379 \cdot 10^{+1}$	41B082	$0,7252 \cdot 10^0$	40B9A5
$0,1000 \cdot 10^{+18}$	79B1A0	$0,1000 \cdot 10^{-16}$	08B87A

### 4.3. СТЕПЕННАЯ ФУНКЦИЯ

Рациональная степенная функция  $y = x^n$ ,  $n \geq 2$ , определена на множестве всех действительных чисел и является четной либо нечетной соответственно при четном  $n = 2k$  или нечетном  $n = 2k + 1$  показателе. График четной функции симметричен относительно оси ординат, расположен выше оси абсцисс и касается ее в точке  $(x, y) = (0, 0)$  (рис. 4.3, а). Точка  $(0, 0)$  считается точкой  $n$ -кратного касания, или  $n$ -кратным нулем: в этой точке функция имеет минимум. График нечетной функции симметричен относительно начала координат (рис. 4.3, б). Точка  $(0, 0)$  является *точкой перегиба графика*. Графики степенной функции называют *параболами  $n$ -го порядка*, все они проходят через точки  $(0, 0)$  и  $(1, 1)$ . Графики степенной функции более общего вида  $y = ax^n$  образуются при  $a > 0$  растяжением ординат, а при  $a < 0$  — растяжением и зеркальным отображением относительно оси абсцисс.

Программа СТЕП вычисляет степенную функцию  $y = x^n$  «в лоб», т. е. путем  $n$ -кратного умножения аргумента  $x$ :

1860		ORG	1860H
1220	УДПЗЗ	SET	1220H
0180	ЕДПЗ	SET	180H

СТЕП:

```

;*****
;ПОДПРОГРАММА ВЫЧИСЛЕНИЯ СТЕПЕНИ Y(X)=X**N (N < 256).
;ВХОДНЫЕ ПАРАМЕТРЫ: (H,L) — АДРЕС АРГУМЕНТА X, ПРЕДСТАВЛЕН-
;НОГО В ФОРМАТЕ 3-БАЙТНОГО ДВОИЧНОГО ЧИСЛА В ДОПОЛНИ-
;ТЕЛЬНОМ КОДЕ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ, (A) — ПОКАЗАТЕЛЬ СТЕПЕНИ
;В ФОРМАТЕ ЦЕЛОЧИСЛЕННОГО ДВОИЧНОГО БЕЗЗНАКОВОГО ЧИСЛА,
;(B,C) — АДРЕС СТЕПЕНИ. ВЫХОДНОЙ ПАРАМЕТР: CY=1 — ПРИЗНАК ПЕ-
;РЕПОЛНЕНИЯ ИЛИ АНТИПЕРЕПОЛНЕНИЯ ПОРЯДКА СТЕПЕНИ. ИСПОЛЬ-
;ЗУЮТСЯ ВСЕ РЕГИСТРЫ, СОХРАНЯЮТСЯ (H,L), (B,C), (A), ГЛУБИНА
;СТЕКА — 20. ИСПОЛЬЗУЮТСЯ ПОДПРОГРАММЫ: *УДПЗЗ*, *КОМЗ*, *ЕДПЗ
;*ОБНЗ*, *УДФ17*, *НМАН2*, *ДОПВ*, *ДОПД*, *УЗ2Б*, *У24А*.
;ОЦЕНКА: ДЛИНА — 26 БАЙТ (+241 БАЙТ ПОДПРОГРАММ), ВРЕМЯ — НЕ
;БОЛЕЕ (135+2117*N) ТАКОВ (С УЧЕТОМ ПОДПРОГРАММ).
;*****
;ЗАНЕСЕНИЕ В РЕЗУЛЬТАТ "1" В ФОРМЕ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ
CALL    ЕДПЗ

```

1860 CDB001

CALL ЕДПЗ

	;ПРОВЕРКА ПОКАЗАТЕЛЯ СТЕПЕНИ НА НОЛЬ	
1863 B7	ORA	A ;ПРОЯВЛЕНИЕ ПОКАЗАТЕЛЯ
1864 C8	RZ	;ЕСЛИ ПОКАЗАТЕЛЬ=0
1865 F5	PUSH	PSW ;СОХРАНЕНИЕ ПОКАЗАТЕЛЯ
	;ЦИКЛ УМНОЖЕНИЯ АРГУМЕНТА НА ЧАСТИЧНОЕ ПРОИЗВЕДЕНИЕ	
1866 F5	ЦИКЛ: PUSH	PSW ;СОХРАНЕНИЕ СЧЕТЧИКА ЦИКЛОВ
1867 CD2012	CALL	УДПЗЗ ;(B,C)-АДРЕС ПРОИЗВЕДЕНИЯ
186A DA7618	JC	ПЕР1 ;ЕСЛИ ОШИБКА ПОРЯДКА
	;ПРОВЕРКА КОНЦА ЦИКЛА	
186D F1	POP	PSW ;ВОССТАНОВЛЕНИЕ СЧЕТЧИКА ЦИКЛОВ
186E 3D	DCR	A
186F C26618	JNZ	ЦИКЛ ;ЗАЦИКЛИВАНИЕ
1872 AF	XRA	A ;CY=0
1873 C37718	JMP	ПЕР2
	;ВОССТАНОВЛЕНИЕ РЕГИСТРОВ	
1876 D1	ПЕР1: POP	D ;БАЛАНС СТЕКА
1877 D1	ПЕР2: POP	D ;ВОССТАНОВЛЕНИЕ ПОКАЗАТЕЛЯ
1878 7A	MOV	A,D ;(A)-ПОКАЗАТЕЛЬ СТЕПЕНИ
1879 C9	RET	;CY=1,ЕСЛИ ОШИБКА
0000	END	

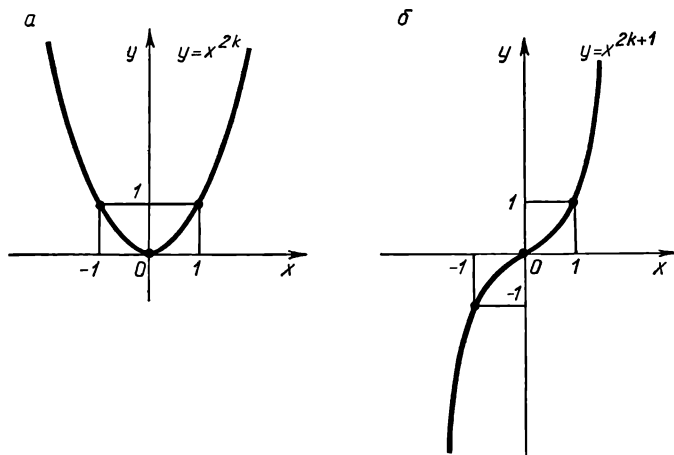


Рис. 4.3. Графики степенной функции

Такой подход уменьшает затраты памяти, но существенно увеличивает время решения, особенно при больших значениях  $n$ .

Число операций умножения при вычислении степенной функции можно уменьшить, если разложить показатель  $n$  на множители, кратные двум [1, 48]. Так, для  $n = 2k$  или  $n = 2k + 1$  имеем соответственно  $x^n = (x^k)^2$  или  $x^n = (x^k)^2 \cdot x$ , т. е. для вычисления  $x^n$  требуется не более  $k$  или  $(k + 1)$  операций умножения. Аналогичный подход

можно применить для вычисления  $x^k$  и т. д. Например, вычисление  $x^8 = [(x \cdot x)^2]^2$  требует всего трех операций умножения вместо семи. В общем случае целочисленный показатель  $n$  можно представить в виде двоичного полинома по схеме Горнера:  $n = (...((0 + a_{k-1}) \cdot 2 + a_{k-2}) \cdot 2 + ... + a_1) \cdot 2 + a_0$ . Тогда метод экономичного вычисления  $x^n$  определяется выражением

$$x^n = (...((x^0 \cdot x^{a_{k-1}})^2 \cdot x^{a_{k-2}})^2 \dots \cdot x^{a_1})^2 \cdot x^{a_0}, \quad (4.8)$$

где  $x^{a_i} = x$ , если  $a_i = 1$ , и  $x^{a_i} = 1$ , если  $a_i = 0$ ,  $i = 0, 1, 2, \dots, k - 1$ . Вычисление выражения (4.8) начинается с анализа старших двоичных разрядов показателя  $n$ . Если  $a_{k-1} = 1$ , вычисляется частичная степень  $(x^0 \cdot x^1)^2 = x^2$ ; если следующая цифра  $a_{k-2} = 1$ , полученная степень умножается на аргумент и возводится в квадрат:  $(x^2 \cdot x)^2 = x^6$ ; если же  $a_{k-2} = 0$ , сразу следует возведение в квадрат:  $(x^2)^2 = x^4$  и т. д. Например, вычисление  $x^7$  в соответствии с методом (4.8) имеет вид  $(7_{10} = 111_2) : ((x^2 \cdot x)^2 \cdot x = x^7$ . Применение метода (4.8) требует не более  $[\log_2 n]$  операций возведения в квадрат и не более такого же количества операций умножения на  $x$ .

Программа СТЕПА вычисляет степенную функцию по методу (4.8):

1890		ORG	1890H
1220	УДПЗЗ	SET	1220H
0180	ЕДПЗ	SET	180H

#### СТЕПА:

```

;*****
;ПОДПРОГРАММА ВЫЧИСЛЕНИЯ СТЕПЕНИ У(Х)=Х**N (N < 256).
;ВХОДНЫЕ ПАРАМЕТРЫ: (Н, L) - АДРЕС АРГУМЕНТА Х, ПРЕДСТАВЛЕН-
;НОГО В ФОРМАТЕ 3-БАЙТНОГО ДВОИЧНОГО ЧИСЛА В ДОПОЛНИ-
;ТЕЛЬНОМ КОДЕ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ, (А) - ПОКАЗАТЕЛЬ СТЕПЕНИ
;В ФОРМАТЕ ЦЕЛОЧИСЛЕННОГО ДВОИЧНОГО БЕЗЗНАКОВОГО ЧИСЛА,
;(В, С) - АДРЕС СТЕПЕНИ. ВЫХОДНОЙ ПАРАМЕТР: СУ=1 - ПРИЗНАК ПЕ-
;РЕПОЛНЕНИЯ ИЛИ АНТИПЕРЕПОЛНЕНИЯ ПОРЯДКА СТЕПЕНИ. ИСПОЛЬ-
;ЗУЮТСЯ ВСЕ РЕГИСТРЫ, СОХРАНЯЮТСЯ (Н, L), (В, С), (А), ГЛУБИНА
;СТЕКА=24. ИСПОЛЬЗУЮТСЯ ПОДПРОГРАММЫ: *УДПЗЗ*, *КОМЗ*, *ОВНЗ
;*УДФ17*, *НМАНЗ*, *ДОПВ*, *ДОПД*, *УЗЗБ*, *У24А*, *ЕДПЗ*.
;ОЦЕНКА: ДЛИНА=48 БАЙТ (+241 БАЙТ ПОДПРОГРАММ), ВРЕМЯ=НЕ
;БОЛЕЕ (142+4285*8) ТАКТОВ (С УЧЕТОМ ПОДПРОГРАММ).
;*****
;ЗАНЕСЕНИЕ "1" В ФОРМЕ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ В РЕЗУЛЬТАТ

```

1890	CD8001	CALL	ЕДПЗ	
		;ПРОВЕРКА ПОКАЗАТЕЛЯ СТЕПЕНИ НА НОЛЬ		
1893	B7	ORA	A	;ПРОЯВЛЕНИЕ ПОКАЗАТЕЛЯ
1894	C8	RZ		;ЕСЛИ ПОКАЗАТЕЛЬ=0
1895	F5	PUSH	PSW	;СОХРАНЕНИЕ ПОКАЗАТЕЛЯ

```

1896 1E08          MVI     E,B      ;(E)-СЧЕТЧИК ЦИКЛОВ
                   ;ЦИКЛ ВОЗВЕДЕНИЯ В КВАДРАТ ЧАСТИЧНОГО ПРОИЗВЕДЕНИЯ
1898 D5          ЦИКЛ: PUSH    D      ;СОХРАНЕНИЕ СЧЕТЧИКА
1899 F5          PUSH    PSW      ;СОХРАНЕНИЕ ПОКАЗАТЕЛЯ
189A E5          PUSH    H      ;СОХРАНЕНИЕ АДРЕСА АРГУМЕНТА
189B 60          MOV     H,B
189C 69          MOV     L,C      ;(H,L)-АДРЕС ПРОИЗВЕДЕНИЯ
189D CD2012       CALL    УДПЗ3
18A0 DAB818       JC      ПЕР1     ;ЕСЛИ ОШИБКА ПОРЯДКА
18A3 E1          POP     H      ;ВОССТАНОВЛЕНИЕ АДРЕСА АРГУМЕНТА
                   ;АНАЛИЗ ОЧЕРЕДНОГО СТАРШЕГО РАЗРЯДА ПОКАЗАТЕЛЯ СТЕПЕНИ
18A4 F1          POP     PSW      ;ВОССТАНОВЛЕНИЕ ПОКАЗАТЕЛЯ
18A5 07          RLC
18A6 D2B118       JNC     ПЕР2     ;ЕСЛИ РАЗРЯД=0
18A9 F5          PUSH    PSW      ;СОХРАНЕНИЕ ПОКАЗАТЕЛЯ
                   ;УМНОЖЕНИЕ ЧАСТИЧНОГО ПРОИЗВЕДЕНИЯ НА АРГУМЕНТ
18AA CD2012       CALL    УДПЗ3
18AD DAB818       JC      ПЕР3     ;ЕСЛИ ОШИБКА ПОРЯДКА
18B0 F1          POP     PSW      ;ВОССТАНОВЛЕНИЕ ПОКАЗАТЕЛЯ
                   ;ПРОВЕРКА КОНЦА ЦИКЛА
18B1 D1          ПЕР2: POP     D      ;ВОССТАНОВЛЕНИЕ СЧЕТЧИКА ЦИКЛА
18B2 1D          DCR     E
18B3 C29818       JNZ     ЦИКЛ    ;ЗАЦИКЛИВАНИЕ
18B6 AF          XRA     A      ;CY=0
18B7 C3BD18       JMP     ПЕР4
                   ;ВОССТАНОВЛЕНИЕ РЕГИСТРОВ
18BA E1          ПЕР1: POP     H      ;БАЛАНС СТЕКА
18BB D1          ПЕР3: POP     D      ;БАЛАНС СТЕКА
18BC D1          POP     D      ;БАЛАНС СТЕКА
18BD D1          ПЕР4: POP     D      ;ВОССТАНОВЛЕНИЕ ПОКАЗАТЕЛЯ
18BE 7A          MOV     A,D      ;(A)-ПОКАЗАТЕЛЬ СТЕПЕНИ
18BF C9          RET
0000             END

```

Программа выполняет анализ очередной двоичной цифры показателя степени путем сдвига его влево. При этом значение цифры совпадает с признаком переноса:  $CY = a_i$ . Сопоставляя программы СТЕП и СТЕПА, можно увидеть, что последняя требует несколько больших затрат памяти, но зато при больших  $n$  ( $n > 16$ ) она вычисляет степенную функцию быстрее, чем программа СТЕП. Заметим, что в обеих программах расчет времени их работы дан для наихудшего случая, когда все двоичные цифры показателя степени равны единице. Тестовые данные для обеих программ приведены в табл. 4.2.

Табл. 4.2. Тестовые данные  $y = x^n$

$y = x^n$		$y = x^n$	
$A_{10}$	$A_{16}$	$A_{10}$	$A_{16}$
1	2	3	4
$2^{10} = 0,1024 \cdot 10^4$	4B8000	$0,1 \approx 0,1000 \cdot 10^{-9}$	1FDBE0
$2^{40} = 0,1100 \cdot 10^{13}$	698000	$0,1^{15} \approx 0,1000 \cdot 10^{-14}$	0F9014

1	2	3	4
$10^5 = 0,1000 \cdot 10^6$	51C350	$0,5^{10} \approx 0,9766 \cdot 10$	378000
$-10^5 = -0,1000 \cdot 10^6$	D13CB0	$(0,1248)^7 \approx 0,4715 \cdot 10^{-6}$	2BFD2A
$10^{10} = 0,1000 \cdot 10^{11}$	629504	$(0,9731)^{12} \approx 0,7209 \cdot 10^0$	40B894

#### 4.4. ПОЛИНОМ

Полином  $y = \sum_{i=0}^n a_i \cdot x^i$  определен на множестве всех

действительных чисел  $x$ ,  $a_i$  и изображается на графике кривой без особых точек и без асимптот, причем кривая имеет не более  $n$  точек пересечения с осью абсцисс, не более  $n-1$  экстремумов и не более  $n-2$  точек перегиба. Конкретный вид кривой существенно зависит от значений  $n$ ,  $a_i$  и  $x$ . Например, если  $a_i = 1$  для всех  $i$ , то графики полиномов любой степени подобны графикам степенной функции, но с иным размещением относительно начала координат и с дополнительной деформацией ординат.

Программа ПОЛИН вычисляет значение полинома по схеме Горнера [см. формулу (3.2)]:

18E0	ORG	18E0H
1220	УДПЗ3 SET	1220H
1000	СДПЗ3 SET	1000H
00A3	ОБНЗ SET	0A3H
00B8	ПМЗ SET	0B8H

ПОЛИН:

```

;*****
;ПОДПРОГРАММА ВЫЧИСЛЕНИЯ ПОЛИНОМА СТЕПЕНИ N (N<256):
;P(X)=A(N)*X**(N)+A(N-1)*X**(N-1)+...+A(1)*X+A(0).
;ВХОДНЫЕ ПАРАМЕТРЫ:(D,E)-НАЧАЛЬНЫЙ АДРЕС МАССИВА КОЭФФИ-
;ЦИЕНТОВ ПОЛИНОМА A(0),A(1),...,A(N),ПРЕДСТАВЛЕННЫХ В
;ВИДЕ ДВОИЧНЫХ 3-БАЙТНЫХ ЧИСЕЛ В ДОПОЛНИТЕЛЬНОМ КОДЕ С
;ПЛАВАЮЩЕЙ ЗАПЯТОЙ,(H,L)-АДРЕС АРГУМЕНТА X,ПРЕДСТАВЛЕН-
;НОГО В АНАЛОГИЧНОМ ВИДЕ,(A)-ПОКАЗАТЕЛЬ СТЕПЕНИ N ПОЛИ-
;НОМА В ФОРМАТЕ ЦЕЛОЧИСЛЕННОГО БЕЗЗНАКОВОГО ДВОИЧНОГО
;ЧИСЛА.РЕЗУЛЬТАТ В ВИДЕ 3-БАЙТНОГО ДВОИЧНОГО ЧИСЛА С
;ПЛАВАЮЩЕЙ ЗАПЯТОЙ ПОМЕЩАЕТСЯ В ОБЛАСТЬ ПАМЯТИ "БУФЕР".
;ВЫХОДНОЙ ПАРАМЕТР:CY=1-ПРИЗНАК ПЕРЕПОЛНЕНИЯ ИЛИ АНТИ-
;ПЕРЕПОЛНЕНИЯ ПОРЯДКА РЕЗУЛЬТАТА.ИСПОЛЬЗУЕТСЯ ДОПОЛНИ-
;ТЕЛЬНО 3-БАЙТНАЯ РАБОЧАЯ ОБЛАСТЬ ПАМЯТИ "БУФД".
;ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ,СОХРАНЯЮТСЯ (D,E),(H,L),(A),
;ГЛУБИНА СТЕКА-26.ИСПОЛЬЗУЮТСЯ ПОДПРОГРАММЫ:УДПЗЗ*,
;КОМЗ*,ОБНЗ*,УДФ17*,НМАН2*,СДПЗЗ*,ОБМЗ*,ДМАН2*,
;ПМАН2*,ПМЗ*,ДОПВ*,ДОПД*,УЗБ2*,У24А*.
;ОЦЕНКА:ДЛИНА-81 БАЙТ (+475 БАЙТ ПОДПРОГРАММ),ВРЕМЯ-НЕ
;БОЛЕЕ (2934+4947*N) ТАКТОВ (С УЧЕТОМ ПОДПРОГРАММ).
;*****
;СОХРАНЕНИЕ РЕГИСТРОВ

```

18E0 D5	PUSH D	;СОХРАНЕНИЕ АДРЕСА КОЭФФИЦИЕНТОВ
18E1 E5	PUSH H	;СОХРАНЕНИЕ АДРЕСА АРГУМЕНТА

18E2 F5	PUSH	PSW	;СОХРАНЕНИЕ СТЕПЕНИ ПОЛИНОМА
18E3 F5	PUSH	PSW	;СОХРАНЕНИЕ СЧЕТЧИКА ЦИКЛОВ
	;ОБНУЛЕНИЕ ОБЛАСТИ РЕЗУЛЬТАТА (ЧАСТИЧНОЙ СУММЫ)		
18E4 E5	PUSH	H	
18E5 213119	LXI	H,БУФЕР	
18E8 CDA300	CALL	ОБНЗ	
18EB E1	POP	H	
	;ПРОВЕРКА КОНЦА ЦИКЛА		
18EC F1	ЦИКЛ: POP	PSW	;ВОССТАНОВЛЕНИЕ СЧЕТЧИКА ЦИКЛОВ
18ED B7	ORA	A	;ПРОЯВЛЕНИЕ СОСТОЯНИЯ СЧЕТЧИКА
18EE CA1E19	JZ	ПЕР1	;ЕСЛИ СЧЕТЧИК=0
18F1 3D	DCR	A	
18F2 F5	PUSH	PSW	;СОХРАНЕНИЕ СЧЕТЧИКА ЦИКЛОВ
	;АДРЕСАЦИЯ ОЧЕРЕДНОГО КОЭФФИЦИЕНТА В МАССИВЕ		
18F3 D5	PUSH	D	;СОХРАНЕНИЕ АДРЕСА МАССИВА
18F4 E5	PUSH	H	;СОХРАНЕНИЕ АДРЕСА АРГУМЕНТА
18F5 3C	INR	A	
18F6 0600	MVI	B,0	
18F8 4F	MOV	C,A	; (B,C)-ИНДЕКС МАССИВА
18F9 EB	XCHG		; (H,L)-АДРЕС МАССИВА
18FA 09	DAD	B	
18FB 09	DAD	B	
18FC 09	DAD	B	; (H,L)-АДРЕС КОЭФФИЦИЕНТА
	;СЛОЖЕНИЕ КОЭФФИЦИЕНТА С ЧАСТИЧНОЙ СУММОЙ В "БУФЕРЕ"		
18FD 113419	LXI	D,БУФД	; (D,E)-АДРЕС "БУФД"
1900 CDB800	CALL	ПМЗ	;ПЕРЕМЕЩЕНИЕ СЛАГАЕМОГО В "БУФД"
1903 EB	XCHG		; (H,L)-АДРЕС "БУФД"
1904 013119	LXI	B,БУФЕР	; (B,C)-АДРЕС ЧАСТИЧНОЙ СУММЫ
1907 CD0010	CALL	САПЗЗ	;РЕЗУЛЬТАТ В "БУФЕРЕ"
190A DA1819	JC	ПЕР2	;ЕСЛИ ОШИБКА ПОРЯДКА
	;УМНОЖЕНИЕ ЧАСТИЧНОЙ СУММЫ НА АРГУМЕНТ		
190D E1	POP	H	; (H,L)-АДРЕС АРГУМЕНТА
190E CD2012	CALL	УАПЗЗ	;РЕЗУЛЬТАТ В "БУФЕРЕ"
1911 DA1919	JC	ПЕР3	;ЕСЛИ ОШИБКА ПОРЯДКА
1914 D1	POP	D	;ВОССТАНОВЛЕНИЕ АДРЕСА МАССИВА
1915 C3EC18	JMP	ЦИКЛ	;ЗАЦИКЛИВАНИЕ
	;ОКОНЧАНИЕ ПОДПРОГРАММЫ ПРИ ОШИБКЕ ПОРЯДКА		
1918 E1	ПЕР2: POP	H	;БАЛАНС СТЕКА
1919 D1	ПЕР3: POP	D	;БАЛАНС СТЕКА
191A D1	POP	D	;БАЛАНС СТЕКА
191B C32C19	JMP	ПЕР4	
	;СЛОЖЕНИЕ КОЭФФИЦИЕНТА A(0) С ЧАСТИЧНОЙ СУММОЙ		
191E 013119	ПЕР1: LXI	B,БУФЕР	; (B,C)-АДРЕС ЧАСТИЧНОЙ СУММЫ
1921 EB	XCHG		; (H,L)-АДРЕС МАССИВА
1922 113419	LXI	D,БУФД	; (D,E)-АДРЕС "БУФД"
1925 CDB800	CALL	ПМЗ	;ПЕРЕМЕЩЕНИЕ КОЭФФИЦИЕНТА В "БУФД"
1928 EB	XCHG		; (H,L)-АДРЕС "БУФД"
1929 CD0010	CALL	САПЗЗ	;РЕЗУЛЬТАТ В "БУФЕРЕ"
	;ВОССТАНОВЛЕНИЕ РЕГИСТРОВ		
192C C1	ПЕР4: POP	B	;ВОССТАНОВЛЕНИЕ ПОКАЗАТЕЛЯ
192D 78	MOV	A,B	; (A)-ПОКАЗАТЕЛЬ СТЕПЕНИ ПОЛИНОМА
192E E1	POP	H	;ВОССТАНОВЛЕНИЕ АДРЕСА АРГУМЕНТА
192F D1	POP	D	;ВОССТАНОВЛЕНИЕ АДРЕСА МАССИВА
1930 C9	RET		;CY=1,ЕСЛИ ОШИБКА ПОРЯДКА

	;ОБЛАСТЬ РЕЗУЛЬТАТА (ЧАСТИЧНОЙ СУММЫ)		
1931	БУФЕР:	DS	3 ;3 БАЙТА
1934	БУФД:	DS	3 ;3 БАЙТА
0000		END	

Показатель степени полинома  $n$  перед работой программы записывают в аккумулятор, коэффициенты полинома  $a_0, a_1, \dots, a_n$  в виде трехбайтных чисел с плавающей запятой размещают последовательно в области памяти, начальный адрес которой задается в регистровой паре (D, E), а значение аргумента — в область памяти по адресу, указываемому в регистровой паре (H, L). В процессе работы программы дополнительно используются две трехбайтные области памяти БУФЕР и БУФД: первая используется для временного хранения вычисляемой частичной суммы полинома, вторая — для временного хранения очередной цифры. При переполнении или антипереполнении порядка результата в программе устанавливается признак переноса  $CY=1$ . В этом случае результат ошибочен. Максимальное время работы программы рассчитано для наихудшего случая, когда все  $a_i \neq 0$ .

В качестве тестового примера выполним вычисление полинома  $P(x) = a_3x^3 + a_2x^2 + a_1x^1 + a_0 = 4x^3 + 3x^2 - 2x + 1$ . В табл. 4.3 приведены соответствующие тестовые данные, а на рис. 4.4 — график полинома.

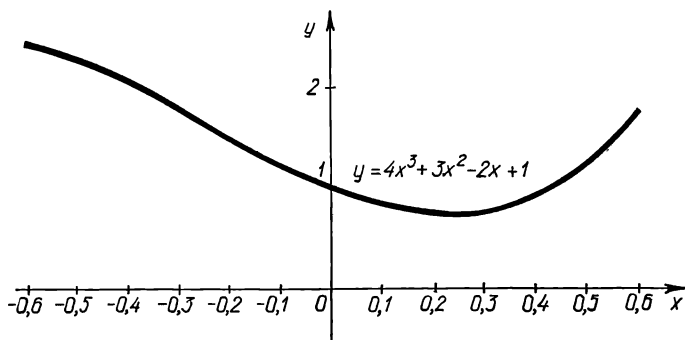


Рис. 4.4. График полинома



Табл. 4.3. Тестовые данные полинома  $P(x)$ 

$x$		$y = P(x)$	
$A_{10}$	$A_{16}$	$A_{10}$	$A_{16}$
0	000000	$0,1000 \cdot 10^1$	418000
0,2	3ECCCC	$0,7520 \cdot 10^0$	40C083
0,4	3FCCCC	$0,9360 \cdot 10^0$	40EF9D
0,6	40999A	$0,1744 \cdot 10^1$	41DF3B
-0,2	BE3334	$0,1488 \cdot 10^1$	41BE77
-0,4	BF3334	$0,2024 \cdot 10^1$	428189
-0,6	CO6666	$0,2416 \cdot 10^1$	429AA0

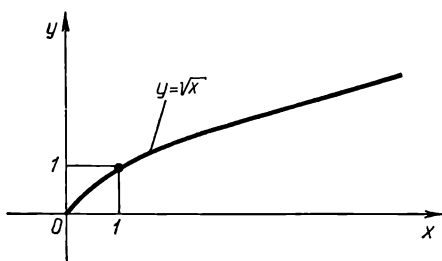


Рис. 4.5. График функции извлечения квадратного корня

## 4.5. КВАДРАТНЫЙ КОРЕНЬ

Иррациональная функция извлечения квадратного корня  $y = \sqrt{x}$  является обратной по отношению к функции возведения в квадрат  $y = x^2$  и определена для всех неотрицательных действительных чисел  $x$ . График функции представляет собой монотонно возрастающую кривую, расположенную вдоль оси абсцисс (рис. 4.5).

Известны различные методы вычисления квадратного корня, основанные на многочленных и рациональных приближениях, разложениях в биномиальный ряд и т. п. [45, 48, 70]. Чаще всего для вычислений используется итерационная формула Герона (частный случай формулы Ньютона):

$$y_{i+1} = 0,5(y_i + x/y_i), \quad (4.9)$$

где  $i = 0, 1, \dots$ ;  $y_0 \approx \sqrt{x}$  — начальное приближение, значение которого выбирается по условиям несложности вычисления (константа или линейная функция) и макси-

мальной близости к искомому значению корня, что позволяет уменьшить число итераций. Обычно это приближение выбирается на основе эмпирической формулы, зависящей от диапазона изменения аргумента. Так, например, при  $0,5 \leq x < 1$  наилучшим приближением считается  $y_0 \approx 0,5903x + 0,4173$  [48]. В простейшем случае в качестве значения  $y_0$  можно использовать непосредственно значение  $x$ , т. е.  $y_0 = x$ . Итерационный процесс вычисления корня заканчивается, когда значения двух соседних приближений  $y_i$  и  $y_{i+1}$  совпадают с требуемой точностью. Пусть, например, необходимо вычислить с четырьмя значащими цифрами  $y = \sqrt{28,34} \approx 5,324$ . Примем  $y_0 = 5$ . Тогда  $y_1 = 0,5(5 + 28,34/5) \approx 5,33$ ,  $y_2 = 0,5(5,33 + 28,34/5,33) \approx 5,324$ ,  $y_3 = 0,5(5,324 + 28,34/5,324) \approx 5,324$ . Так как  $y_2 \approx y_3$ , то процесс закончен. Сходимость итерационного процесса хорошая, и в каждом шаге число верных цифр примерно удваивается.

Рассмотренный метод реализован в программе вычисления квадратного корня ККОР:

1980		ORG	1980H
0088	ПМЗ	SET	008H
12C0	ДПЗЗ	SET	12C0H
1000	СПЗЗ	SET	1000H
19E0	СППЗЗ	SET	19E0H

ККОР:

```

;*****
;ПОДПРОГРАММА ВЫЧИСЛЕНИЯ КВАДРАТНОГО КОРНЯ Y=X**1/2.
;ВХОДНЫЕ ПАРАМЕТРЫ: (H,L)-АДРЕС АРГУМЕНТА X, ПРЕДСТАВЛЕН-
;НОГО В ФОРМЕ ТРЕХБАЙТНОГО ДВОИЧНОГО В ПРЯМОМ КОДЕ ЧИСЛА
;С ПЛАВАЮЩЕЙ ЗАПЯТОЙ, (B,C)-МАСКА ТОЧНОСТИ, ОПРЕДЕЛЯЮЩАЯ
;ТОЧНОСТЬ ВЫЧИСЛЕНИЯ ФУНКЦИИ Y (СОДЕРЖИТ НУЛИ В МЛАДШИХ
;НЕЗНАЧАЩИХ И ЕДИНИЦЫ В СТАРШИХ ЗНАЧАЩИХ РАЗРЯДХ). РЕ-
;ЗУЛЬТАТ ФОРМИРУЕТСЯ В ОБЛАСТИ ПАМЯТИ "БУФЕР". ВЫХОДНЫЕ
;ПАРАМЕТРЫ: (A)-КОЛИЧЕСТВО ВЫПОЛНЕННЫХ ИТЕРАЦИЙ В ПРОЦЕ-
;ССЕ ВЫЧИСЛЕНИЯ Y, CY=1-ПРИЗНАК ПЕРЕПОЛНЕНИЯ ИЛИ АНТИПЕРЕ-
;ПОЛНЕНИЯ ПОРЯДКА РЕЗУЛЬТАТА. ИСПОЛЗУЮТСЯ ВСЕ РЕГИСТРЫ,
;СОХРАНЯЮТСЯ (H,L), (B,C). ГЛУБИНА СТЕКА-X. ИСПОЛЗУЮТСЯ
;ПОДПРОГРАММЫ: *ПМЗ*, *ДПЗЗ*, *КОМЗ*, *ДДФ17*, *ПМА2*,
;*СППЗЗ*, *ОБМЗ*, *ДМАН2*, *ПМАН2*, *НМАН2*, *ДОПД*, *ДОПН*.
;ОЦЕНКА: ДЛИНА-82 БАЙТ (+424 БАЙТ ПОДПРОГРАММ), ВРЕМЯ-НЕ
;БОЛЕЕ (219+6350*K) ТАКОВ, ГДЕ K-КОЛИЧЕСТВО ИТЕРАЦИЙ.
;*****

```

1980 AF	XRA	A	; (A)-СЧЕТЧИК ИТЕРАЦИЙ=0
1981 F5	PUSH	PSW	; СОХРАНЕНИЕ СЧЕТЧИКА
1982 E5	PUSH	H	; СОХРАНЕНИЕ АДРЕСА X
1983 C5	PUSH	B	; СОХРАНЕНИЕ МАСКИ
	; ПЕРЕМЕЩЕНИЕ АРГУМЕНТА X В "БУФЕР" И "БУФД"		
1984 11D519	LXI	D, БУФД	
1987 CDB800	CALL	ПМЗ	; ПЕРЕМЕЩЕНИЕ В "БУФД"
198A 11D219	LXI	D, БУФЕР	

198D CDB800	CALL	ПМЗ	; ПЕРЕМЕЩЕНИЕ В "БУФЕР"
	; ВЫЧИСЛЕНИЕ НАЧАЛЬНОГО ПРИБЛИЖЕНИЯ $Y(0) = X/2$ В "БУФЕР"		
1990 EB	XCHG		; (H,L) - АДРЕС "БУФЕР"
1991 35	DCR	М	; УМЕНЬШЕНИЕ ПОРЯДКА: $X/2$
	; ДЕЛЕНИЕ АРГУМЕНТА X НА ТЕКУЩЕЕ ПРИБЛИЖЕНИЕ: $X/Y(I)$		
1992 01D519	CHKL: LXI	В, БУФД	
1995 CDC012	CALL	ДАПЗ3	; ЧАСТНОЕ В "БУФД"
1998 DACD19	JC	ПЕР1	; ЕСЛИ ОШИБКА ПОРЯДКА
	; ПЕРЕМЕЩЕНИЕ ТЕКУЩЕГО ПРИБЛИЖЕНИЯ ДЛЯ СЛОЖЕНИЯ В "БУФ"		
199B 11D819	LXI	Д, БУФ	
199E CDB800	CALL	ПМЗ	; ПЕРЕМЕЩЕНИЕ ИЗ "БУФЕР" В "БУФ"
19A1 EB	XCHG		; (H,L) - АДРЕС "БУФ"
	; ВЫЧИСЛЕНИЕ ПРИБЛИЖЕНИЯ: $Y(I+1) = 0,5 * (Y(I) + X/Y(I))$		
19A2 CD0010	CALL	САПЗ3	; $Y(I+1)$ В "БУФД"
19A5 DACD19	JC	ПЕР1	; ЕСЛИ ОШИБКА ПОРЯДКА
19A8 60	MOV	Н, В	
19A9 69	MOV	Л, С	; (H,L) - АДРЕС "БУФД"
19AA 35	DCR	М	; ДЕЛЕНИЕ $Y(I+1)/2$
	; СРАВНЕНИЕ ТЕКУЩЕГО ПРИБЛИЖЕНИЯ $Y(I)$ С ОЧЕРЕДНЫМ $Y(I+1)$		
19AB EB	XCHG		; (D,E) - АДРЕС "БУФД"
19AC C1	POP	В	; ВОССТАНОВЛЕНИЕ МАСКИ
19AD 21D219	LXI	Н, БУФЕР	
19B0 CDE019	CALL	СРПЗ3	; ВЫЧИСЛЕНИЕ ПРИЗНАКОВ $SY, Z$
19B3 CACE19	JZ	ПЕР2	; ЕСЛИ $Y(I+1) = Y(I)$
	; ПЕРЕМЕЩЕНИЕ ПРИБЛИЖЕНИЯ $Y(I+1)$ НА МЕСТО $Y(I)$		
19B6 EB	XCHG		; (H,L) - АДРЕС $Y(I+1)$
19B7 11D219	LXI	Д, БУФЕР	
19BA CDB800	CALL	ПМЗ	
	; ПЕРЕМЕЩЕНИЕ АРГУМЕНТА X В "БУФД"		
19BD EB	XCHG		; (D,E) - АДРЕС "БУФД"
19BE E1	POP	Н	; ВОССТАНОВЛЕНИЕ АДРЕСА X
19BF CDB800	CALL	ПМЗ	
	; СЧЕТ ИТЕРАЦИЙ		
19C2 F1	POP	PSW	; ВОССТАНОВЛЕНИЕ СЧЕТЧИКА ИТЕРАЦИЙ
19C3 3C	INR	А	; УВЕЛИЧЕНИЕ СЧЕТЧИКА
19C4 F5	PUSH	PSW	; СОХРАНЕНИЕ СЧЕТЧИКА
19C5 E5	PUSH	Н	; СОХРАНЕНИЕ АДРЕСА X
19C6 C5	PUSH	В	; СОХРАНЕНИЕ МАСКИ
19C7 21D219	LXI	Н, БУФЕР	; (H,L) - АДРЕС $Y(I+1)$
19CA C39219	JMP	ЦИКЛ	; ЗАЦИКЛИВАНИЕ
19CD C1	ПЕР1: POP	В	; ВОССТАНОВЛЕНИЕ МАСКИ
19CE E1	ПЕР2: POP	Н	; ВОССТАНОВЛЕНИЕ АДРЕСА X
19CF D1	POP	Д	; ВОССТАНОВЛЕНИЕ СЧЕТЧИКА
19D0 7A	MOV	А, Д	; (А) - СЧЕТЧИК ИТЕРАЦИЙ
19D1 C9	RET		
	; ОБЛАСТЬ ХРАНЕНИЯ РЕЗУЛЬТАТА		
19D2	БУФЕР: DS	3	; 3 БАЙТА
19D5	БУФД: DS	3	; 3 БАЙТА
19D8	БУФ: DS	3	; 3 БАЙТА
0000	END		

Эта программа носит исследовательский характер, поскольку, во-первых, область значений аргумента неограниченна и в качестве начального приближения выбрано

$y_0 = 0,5x$ ; во-вторых, имеется возможность задавать в качестве параметра точность вычисления результата (количество значащих цифр) и, в-третьих, можно проверять количество выполненных итераций, зависящих как от требуемой точности вычислений, так и от значений аргумента. В программе используются две дополнительные трехбайтные области памяти БУФД и БУФ для хранения промежуточных результатов.

Сравнение результатов двух соседних итераций выполняет подпрограмма СРПЗЗ:

19E0		ORG	19E0H
0058	ДОПД	SET	58H
0060	ДОПН	SET	60H

СРПЗЗ:

```

;*****
;ПОДПРОГРАММА СРАВНЕНИЯ С ЗАДАННОЙ ТОЧНОСТЬЮ ДВУХ ТРЕХ-
;БАЙТНЫХ В ДОПОЛНИТЕЛЬНОМ КОДЕ ЧИСЕЛ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ
;ФОРМАТА (8,16)=(ПОР;СТБ;МЛБ).
;ВХОДНЫЕ ПАРАМЕТРЫ:(Н,L)-АДРЕС ЧИСЛА 1,(D,E)-АДРЕС ЧИСЛА
;2,(B,C)-МАСКА ТОЧНОСТИ (СОДЕРЖИТ НУЛИ В МЛАДШИХ НЕЗНА-
;ЧАЧИХ И ЕДИНИЦЫ В СТАРШИХ ЗНАЧАЩИХ РАЗРЯДАХ).ВЫХОДНЫЕ
;ПАРАМЕТРЫ:CY=1-ПРИЗНАК ТОГО,ЧТО Ч1 < Ч2;CY=0 (Z=0)-
;ПРИЗНАК ТОГО,ЧТО Ч1 > Ч2;Z=1-ПРИЗНАК РАВЕНСТВА ЧИСЕЛ.
;ИСПОЛЬЗУЮТСЯ И СОХРАНЯЮТСЯ ВСЕ РЕГИСТРЫ,КРОМЕ А.ГЛУБИНА
;СТЕКА-10.ИСПОЛЬЗУЮТСЯ ПОДПРОГРАММЫ:**ДОПД,**ДОПН*.
;ОЦЕНКА:ДЛИНА-64 БАЙТА (+16 БАЙТ ПОДПРОГРАММ),ВРЕМЯ-НЕ
;БОЛЕЕ 437 ТАКТОВ (С УЧЕТОМ ПОДПРОГРАММ).
;*****
;СРАВНЕНИЕ ПОРЯДКОВ
19E0 C5      PUSH    B      ;СОХРАНЕНИЕ МАСКИ
19E1 1A      LDAX    D      ;(A)-БАЙТ ПОРЯДКА ПОР2
19E2 E67F    ANI      7FH   ;ИСКЛЮЧЕНИЕ ЗНАКА
19E4 4F      MOV     C,A     ;(C)-ПОРЯДОК ПОР2
19E5 7E      MOV     A,M     ;(A)-БАЙТ ПОРЯДКА ПОР1
19E6 E67F    ANI      7FH   ;ИСКЛЮЧЕНИЕ ЗНАКА
19E8 B9      CMP     C      ;СРАВНЕНИЕ ПОР1 И ПОР2
19E9 CAEE19  JZ       PER1   ;ЕСЛИ ПОР1=ПОР2
19EC C1      POP     B
19ED C9      RET          ;Z=0,ПОР1 НЕ= ПОР2
;ПРЕОБРАЗОВАНИЕ МАНТИСС В ПРЯМОЙ КОД
19EE C1      PER1: POP     B      ;ВОССТАНОВЛЕНИЕ МАСКИ
19EF E5      PUSH    H      ;СОХРАНЕНИЕ АДРЕСА 1
19F0 D5      PUSH    D      ;СОХРАНЕНИЕ АДРЕСА 2
19F1 C5      PUSH    B      ;СОХРАНЕНИЕ МАСКИ
19F2 D5      PUSH    D      ;СОХРАНЕНИЕ АДРЕСА 2
19F3 7E      MOV     A,M
19F4 17      RAL          ;CY-ЗНАК ЧИСЛА1
19F5 23      INX     H
19F6 56      MOV     D,M
19F7 23      INX     H
19F8 5E      MOV     E,M     ;(D,E)-СТБ,МЛБ МАНТИССЫ 1
19F9 DC5800  CC        ДОПД   ;ДОПОЛНЕНИЕ МАНТИССЫ 1
19FC C1      POP     B      ;ВОССТАНОВЛЕНИЕ АДРЕСА 2
19FD 0A      LDAX    B

```

19FE 17	RAL		#CY-ЗНАК ЧИСЛА 2
19FF 03	INX	B	
1A00 0A	LDAX	B	
1A01 67	MOV	H,A	
1A02 03	INX	B	
1A03 0A	LDAX	B	
1A04 6F	MOV	L,A	#(H,L)-СТБ,МЛБ МАНТИССЫ 2
1A05 DC6000	CC	ДППН	#ДОПОЛНЕНИЕ МАНТИССЫ 2
	#МАСКИРОВАНИЕ НЕЗНАЧАЩИХ РАЗРЯДОВ МАНТИССЫ 2		
1A08 C1	POP	B	#ВОССТАНОВЛЕНИЕ МАСКИ
1A09 79	MOV	A,C	#(A)-МАСКА НА МЛБ
1A0A A5	ANA	L	
1A0B 6F	MOV	L,A	
1A0C 7B	MOV	A,B	#(A)-МАСКА НА СТБ
1A0D A4	ANA	H	
1A0E 67	MOV	H,A	
	#МАСКИРОВАНИЕ НЕЗНАЧАЩИХ РАЗРЯДОВ МАНТИССЫ 1		
1A0F 79	MOV	A,C	#(A)-МАСКА НА МЛБ
1A10 A3	ANA	E	
1A11 5F	MOV	E,A	
1A12 7B	MOV	A,B	#(A)-МАСКА НА СТБ
1A13 A2	ANA	D	#(A)-СТБ1*(МАСКА)
	#СРАВНЕНИЕ МАНТИСС		
1A14 BC	CMP	H	#СРАВНЕНИЕ СТБ1 И СТБ2
1A15 CA1B1A	JZ	ПЕР2	#ЕСЛИ СТБ1=СТБ2
1A1B D1	POP	D	#ВОССТАНОВЛЕНИЕ АДРЕСА 2
1A19 E1	POP	H	#ВОССТАНОВЛЕНИЕ АДРЕСА 1
1A1A C9	RET		#Z=0,СТБ1 НЕ= СТБ2
1A1B 7B	ПЕР2: MOV	A,E	#(A)-МЛБ1*(МАСКА)
1A1C BD	CMP	L	
1A1D D1	POP	D	#ВОССТАНОВЛЕНИЕ АДРЕСА 2
1A1E E1	POP	H	#ВОССТАНОВЛЕНИЕ АДРЕСА 1
1A1F C9	RET		#Z=?,CY=?
0000	END		

Программа сравнивает два числа с плавающей запятой, причем сравнение мантисс выполняется лишь в случае равенства порядков чисел. Программа СРПЗЗ ориентирована на сравнение не только положительных чисел, которые используются в выражении (4.9), но и чисел в дополнительных кодах, поэтому в последнем случае значение мантиссы перед сравнением преобразуется в прямой код. Сам процесс сравнения мантисс производится с маскированием их младших разрядов, что позволяет управлять точностью получения результата и количеством выполняемых итераций.

Тестовые данные для программы ККОР при маске FFFF приведены в табл. 4.4, где  $k$  — количество итераций.

Табл. 4.4. Тестовые данные  $y = \sqrt{x}$ 

x		$y = \sqrt{x}$		k
$A_{10}$	$A_{16}$	$A_{10}$	$A_{16}$	
$0,2000 \cdot 10^0$	3ECCCC	$0,4472 \cdot 10^0$	3FE4F9	5
$0,2000 \cdot 10^1$	428000	$0,1414 \cdot 10^1$	41B505	3
$0,2000 \cdot 10^2$	459FFF	$0,4472 \cdot 10^1$	438F1B	4
$0,2000 \cdot 10^3$	48C7FF	$0,1414 \cdot 10^2$	44E246	6
$0,1000 \cdot 10^6$	51C34F	$0,3162 \cdot 10^3$	499E19	10

В случае маски F000, сохраняющей лишь две старшие значащие десятичные цифры результата, вычисление, например,  $y = \sqrt{0,2}$  требует двух итераций вместо пяти. Заметим, что искомый результат получается на предпоследней итерации, а последняя итерация лишь контролирует его правильность.

#### 4.6. ФАКТОРИАЛ

Программа ФАКТОР вычисляет факториал  $y = N!$  методом последовательного перемножения сомножителей:

```

1810          ORG      1810H
1220          УАПЗ3   SET      1220H
1000          САПЗ3   SET      1000H
00ВВ          ПМЗ     SET      0ВВН

```

ФАКТОР:

```

;*****
;ПОДПРОГРАММА ВЫЧИСЛЕНИЯ ФАКТОРИАЛА Y=N! (N < 21)
;ВХОДНЫЕ ПАРАМЕТРЫ: (В,С)–АДРЕС РЕЗУЛЬТАТА, ПРЕДСТАВЛЯЕМО-
;ГО В ФОРМАТЕ 3-БАЙТНОГО ДВОИЧНОГО ЧИСЛА В ДОПОЛНИТЕЛЬ-
;НОМ КОДЕ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ, (А)–ВЕЛИЧИНА N В ФОРМАТЕ
;ЦЕЛОГО ДВОИЧНОГО БЕЗЗНАКОВОГО ЧИСЛА. ВЫХОДНОЙ ПАРАМЕТР:
;СУ=1–ПРИЗНАК ПЕРЕПОЛНЕНИЯ ПОРЯДКА. ИСПОЛЬЗУЮТСЯ ВСЕ РЕ-
;ГИСТРЫ, СОХРАНЯЮТСЯ (В,С), (А), ГЛУБИНА СТЕКА–22. ИСПОЛЬ-
;ЗУЕТСЯ 3-БАЙТНАЯ КОНСТАНТА "ОДИН" И 3-БАЙТНАЯ РАБО-
;ЧАЯ ОБЛАСТЬ ПАМЯТИ "БУФЕР". ИСПОЛЬЗУЮТСЯ ПОДПРОГРАММЫ:
;*УАПЗ3*, *КОМЗ*, *ОБНЗ*, *УДФ17*, *НМАН2*, *САПЗ3*, *ПМЗ*,
;*ОБМЗ*, *ДМАН2*, *ПМАН2*, *ДОПВ*, *ДОПД*, *УЗЗБ*, *УЗ4А*.
;ОЦЕНКА: ДЛИНА–56 БАЙТ (+475 БАЙТ ПОДПРОГРАММ), ВРЕМЯ–НЕ
;БОЛЕЕ (350+4779*(N–1)) ТАКТОВ (С УЧЕТОМ ПОДПРОГРАММ).
;*****
;ЗАНЕСЕНИЕ В "БУФЕР" И ОБЛАСТЬ РЕЗУЛЬТАТА КОНСТАНТЫ "1"

```

```

1810 F5          PUSH   PSW      ;СОХРАНЕНИЕ N
1811 214918      LXI     H,ОДИН  ;(H,L)–АДРЕС КОНСТАНТЫ
1814 114C18      LXI     D,БУФЕР ;(D,E)–АДРЕС БУФЕРА
1817 CDВ800      CALL    ПМЗ     ;ПЕРЕМЕЩЕНИЕ "1" В "БУФЕР"
181A 50          MOV     D,B

```

181B 59	MOV	E,C	; (D,E) - АДРЕС РЕЗУЛЬТАТА
181C CDB800	CALL	PM3	; ПЕРЕМЕЩЕНИЕ "1" В РЕЗУЛЬТАТ
181F F1	POP	PSW	; ВОССТАНОВЛЕНИЕ N
	; ПРОВЕРКА: N=0 ?		
1820 B7	ORA	A	; ПРОЯВЛЕНИЕ N
1821 C8	RZ		; ЕСЛИ N=0, ТО 0!=1
1822 F5	PUSH	PSW	; СОХРАНЕНИЕ N
1823 F5	PUSH	PSW	; СОХРАНЕНИЕ СЧЕТЧИКА ЦИКЛА
	; ПРОВЕРКА КОНЦА ЦИКЛА		
1824 F1	ЦИКЛ: POP	PSW	; ВОССТАНОВЛЕНИЕ СЧЕТЧИКА ЦИКЛА
1825 3D	DCR	A	
1826 CA4618	JZ	PER1	; ЕСЛИ СЧЕТЧИК=0
1829 F5	PUSH	PSW	; СОХРАНЕНИЕ СЧЕТЧИКА
	; УВЕЛИЧЕНИЕ МНОЖИТЕЛЯ НА ЕДИНИЦУ		
182A E5	PUSH	H	; СОХРАНЕНИЕ АДРЕСА КОНСТАНТЫ
182B C5	PUSH	B	; СОХРАНЕНИЕ АДРЕСА РЕЗУЛЬТАТА
182C 014C18	LXI	B, БУФЕР	; (B,C) - АДРЕС БУФЕРА
182F CD0010	CALL	САПЗ3	
1832 DA4318	JC	PER2	; ЕСЛИ ОШИБКА ПОРЯДКА
1835 C1	POP	B	; ВОССТАНОВЛЕНИЕ АДРЕСА РЕЗУЛЬТАТА
	; УМНОЖЕНИЕ ЧАСТИЧНОГО ПРОИЗВЕДЕНИЯ НА МНОЖИТЕЛЬ		
1836 214C18	LXI	H, БУФЕР	; (H,L) - АДРЕС БУФЕРА
1839 CD2012	CALL	УАПЗ3	
183C DA4418	JC	PER3	; ЕСЛИ ОШИБКА ПОРЯДКА
183F E1	POP	H	; ВОССТАНОВЛЕНИЕ АДРЕСА КОНСТАНТЫ
1840 C32418	JMP	ЦИКЛ	; ЗАЦИКЛИВАНИЕ
	; ВОССТАНОВЛЕНИЕ РЕГИСТРОВ		
1843 C1	PER2: POP	B	; ВОССТАНОВЛЕНИЕ АДРЕСА РЕЗУЛЬТАТА
1844 E1	PER3: POP	H	; БАЛАНС СТЕКА
1845 D1	POP	D	; БАЛАНС СТЕКА
1846 D1	PER1: POP	D	; ВОССТАНОВЛЕНИЕ N
1847 7A	MOV	A,D	; (A) - ВЕЛИЧИНА N
1848 C9	RET		; CY=1, ЕСЛИ ОШИБКА
	; ОБЛАСТИ КОНСТАНТЫ И БУФЕРА		
1849 41	ОДИН: DB	41H	; ПОРЯДОК
184A 8000	DW	0080H	; МАНТИССА
184C	БУФЕР: DS	3	; 3 БАЙТА
0000	END		

Программа использует две дополнительные трехбайтные области памяти ОДИН и БУФЕР для хранения константы единицы в форме с плавающей запятой и временного запоминания промежуточного произведения. Поскольку формат результата ограничивает представление максимального числа (см. табл. 2.1), вычисление факториала ограничено значением  $N = 19$ . Программу ФАКТОР можно использовать для вычисления всех 19 значений факториала с последующим их табличным представлением, аналогичным представлению степеней в программе СТЕПА. Такой подход позволяет резко сократить время вычисления (поиска) факториала. Тестовые данные для программы ФАКТОР приведены в табл. 4.5.

Табл. 4.5. Тестовые данные  $y=N!$ 

$N$	$A_{10}$	$A_{16}$	$N$	$A_{10}$	$A_{16}$
1	$0,1000 \cdot 10^1$	418000	6	$0,7200 \cdot 10^3$	4AB400
2	$0,2000 \cdot 10^1$	428000	7	$0,5040 \cdot 10^4$	4D9D80
3	$0,6000 \cdot 10^1$	43C000	8	$0,4032 \cdot 10^5$	509D80
4	$0,2400 \cdot 10^2$	45C000	9	$0,3629 \cdot 10^6$	53B130
5	$0,1200 \cdot 10^3$	47EFFF	10	$0,3629 \cdot 10^7$	56DD7C

## 4.7. ПОКАЗАТЕЛЬНАЯ ФУНКЦИЯ

Показательной является функция вида  $y=a^x$ , где  $a > 0$  и  $a \neq 1$ . Функция определена на всем множестве действительных чисел и принимает только положительные значения, ограничена снизу, не имеет нулей и экстремумов, выпукла и непрерывна. При  $a > 1$  она монотонно возрастает, при  $a < 1$  монотонно убывает. График функции расположен выше оси абсцисс и пересекает ось ординат в точке  $y=1$  (рис. 4.6). Графики функций  $y=a^x$  и  $y=a^{-x}$  симметричны относительно оси ординат. Показательную функцию с основанием  $a$  можно свести к показательной функции с натуральным основанием  $e$ . Обозначим  $a=e^b$  ( $b \neq 0$ ). Тогда  $y=a^x=(e^b)^x=e^{bx}=\exp(bx)$ . Функцию полученного вида называют **экспоненциальной** или **экспонентой**. Взяв натуральный логарифм от обеих частей последнего равенства, получим  $b=\ln a$ . Таким образом, вычисление  $a^x$  можно свести к вычислению  $\exp(bx)$ , где  $b=\ln a$ , или к вычислению  $\exp z$ , где  $z=bx$ . График функции  $y=e^x$  расположен между графиками функций  $y=2^x$  и  $y=3^x$ .

Экспоненциальная функция  $y=e^x$ , как было показано в § 4.1, хорошо аппроксимируется рядом Тейлора (4.5). Для обеспечения точности приближения порядка  $0,5 \times 10^{-4}$  на сегменте  $[-1, +1]$  достаточно взять первые 8 членов ряда с коэффициентами 1,  $1/1! = 1$ ,  $1/2! = 0,5$ ,  $1/3! = 0,1667$ ,  $1/4! = 0,4167 \cdot 10^{-1}$ ,  $1/5! = 0,8333 \cdot 10^{-2}$ ,  $1/6! = 0,1389 \cdot 10^{-2}$  и  $1/7! = 0,1984 \cdot 10^{-3}$ . Этот метод вычисления экспоненты реализует программа ЭКСПО:

1950		ORG	1950H
18E0	ПОЛИН	SET	18E0H
1931	БУФЕР	SET	1931H

ЭКСПО:

```

*****
;ПОДПРОГРАММА ВЫЧИСЛЕНИЯ ЭКСПОНЕНТЫ Y=E**X (-1<X<=+1).
;ВХОДНЫЕ ПАРАМЕТРЫ:(H,L)-АДРЕС АРГУМЕНТА,ПРЕДСТАВЛЕННОГО

```



```

;В ФОРМЕ 3-БАЙТНОГО ДВОИЧНОГО ЧИСЛА В ДОПОЛНИТЕЛЬНОМ
;КОДЕ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ.РЕЗУЛЬТАТ ,ПРЕДСТАВЛЯЕМЫЙ В
;АНАЛОГИЧНОМ ВИДЕ,ПОМЕЩАЕТСЯ В ОБЛАСТЬ ПАМЯТИ "БУФЕР".
;ВЫХОДНОЙ ПАРАМЕТР:CY=1-ПРИЗНАК ПЕРЕПОЛНЕНИЯ ИЛИ АНТИПЕ-
;РЕПОЛНЕНИЯ ПОРЯДКА РЕЗУЛЬТАТА.ИСПОЛЬЗУЕТСЯ ДОПОЛНИТЕЛЬ-
;НАЯ РАБОЧАЯ ОБЛАСТЬ ПАМЯТИ "КОЭФ" ОБЪЕМОМ 8*3=24 БАЙТ
;ДЛЯ ХРАНЕНИЯ КОЭФФИЦИЕНТОВ СТЕПЕННОГО РЯДА.ИСПОЛЬЗУЮТСЯ
;ВСЕ РЕГИСТРЫ,СОХРАНЯЕТСЯ (H,L).ГЛУБИНА СТЕКА=26.ИСПОЛЬ-
;ЗУЮТСЯ ПОДПРОГРАММЫ:*ПОЛИН*,*УДПЗ3*,*КОМЗ*,*ОВНЗ*,
;*УДФ17*,*МНАН2*,*СППЗ3*,*ОЕМЗ*,*ДМАН2*,*ПМАН2*,*ПМЗ*,
;*ДОПВ*,*ДОПД*,*УЗ2Б*,*У24А*.
;ОЦЕНКА:ДЛИНА-9 БАЙТ (+556 БАЙТ ПОДПРОГРАММ),ВРЕМЯ-НЕ
;БОЛЕЕ 37 610 ТАКТОВ (С УЧЕТОМ ПОДПРОГРАММ).
;*****
;ПОДГОТОВКА РЕГИСТРОВ К ВЫЧИСЛЕНИЮ ПОЛИНОМА

```

```

1950 3E07      MVI    A,7          ;(A)-СТЕПЕНЬ ПОЛИНОМА=7
1952 115919    LXI    D,КОЭФ      ;(D,E)-АДРЕС КОЭФФИЦИЕНТОВ
;ВЫЧИСЛЕНИЕ ПОЛИНОМА
1955 CDE018    CALL   ПОЛИН
1958 C9        RET              ;CY=1,ЕСЛИ ОШИБКА ПОРЯДКА
;ОБЛАСТЬ КОЭФФИЦИЕНТОВ A(0),...,A(7)
1959 41      КОЭФ:  DB      41H      ;A(0)=1=41 80 00H
195A 8000    DW      0080H
195C 41      DB      41H      ;A(1)=1=41 80 00H
195D 8000    DW      0080H
195F 40      DB      40H      ;A(2)=0,5=40 80 00H
1960 8000    DW      0080H
1962 3E      DB      3EH      ;A(3)=0,1667=3E AA B4H
1963 AAB4    DW      0B4AAH
1965 3C      DB      3CH      ;A(4)=0,4167*10**(-1)=3C AA ADH
1966 AAAD    DW      0ADAAH
1968 3A      DB      3AH      ;A(5)=0,8333*10**(-2)=3A 88 87H
1969 8887    DW      8788H
196B 37      DB      37H      ;A(6)=0,1389*10**(-2)=37 B6 0EH
196C B60E    DW      0EB6H
196E 34      DB      34H      ;A(7)=0,1984*10**(-3)=34 D0 09H
196F D009    DW      09D0H
0000        END

```

Программа вычисляет экспоненту с помощью подпрограммы нахождения полинома ПОЛИН. Двоичные значения коэффициентов полинома получены предварительно посредством программы ППЗ10. Тестовые данные для программы ЭКСПО приведены в табл. 4.6.

Табл. 4.6. Тестовые данные  $y=e_x$

x		y = e <sub>x</sub>	
A <sub>10</sub>	A <sub>16</sub>	A <sub>10</sub>	A <sub>16</sub>
0,1000·10 <sup>0</sup>	3DCCCC	0,1105·10 <sup>1</sup>	418D76
0,3000·10 <sup>0</sup>	3F999A	0,1350·10 <sup>1</sup>	41ACC9
0,8000·10 <sup>0</sup>	40CCCC	0,2226·10 <sup>1</sup>	428E70
0,1000·10 <sup>1</sup>	418000	0,2718·10 <sup>1</sup>	42ADF9
-0,8000·10 <sup>0</sup>	C03334	0,4493·10 <sup>0</sup>	3FE60C

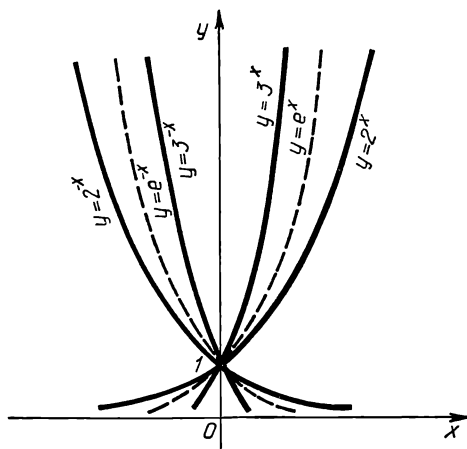


Рис. 4.6. Графики показательной функции

Для вычисления экспоненты на более широком интервале значений аргумента значение  $x$  можно представить в виде  $x = n + z$ , где  $n$  — целая часть аргумента;  $z$  — дробная. Тогда вычисление  $e^x$  сводится к вычислению  $e^z$  на интервале  $(-1, 1)$ , а также степенной функции  $e^n \approx (2,718)^n$ .

## 4.8. ТРИГОНОМЕТРИЧЕСКИЕ ФУНКЦИИ

Тригонометрические функции определяют трансцендентную зависимость, т. е. зависимость, которая не может быть точно выражена в виде конечного алгебраического уравнения, между угловыми  $\alpha$  и прямоугольными  $(x, y)$  координатами некоторой точки  $C$  окружности единичного радиуса  $x^2 + y^2 = 1$  (рис. 4.7). Ордината точки определяет функцию синуса  $y = \sin \alpha$ , абсцисса — функцию косинуса  $x = \cos \alpha$ , отношение координат  $y/x$ , или, что то же самое, величина отрезка касательной  $BD$ , — функцию тангенса  $\operatorname{tg} \alpha = \sin \alpha / \cos \alpha$ , обратное отношение координат  $x/y$ , или величина отрезка касательной  $FD$ , — функцию котангенса  $\operatorname{ctg} \alpha = \cos \alpha / \sin \alpha$  (на рис. 4.7  $\alpha = 45^\circ$  и  $\operatorname{tg} 45^\circ = \operatorname{ctg} 45^\circ = 1$ ). Аналогично тригонометрические функции определяются для любого угла  $\alpha'$ , отличающегося от острого угла  $\alpha$ . Аргумент  $\alpha$  тригонометрической функции представляет собой действительное число, которое можно понимать не только как величину централь-

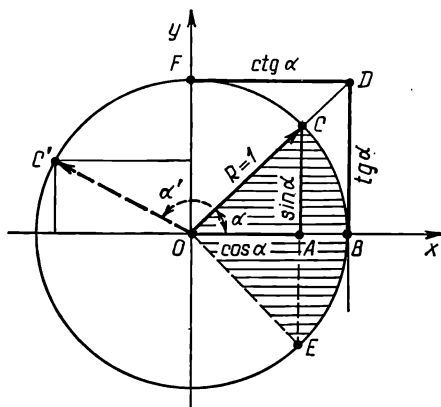


Рис. 4.7. График интерпретации определений тригонометрических функций

ного угла или длины дуги  $BC$ , но и как площадь сектора круга с углом  $2\alpha$ : площадь равна  $0,5R^2 \cdot 2\alpha = \alpha$  (на рис. 4.7 сектор заштрихован). Именно поэтому тригонометрические функции иногда называют *круговыми* [25, 70].

Функции  $y = \sin x$  и  $y = \cos x$  определены на всем множестве действительных чисел, но область их значений ограничена:  $y \in [-1, 1]$ . Обе функции являются периодическими с периодом  $T = 2\pi$ , функция  $\sin x$  — нечетная, а  $\cos x$  — четная. Графики функций  $\sin x$  и  $\cos x$  пересекаются с осью абсцисс соответственно в точках  $x = k\pi$  и  $x = \pi/2 + k\pi$  и имеют экстремумы (максимум или минимум) в точках  $x = \pi/2 + \pi n$  и  $x = k\pi$ ,  $k = 0, \pm 1, \pm 2, \dots$  (рис. 4.8, а). Функции  $y = \tg x$  и  $y = \ctg x$  определены для всех  $x$ , кроме  $x = \pi/2 + k\pi$  для тангенса и  $x = k\pi$  для котангенса, в которых соответственно  $\cos x$  и  $\sin x$  обращаются в нуль. Обе функции периодические с периодом  $T = \pi$ , нечетные и не ограниченные в области своих значений. Графики функций  $\tg x$  и  $\ctg x$  пересекаются с осью абсцисс соответственно в точках  $x = k\pi$  и  $x = \pi/2 + k\pi$  (рис. 4.8, б, в).

Функция  $y = \sin x$  разлагается в ряд Тейлора:

$$\sin x = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \dots \quad (4.10)$$

с радиусом сходимости  $R = \infty$ . Определим количество членов ряда (4.10), необходимое при вычислении функции

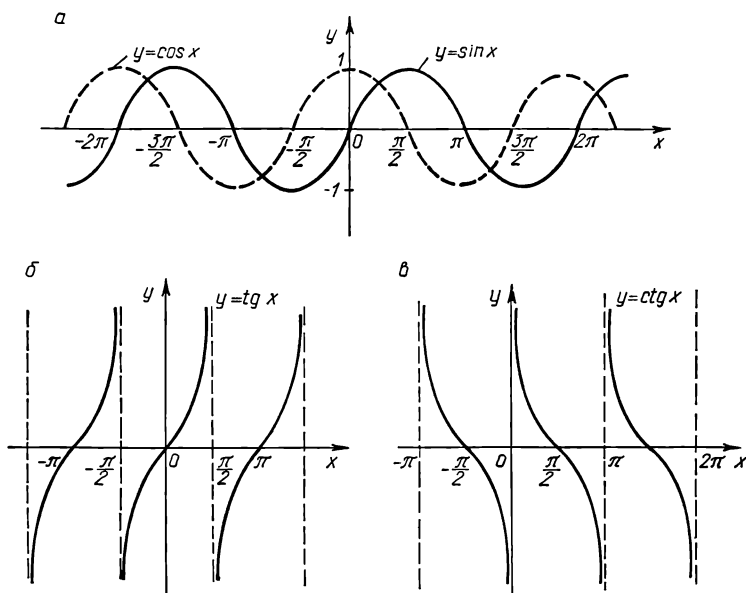


Рис. 4.8. Графики тригонометрических функций:  
а — синуса и косинуса; б — тангенса; в — котангенса

с точностью до четырех значащих десятичных цифр, т. е.  $\Delta = 0,5 \cdot 10^{-4}$ , для диапазона аргумента  $x \in [-\pi/4, \pi/4]$ . Подставляя в ряд (4.10) значение  $x = \pi/4 \approx 0,7854$ , найдем, что отбрасывание члена  $x^7/7!$  вносит погрешность  $0,37 \cdot 10^{-4} < \Delta$ . Поэтому для вычисления функции  $y = \sin x$  в заданном диапазоне и с требуемой точностью достаточно взять (с запасом) первые 4 члена ряда (4.10) с коэффициентами:  $1/1! = 1$ ,  $-1/3! = -0,1667$ ,  $1/5! = 0,8333 \cdot 10^{-2}$  и  $-1/7! = -0,1984 \cdot 10^{-3}$ . Этот метод вычисления синуса положен в основу программы СИН:

1A30		ORG	1A30H
18E0	ПОЛИН	SET	18E0H
1931	БУФЕР	SET	1931H
00B8	ПМЗ	SET	00B8H

СИН:

```

*****
;ПОДПРОГРАММА ВЫЧИСЛЕНИЯ СИНУСА ОСТРОГО УГЛА Y= SIN X,
;ГДЕ (-"ПИ"/4 <= X <= +"ПИ"/4).
;ВХОДНЫЕ ПАРАМЕТРЫ: (H,L)—АДРЕС АРГУМЕНТА, ПРЕДСТАВЛЯЕМОГО
;В ФОРМЕ 3-БАЙТНОГО ДВОИЧНОГО ЧИСЛА В ДОПОЛНИТЕЛЬНОМ
;КОДЕ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ, (D,E)—АДРЕС ФУНКЦИИ, ПРЕДСТАВ-
```

```

;ЛЯЕМОЙ В АНАЛОГИЧНОМ ВИДЕ. ВЫХОДНОЙ ПАРАМЕТР: CY=1-ПРИЗ-
;НАК ПЕРЕПОЛНЕНИЯ ИЛИ АНТИПЕРЕПОЛНЕНИЯ ПОРЯДКА РЕЗУЛЬТА-
;ТА. ИСПОЛЬЗУЕТСЯ ДОПОЛНИТЕЛЬНАЯ РАБОЧАЯ ОБЛАСТЬ ПАМЯТИ:
; "КОЗФ" ОБЪЕМОМ 8*3=24 БАЙТ ДЛЯ ХРАНЕНИЯ КОЭФФИЦИЕНТОВ
; СТЕПЕННОГО РЯДА. ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ, СОХРАНЯЮТСЯ
; (H,L), (D,E). ГЛУБИНА СТЕКА-30. ИСПОЛЬЗУЮТСЯ ПОДПРОГРАММЫ:
; *ПОЛИН*, *УДПЗ3*, *КОМЗ*, *ОБНЗ*, *УДФ17*, *НМАН2*, *СДПЗ3*,
; *ОБМЗ*, *ДМАН2*, *ПМАН2*, *ПМЗ*, *ДОПВ*, *ДОПД*, *УЗБ2*, *УЗ24*
; ОЦЕНКА ДЛИНА-19 БАЙТ (+556 БАЙТ ПОДПРОГРАММ), ВРЕМЯ-НЕ
; БОЛЕЕ 37770 ТАКТОВ (С УЧЕТОМ ПОДПРОГРАММ).

```

```

;*****

```

```

; ПОДГОТОВКА РЕГИСТРОВ К ВЫЧИСЛЕНИЮ ПОЛИНОМА

```

1A30 E5	PUSH	H	; СОХРАНЕНИЕ АДРЕСА АРГУМЕНТА
1A31 D5	PUSH	D	; СОХРАНЕНИЕ АДРЕСА ФУНКЦИИ
1A32 3E07	MVI	A,7	; (A)-СТЕПЕНЬ ПОЛИНОМА
1A34 11431A	LXI	D,КОЗФ	; (D,E)-АДРЕС КОЭФФИЦИЕНТОВ
; ВЫЧИСЛЕНИЕ ПОЛИНОМА			
1A37 CDE018	CALL	ПОЛИН	
1A3A D1	POP	D	; ВОССТАНОВЛЕНИЕ АДРЕСА РЕЗУЛЬТАТА
; ПЕРЕМЕЩЕНИЕ РЕЗУЛЬТАТА ИЗ ОБЛАСТИ "БУФЕР"			
1A3B 213119	LXI	H,БУФЕР	
1A3E CDB800	CALL	ПМЗ	
1A41 E1	POP	H	; ВОССТАНОВЛЕНИЕ АДРЕСА АРГУМЕНТА
1A42 C9	RET		; CY=1, ЕСЛИ ОШИБКА ПОРЯДКА
; ОБЛАСТЬ КОЭФФИЦИЕНТОВ A(0), ..., A(7)			
1A43 00	КОЗФ: DB	00H	; A(0)=0
1A44 0000	DW	0000H	
1A46 41	DB	41H	; A(1)=1=41 80 00H
1A47 8000	DW	0080H	
1A49 00	DB	00H	; A(2)=0
1A4A 0000	DW	0000H	
1A4C BE	DB	0BEH	; A(3)=-0,1667=BE 55 4CH
1A4D 554C	DW	4C55H	
1A4F 00	DB	00H	; A(4)=0
1A50 0000	DW	0000H	
1A52 3A	DB	3AH	; A(5)=0,8333*10**(-2)=3A 88 87H
1A53 8887	DW	8788H	
1A55 00	DB	00H	; A(6)=0
1A56 0000	DW	0000H	
1A58 B4	DB	0B4H	; A(7)=-0,1984*10**(-3)=B4 2F F7H
1A59 2FF7	DW	0F72FH	
0000	END		

Программа вычисляет синус посредством подпрограммы ПОЛИН, поэтому в ней фиксированы значения и младших нулевых коэффициентов полинома Тейлора. Для нахождения синуса при значениях аргумента вне диапазона  $[-\pi/4, \pi/4]$  полезны формулы приведения:  $\sin x = \cos(\pi/2 - x)$ ;  $\sin x = \sin(\pi - x) = -\sin(\pi + x) = -\sin(2\pi - x)$ ;  $\sin x = \sin(x + k \cdot 2\pi)$ .

Функция  $y = \cos x$  разлагается в ряд Тейлора:

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \dots \quad (4.11)$$

с радиусом сходимости  $R = \infty$ . Для обеспечения той же точности вычислений и в том же диапазоне, что и для функций синуса, достаточно взять первые 4 члена ряда (4.11) с коэффициентами:  $1$ ;  $-1/2! = -0,5$ ;  $1/4! = 0,4167 \cdot 10^{-1}$  и  $-1/6! = -0,1389 \cdot 10^{-2}$ . Этот метод использует программа вычисления косинуса КОС:

1A70		ORG	1A70H
18E0	ПОЛИН	SET	18E0H
1931	БУФЕР	SET	1931H
00B8	ПМЗ	SET	00B8H

КОС:

```

;*****
;ПОДПРОГРАММА ВЫЧИСЛЕНИЯ КОСИНУСА ОСТРОГО УГЛА Y=cos X,
;ГДЕ (-"ПИ"/4 <= X <= +"ПИ"/4).
;ВХОДНЫЕ ПАРАМЕТРЫ: (H,L) -АДРЕС АРГУМЕНТА, ПРЕДСТАВЛЕННОГО
;В ФОРМЕ 3-БАЙТНОГО ДВОИЧНОГО ЧИСЛА В ДОПОЛНИТЕЛЬНОМ
;КОДЕ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ, (D,E) -АДРЕС ФУНКЦИИ, ПРЕДСТАВЛЯ-
;ЕМОЙ В АНАЛОГИЧНОМ БИДЕ. ВЫХОДНОЙ ПАРАМЕТР: CY=1-ПРИЗНАК
;ПЕРЕПОЛНЕНИЯ ИЛИ АНТИПЕРЕПОЛНЕНИЯ ПОРЯДКА РЕЗУЛЬТАТА.
;ИСПОЛЬЗУЕТСЯ ДОПОЛНИТЕЛЬНАЯ РАБОЧАЯ ОБЛАСТЬ ПАМЯТИ
;"КОЗФ" ОБЪЕМОМ 7*3=21 БАЙТ ДЛЯ ХРАНЕНИЯ КОЭФФИЦИЕНТОВ
;СТЕПЕННОГО РЯДА. ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ, СОХРАНЯЮТСЯ
;(H,L), (D,E). ГЛУБИНА СТЕКА=30. ИСПОЛЬЗУЮТСЯ ПОДПРОГРАММЫ:
;*ПОЛИН*, *УДПЗ3*, *КОМЗ*, *ОВНЗ*, *УДФ17*, *НМАН2*, *СДПЗ3*,
;*ОЕМЗ*, *ДМАН2*, *ПМАН2*, *ПМЗ*, *ДОПВ*, *ДОПД*, *УЗЗБ*, *УЗ4А
;ОЦЕНКА: ДЛИНА=19 БАЙТ (+556 БАЙТ ПОДПРОГРАММ), ВРЕМЯ=НЕ
;БОЛЕЕ 32820 ТАКТОВ (С УЧЕТОМ ПОДПРОГРАММ).
;*****
;ПОДГОТОВКА РЕГИСТРОВ К ВЫЧИСЛЕНИЮ ПОЛИНОМА
1A70 E5      PUSH    H      ;СОХРАНЕНИЕ АДРЕСА АРГУМЕНТА
1A71 D5      PUSH    D      ;СОХРАНЕНИЕ АДРЕСА ФУНКЦИИ
1A72 3E06    MVI     A,6     ;(A)-СТЕПЕНЬ ПОЛИНОМА=6
1A74 11831A  LXI     D,КОЗФ  ;(D,E)-АДРЕС КОЭФФИЦИЕНТОВ

;ВЫЧИСЛЕНИЕ ПОЛИНОМА
1A77 CDE018  CALL    ПОЛИН
1A7A D1      POP     D      ;ВОССТАНОВЛЕНИЕ АДРЕСА РЕЗУЛЬТАТА
;ПЕРЕМЕЩЕНИЕ РЕЗУЛЬТАТА ИЗ ОБЛАСТИ "БУФЕР"
1A7B 213119  LXI     H,БУФЕР
1A7E CDB800  CALL    ПМЗ
1A81 E1      POP     H      ;ВОССТАНОВЛЕНИЕ АДРЕСА АРГУМЕНТА
1A82 C9      RET          ;CY=1, ЕСЛИ ОШИБКА ПОРЯДКА

;ОБЛАСТЬ КОЭФФИЦИЕНТОВ A(0), ..., A(6)
1A83 41      КОЗФ:  DB     41H      ;A(0)=1=41 80 00H
1A84 8000    DW     0080H
1A86 00      DB     00H          ;A(1)=0
1A87 0000    DW     0000H
1A89 C0      DB     0C0H        ;A(2)=-0,5=C0 80 00H
1A8A 8000    DW     0080H
1A8C 00      DB     00H          ;A(3)=0
1A8D 0000    DW     0000H
1A8F 3C      DB     3CH        ;A(4)=0,4167*10**(-1)=3C AA ADH
1A90 AAAD    DW     0AADAH

```

1A92 00	DB	00H	‡A(5)=0
1A93 0000	DW	0000H	
1A95 B7	DB	0B7H	‡A(6)=-0,1389*10**(-2)=B7 49 F2H
1A96 49F2	DW	0F249H	
0000	END		

Для нахождения функции  $y = \cos x$  при значениях аргумента вне диапазона  $[-\pi/4, \pi/4]$  можно использовать формулы приведения  $\cos x = \sin(\pi/2 - x)$ ;  $\cos x = -\cos(\pi - x) = -\cos(\pi + x) = \cos(2\pi - x)$ ;  $\cos x = \cos(x + k \cdot 2\pi)$ . Тестовые данные для программ СИН и КОС приведены в табл. 4.7 и 4.8.

Табл. 4.7. Меры углов

Градусы	Рadiany	
	$A_{10}$	$A_{16}$
45	$0,7854 \cdot 10^0$	40C910
20	$0,3491 \cdot 10^0$	3FB2BE
5	$0,8727 \cdot 10^{-1}$	3DB2BA
-5	$-0,8727 \cdot 10^{-1}$	BD4D46
-20	$-0,3491 \cdot 10^0$	BF4D42
-45	$-0,7854 \cdot 10^0$	C036F0

Табл. 4.8. Тестовые данные  $y = \sin x$ ,  $y = \cos x$

x, град	$y = \sin x$		$y = \cos x$	
	$A_{10}$	$A_{16}$	$A_{10}$	$A_{16}$
45	$0,7071 \cdot 10^0$	40B504	$0,7071 \cdot 10^0$	40B505
20	$0,3420 \cdot 10^0$	3FAF22	$0,9397 \cdot 10^0$	40F08F
5	$0,8716 \cdot 10^{-1}$	3DB280	$0,9962 \cdot 10^0$	40FF07
-5	$-0,8716 \cdot 10^{-1}$	BD4D80	$0,9962 \cdot 10^0$	40FF07
-20	$-0,3420 \cdot 10^0$	BF50DE	$0,9397 \cdot 10^0$	40F08F
-45	$-0,7071 \cdot 10^0$	CO4AFC	$0,7071 \cdot 10^0$	40B505

Функция  $y = \operatorname{tg} x$ , как и функции синуса, косинуса, может быть представлена рядом Тейлора. Однако этот ряд сходится гораздо медленнее, чем ряды (4.10), (4.11), и, следовательно, требует много членов для обеспечения точности и необходимого диапазона аргумента. Для вычисления функций  $y = \operatorname{tg} x$  и  $y = \operatorname{ctg} x$  воспользуемся их определением посредством функции синуса и косинуса:  $\operatorname{tg} x = \sin x / \cos x$ ,  $\operatorname{ctg} x = \cos x / \sin x$ . Программы ТАН и КОТАН вычисляют функции тангенса и котангенса на основе этих определений:

1A80		ORG	1A80H
1A30	SIN	SET	1A30H
1A70	KOC	SET	1A70H
12C0	ДДПЗЗ	SET	12C0H

ТАН:

```

;*****
;ПОДПРОГРАММА ВЫЧИСЛЕНИЯ ТАНГЕНСА ОСТРОГО УГЛА  $Y = \text{TG } X =$ 
;  $=(\sin X / \cos X)$ , ГДЕ  $(-\pi/4 \leq X \leq +\pi/4)$ .
;ВХОДНЫЕ ПАРАМЕТРЫ: (H,L) - АДРЕС АРГУМЕНТА X, ПРЕДСТАВЛЕН-
;НОГО В ФОРМЕ 3-БАЙТНОГО ДВОИЧНОГО ЧИСЛА В ДОПОЛНИТЕЛЬ-
;НОМ КОДЕ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ, (D,E) - АДРЕС ФУНКЦИИ Y,
;ПРЕДСТАВЛЯЕМОЙ В АНАЛОГИЧНОМ ВИДЕ. ВЫХОДНОЙ ПАРАМЕТР: CY=
;=1 - ПРИЗНАК ПЕРЕПОЛНЕНИЯ ИЛИ АНТИПЕРЕПОЛНЕНИЯ ПОРЯДКА
;РЕЗУЛЬТАТА, ПРИЗНАК ДЕЛЕНИЯ НА НОЛЬ. ИСПОЛЬЗУЕТСЯ ТРЕХ-
;БАЙТНАЯ ОБЛАСТЬ ПАМЯТИ "БУФЕР" ДЛЯ ХРАНЕНИЯ ПРОМЕЖУТОЧ-
;НОГО РЕЗУЛЬТАТА. ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ, СОХРАНЯЮТСЯ
; (H,L), (D,E). ГЛУБИНА СТЕКА - 32. ИСПОЛЬЗУЮТСЯ ПОДПРОГРАММЫ:
;*СИН*, *КОС*, *ПОЛИН*, *ПМЗ*, *УДПЗЗ*, *КОМЗ*, *ОВНЗ*, *УДФ17*
;*МНАН2*, *САПЗЗ*, *ОВМЗ*, *ДМАН2*, *ПМАН2*, *ДОПВ*, *ДОПД*,
;*УЗ2В*, *У24А*, *ДДПЗЗ*, *ДДФ17*.
;ОЦЕНКА: ДЛИНА - 26 БАЙТ (+714 БАЙТ ПОДПРОГРАММ), ВРЕМЯ - НЕ
;БОЛЕЕ 73520 ТАКТОВ (С УЧЕТОМ ПОДПРОГРАММ).
;*****

```

;ВЫЧИСЛЕНИЕ SIN X

1AB0	CD301A	CALL	SIN	; (D,E) - АДРЕС SIN X
1AB3	D8	RC		; ЕСЛИ ОШИБКА ПОРЯДКА
;ВЫЧИСЛЕНИЕ COS X				
1AB4	D5	PUSH	D	; СОХРАНЕНИЕ АДРЕСА РЕЗУЛЬТАТА
1AB5	11CA1A	LXI	D, БУФЕР	
1AB8	CD701A	CALL	KOC	; (D,E) - АДРЕС COS X
1ABB	D2C01A	JNC	ПЕР	; ЕСЛИ НЕТ ОШИБКИ ПОРЯДКА
1ABE	D1	POP	D	; ВОССТАНОВЛЕНИЕ АДРЕСА РЕЗУЛЬТАТА
1ABF	C9	RET		; CY=1 - ОШИБКА ПОРЯДКА
;ВЫЧИСЛЕНИЕ $\text{TG } X = \sin X / \cos X$				
1AC0	C1	PER:	POP	B
1AC1	E5		PUSH	H
1AC2	EB		XCHG	; (H,L) - АДРЕС COS X
1AC3	CD0C12	CALL	ДДПЗЗ	; (B,C) - АДРЕС TG X
1AC6	50	MOV	D, B	
1AC7	59	MOV	E, C	; (D,E) - АДРЕС TG X
1AC8	E1	POP	H	; ВОССТАНОВЛЕНИЕ АДРЕСА АРГУМЕНТА
1AC9	C9	RET		; CY=1, ЕСЛИ ОШИБКА
;РАБОЧАЯ ОБЛАСТЬ ПАМЯТИ "БУФЕР"				
1ACA		БУФЕР:	DS	3
0000		END		;3 БАЙТА

1AE0		ORG	1AE0H
1A30	SIN	SET	1A30H
1A70	KOC	SET	1A70H
12C0	ДДПЗЗ	SET	12C0H

КОТАН:

```

;*****
;ПОДПРОГРАММА ВЫЧИСЛЕНИЯ КОТАНГЕНСА ОСТРОГО УГЛА  $Y =$ 
;  $= \text{CTG } X = (\cos X / \sin X)$ , ГДЕ  $(-\pi/4 \leq X \leq +\pi/4)$ .
;ВХОДНЫЕ ПАРАМЕТРЫ: (H,L) - АДРЕС АРГУМЕНТА, ПРЕДСТАВЛЕННОГО
;В ФОРМЕ 3-БАЙТНОГО ДВОИЧНОГО ЧИСЛА В ДОПОЛНИТЕЛЬНОМ
;КОДЕ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ, (D,E) - АДРЕС ФУНКЦИИ, ПРЕДСТАВ-

```



```

;ЛЯМОЙ В АНАЛОГИЧНОМ ВИДЕ. ВЫХОДНОЙ ПАРАМЕТР: CY=1-ПРИЗ-
;НАК ПЕРЕПОЛНЕНИЯ ИЛИ АНТИПЕРЕПОЛНЕНИЯ ПОРЯДКА РЕЗУЛЬТА-
;ТА, ПРИЗНАК ДЕЛЕНИЯ НА НОЛЬ. ИСПОЛЬЗУЕТСЯ ТРЕХБАЙТНАЯ ОБ-
;ЛАСТЬ ПАМЯТИ "БУФЕР" ДЛЯ ХРАНЕНИЯ ПРОМЕЖУТОЧНОГО РЕЗУЛЬ-
;ТАТА. ИСПОЛЬЗУЮТСЯ ВСЕ РЕГИСТРЫ, СОХРАНЯЮТСЯ (H,L), (D,E).
;ГЛУБИНА СТЕКА-32. ИСПОЛЬЗУЮТСЯ ПОДПРОГРАММЫ: *СИН*, *КОС*,
; *ПОЛИН*, *ПМЗ*, *УДПЗ*, *КОМЗ*, *ОБНЗ*, *УДФ17*, *НМАН2*,
; *СДПЗ3*, *ОБМЗ*, *ДМАН2*, *ПМАН2*, *ДОПВ*, *ДОПД*, *УЗБВ*,
; *У24А*, *ДДПЗ3*, *ДДФ17*.
;ОЦЕНКА: ДЛИНА-26 БАЙТ (+714 БАЙТ ПОДПРОГРАММ), ВРЕМЯ- НЕ
; БОЛЕЕ 73520 ТАКТОВ (С УЧЕТОМ ПОДПРОГРАММ).
; *****
; ВЫЧИСЛЕНИЕ COS X

```

```

1AE0 CD701A      CALL    KOC      ; (D,E)-АДРЕС COS X
1AE3 D8          RC              ; ЕСЛИ ОШИБКА ПОРЯДКА

; ВЫЧИСЛЕНИЕ SIN X
1AE4 D5          PUSH    D        ; СОХРАНЕНИЕ АДРЕСА РЕЗУЛЬТАТА
1AE5 11FA1A      LXI      D, БУФЕР
1AE8 CD301A      CALL    СИН      ; (D,E)-АДРЕС SIN X
1AEB D2F01A      JNC      ПЕР      ; ЕСЛИ НЕТ ОШИБКИ ПОРЯДКА
1AEE D1          POP      D        ; ВОССТАНОВЛЕНИЕ АДРЕСА РЕЗУЛЬТАТА
1AEF C9          RET             ; CY=1-ОШИБКА ПОРЯДКА

; ВЫЧИСЛЕНИЕ CTG X= COS X / SIN X
1AF0 C1          PER:  POP      B        ; ВОССТАНОВЛЕНИЕ АДРЕСА РЕЗУЛЬТАТА
1AF1 E5          PUSH    H        ; СОХРАНЕНИЕ АДРЕСА АРГУМЕНТА
1AF2 EB          XCHG         ; (H,L)-АДРЕС SIN X
1AF3 CDC012      CALL    ДДПЗ3      ; (B,C)-АДРЕС CTG X
1AF6 50          MOV      D, B
1AF7 59          MOV      E, C      ; (D,E)-АДРЕС CTG X
1AF8 E1          POP      H        ; ВОССТАНОВЛЕНИЕ АДРЕСА АРГУМЕНТА
1AF9 C9          RET             ; CY=1, ЕСЛИ ОШИБКА

; РАБОЧАЯ ОБЛАСТЬ ПАМЯТИ "БУФЕР"
1AFA            БУФЕР: DS      3      ; 3 БАЙТА
0Q00            END

```

Для нахождения функций при других значениях аргумента, отличных от диапазона  $[-\pi/4, \pi/4]$ , полезны формулы приведения:  $\operatorname{tg} x = \operatorname{ctg}(\pi/2 - x)$ ;  $\operatorname{ctg} x = \operatorname{tg}(\pi/2 - x)$ ;  $\operatorname{tg} x = -\operatorname{tg}(\pi - x) = \operatorname{tg}(\pi + x) = -\operatorname{tg}(2\pi - x)$ ;  $\operatorname{ctg} x = -\operatorname{ctg}(\pi - x) = \operatorname{ctg}(\pi + x) = -\operatorname{ctg}(\pi - x)$ ;  $\operatorname{tg} x = \operatorname{tg}(x + k \cdot \pi)$ ;  $\operatorname{ctg} x = \operatorname{ctg}(x + k \cdot \pi)$ . Тестовые данные для программ ТАН и КОТАН приведены в табл. 4.9.

Табл. 4.9. Тестовые данные  $y=\operatorname{tg} x$ ,  $y=\operatorname{ctg} x$

x, град	y=tgx		y=ctgx	
	A <sub>10</sub>	A <sub>16</sub>	A <sub>10</sub>	A <sub>16</sub>
45	0,1000·10 <sup>1</sup>	418000	0,10000·10 <sup>1</sup>	418000
20	0,3640·10 <sup>0</sup>	3FBA5E	0, 2 7 4 7 · 1 0 <sup>-1</sup>	42AFCE
5	0,8749·10 <sup>-1</sup>	3DB32D	0,1143·10 <sup>2</sup>	44B6E1
-5	-0,8749·10 <sup>-1</sup>	BD4CD3	-0,1143·10 <sup>2</sup>	C44A1F
-20	-0,3640·10 <sup>0</sup>	BF45A2	-0,2747·10 <sup>1</sup>	C25032
-45	-0,1000·10 <sup>1</sup>	C18000	-0,1000·10 <sup>1</sup>	C18000

## 4.9. ГИПЕРБОЛИЧЕСКИЕ ФУНКЦИИ

Гиперболические функции, как и тригонометрические, определяют трансцендентную зависимость между прямоугольными координатами  $(x, y)$  некоторой точки  $C$  гиперболы  $x^2 - y^2 = 1$  (а не окружности  $x^2 + y^2 = 1$ , как в случае тригонометрических функций), с одной стороны, и аргументом  $t$ , представляющим площадь гиперболического сектора, с другой стороны (рис. 4.9). Ордината точки  $C$  определяет гиперболический синус  $y = \operatorname{sh} t$ , а ее абсцисса — гиперболический косинус  $y = \operatorname{ch} t$ , где величина  $t$  представляет площадь гиперболического сектора  $OCD$ . Вычислив эту площадь как разность площадей  $x$  треугольника  $OCD$  и криволинейного интеграла  $2 \int y dx$ , можно определить, например, гиперболический косинус  $\operatorname{ch} t = (e^t + e^{-t})/2$  [25].

Гиперболические функции выражаются линейными комбинациями показательных функций  $e^x$  и  $e^{-x}$  и их отношениями:  $\operatorname{sh} x = (e^x - e^{-x})/2$ ;  $\operatorname{ch} x = (e^x + e^{-x})/2$ . Гиперболические тангенс  $\operatorname{th} x$  и котангенс  $\operatorname{cth} x$  определяются аналогично одноименным тригонометрическим функциям как отношения:  $\operatorname{th} x = \operatorname{sh} x / \operatorname{ch} x$  и  $\operatorname{cth} x = \operatorname{ch} x / \operatorname{sh} x$ . Функ-

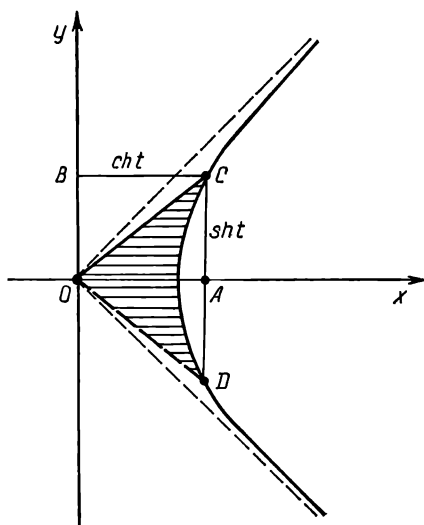


Рис. 4.9. График интерпретации определений гиперболических функций

ция  $\operatorname{sh} x$  определена для всех  $x$ , не ограничена в области значений, нечетна и равна нулю, в точке  $x = 0$ . Ее график подобен графику нечетной степенной функции. Функция  $\operatorname{ch} x$  также определена для всех  $x$ , изменяется в области значений  $[1, \infty)$ , четна и равна единице в точке  $x = 0$ . Ее график подобен графику четной степенной функции. Функция  $\operatorname{th} x$  определена для всех  $x$ , изменяется в диапазоне  $(-1, 1)$ , нечетна и равна нулю в точке  $x = 0$ . Функция  $\operatorname{cth} x$  определена для всех ненулевых  $x$ , изменяется в области значений  $(-\infty, -1)$  и  $(1, +\infty)$ , нечетна.

Функции  $y = \operatorname{sh} x$  и  $y = \operatorname{ch} x$  разлагаются в ряд Тейлора:

$$\operatorname{sh} x = \frac{x}{1!} + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \dots;$$

$$\operatorname{ch} x = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \dots$$

с радиусом сходимости  $R = \infty$ . Легко заметить, что разложения гиперболических функций отличаются от разложений одноименных тригонометрических функций (4.10) и (4.11) только знаками. Поэтому для вычисления этих функций можно воспользоваться программами СИН и КОС, в которых необходимо предварительно перевести в прямой код отрицательные коэффициенты. Программы ТАН и КОТАН можно использовать для вычисления гиперболических тангенса и котангенса без изменений (применяя, естественно, подпрограммы вычисления гиперболических синуса и косинуса).

## 5. ПРОГРАММЫ ОБРАБОТКИ СТРУКТУР ДАННЫХ

### 5.1. ОБЩИЕ СВЕДЕНИЯ

В предыдущих главах рассмотрена обработка данных, представленных в виде чисел различных систем счисления и форматов. На машинном уровне эти числа фиксируются как последовательности бит с примитивной организацией в виде байтов или многобайтных слов. Однако с точки зрения пользователя такие числа представляют собой информационные объекты более высокого уровня абстракции. Например, числа с плавающей запятой определяются двумя взаимосвязанными элементами — мантиссой, порядком — и арифметическо-логическими операциями над ними. Если эти составляющие определены, то говорят о реализации числового типа данных в формате с плавающей запятой или типа действительных данных. Обработка информации в микропроцессорных системах (МП системах) не ограничивается числами, а использует различные типы данных и их совокупности — *структуры данных*.

Считается, что тип данных определен, если определены множества его элементов и отношений (операций) между элементами, причем эти отношения реализуются машинными средствами (аппаратурой или программами). *Тип данных* — это некоторая структура, с помощью которой осуществляется в машине иерархическое отображение информационных объектов: структуры нижнего уровня используются в качестве элементов структур более высокого уровня. Понятие структуры данных обобщает понятие типа данных для такой иерархии представлений. При выборе конкретных структур данных — *структурировании данных* — необходимо, с одной стороны, учитывать их проблемный аспект, т. е. пригодность для решения задач определенного класса, а с другой стороны, — ограничения на оперативность доступа к элементам структур и возможности их изменения (изменения числа элементов и отношений между ними), что определяет эффективность решения задач. Известны различные типы структур данных [40, 65].

Наиболее простой структурой является *одномерный*

*массив*, представляющий конечное множество простых данных одного типа — множество однородных элементов. Все элементы массива связаны отношением непосредственного следования, что позволяет их пронумеровать. Номер (индекс) позволяет однозначно идентифицировать любой элемент массива. *Многомерные массивы* можно рассматривать как одномерные, элементами которых являются не простые данные, а одномерные массивы.

Если составить массив из данных различных типов, образуется структура другого вида — *запись*. Запись, как правило, содержит совокупность информации об одном объекте. Элементы записи называются *полями*. Доступ к элементу возможен либо по его номеру в записи, либо по значению поля. Совокупность записей, имеющих одинаковую организацию, образует *таблицу*. Таблицу можно рассматривать как массив, элементами которого являются записи. Обычно одно из полей каждой записи отводится для хранения уникального кода — *ключа*. Таблица является наиболее распространенной структурой данных в большинстве программных комплексов. Массивы, записи и таблицы сохраняют свою структуру постоянной в течение всего времени существования. В силу этого их называют *статическими структурами*. Достоинством этих структур является смежность элементов и как следствие непрерывность области памяти, отводимой под структуры, недостатком — постоянство отношений между элементами, которое задается при создании структуры и не подлежит изменению в процессе ее обработки.

Помимо статических, существуют *динамические структуры*, длины и отношения между элементами которых изменяются в процессе обработки. Одним из представителей динамических структур является *связный список*, элементы которого — это записи одного формата, содержащие указатели на другие элементы списка. Простейший связный список — *односвязный*. Поле указателя элемента односвязного списка содержит адрес следующего элемента (рис. 5.1), поле указателя последнего элемента — нулевой указатель, свидетельствующий об окончании списка. Удаление элементов из списка, вставка их в любое место списка сводятся лишь к изменению указателей. Организация списка не требует физической смежности элементов в памяти: они могут быть расположены различным образом относительно друг друга. Вследствие этого в динамических структурах увели-

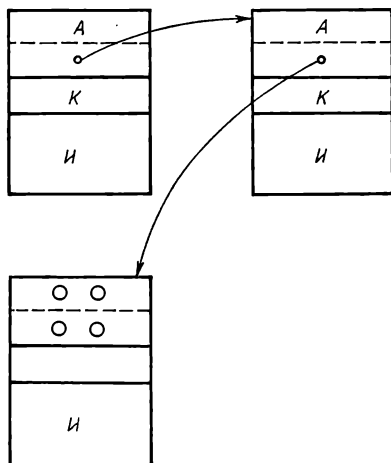


Рис. 5.1. Структура однонаправленного связанного списка:  
 А — адресная ссылка на следующий элемент; К — «ключ» — байт с уникальным для каждого элемента кодом; И — информационная основа элемента

чивается время доступа к элементам. Если в статических структурах по номеру элемента можно сразу определить его адрес в памяти, то в динамических структурах для доступа к искомому элементу необходимо прочесть поля указателей всех предыдущих элементов списка.

Кроме односвязных, существуют *двухсвязные* и *многосвязные (сетевые) списки*. В элементах двухсвязных списков присутствуют два поля указателей: один указывает на последующий элемент, второй — на предыдущий. В элементах многосвязных списков имеется несколько указателей.

Статические и динамические структуры обладают противоположными свойствами с точки зрения изменяемости и оперативности доступа к элементам. Компромиссное решение представляют полустатические структуры, к которым относятся стеки, очереди и деки. *Полустатическая структура* — последовательный список, в котором элементы связаны между собой не указателями, а отношениями непосредственного следования. В отличие от статических структур в полустатических имеется возможность изменять количество элементов. На процедуры изменения в полустатических структурах накладываются

определенные ограничения, которые и характеризуют разновидности таких структур. *Стек* представляет собой последовательный список переменной длины; элементы включаются в этот список и исключаются из него только с одной стороны: с начала или конца списка. Со стеком связан лишь один указатель, содержащий адрес первого (или последнего) элемента стека. *Очередь* отличается от стека тем, что позволяет включать элементы только с конца («хвоста») списка, а исключать только с его начала («головы»). В связи с этим для очереди требуются два указателя, один из которых содержит адрес «хвоста», а второй — адрес «головы» очереди. Очередь специального вида, в которой элементы можно включать и исключать с любого конца, называется *деком*.

В этой главе описываются простые операции над структурами данных: организация доступа (прямого и ассоциативного), исключение, добавление элементов и их модификация. Все приведенные далее программы обладают свойством *реентерабельности*: программа в процессе выполнения может быть в любой момент прервана, вызвана в прерывающей программе и выполнена вновь полностью, а затем после возврата из прерывания продолжена с прерванного места, причем оба процесса выполнения программы будут правильными по результатам (второй вызов программы не изменяет промежуточных данных первого ее вызова). Требование реентерабельности предъявляется к программам, используемым в системах реального времени. Реентерабельность можно обеспечить, если все рабочие переменные, с которыми работает программа, располагать в регистрах микропроцессора. В этом случае в прерывающей программе достаточно «сохранять регистры в стеке» перед началом ее работы и «восстанавливать регистры из стека» после завершения работы программы. Тем самым гарантируется сохранность промежуточных данных предыдущего процесса выполнения прерываемой программы.

Однако не всегда удастся размещать все данные в регистрах: часть переменных приходится размещать в оперативной памяти. В этом случае для вложенных вызовов данной программы одни и те же рабочие переменные должны для каждого вызова размещаться в различных ячейках памяти, что выполнимо, если адрес рабочей области (области сохранения параметров) передавать в списке параметров вызываемой программы.

Другой, более эффективный прием — адресация рабочих переменных, а также части параметров через общий стек. Такой прием экономит оперативную память, но затрудняет доступ к рабочим переменным. Использовать для реализации этого приема команды работы со стекком неудобно, так как они оперируют только с вершиной стека. Целесообразно оформлять процедуры доступа к данным, расположенным в общем стеке, в виде *макрокоманд* (см. прил. 3).

Рассмотрим ряд макрокоманд:

```

; *****
; МАКРОКОМАНДА ОБМЕНА СОДЕРЖИМЫМ РЕГИСТРОВ.
; ПАРАМЕТРЫ: R1,R2-ИМЕНА РЕГИСТРОВ.
; ОЦЕНКА: ВРЕМЯ-15 ТАКТОВ; ДЛИНА-3 БАЙТА.
; *****
RCHG    MACRO    R1,R2
        MOV      A,R1
        MOV      R1,R2
        MOV      R2,A
        ENDM

; *****
; МАКРОКОМАНДА ОБМЕНА СОДЕРЖИМЫМ РЕГИСТРОВЫХ ПАР.
; ПАРАМЕТРЫ: X1,X2 - ИМЕНА РЕГИСТРОВЫХ ПАР.
; ОЦЕНКА:ВРЕМЯ - 44 ТАКТА;ДЛИНА-4 БАЙТА; ГЛУБИНА СТЕКА-4
; БАЙТА.
; *****
XCHR    MACRO    X1,X2
        PUSH     X1
        PUSH     X2
        POP      X1
        POP      X2
        ENDM

; *****
; МАКРОКОМАНДА ОБМЕНА СОДЕРЖИМЫМ РЕГИСТРОВОЙ ПАРЫ И ВЕР-
; ШИНЫ СТЕКА.
; ПАРАМЕТРЫ: X1-ИМЯ РЕГИСТРОВОЙ ПАРЫ (D ИЛИ B ).
; ОЦЕНКА: ВРЕМЯ - 77 ТАКТОВ; ДЛИНА - 13 БАЙТ; ГЛУБИНА
; СТЕКА - 2 БАЙТА.
; *****
XTRR    MACRO    X1
        XCHR     H,X1
        XTHL
        XCHR     H,X1
        ENDM

; *****
; МАКРОКОМАНДА ОБМЕНА СОДЕРЖИМЫМ РЕГИСТРОВОЙ ПАРЫ И
; N-М СЛОВОМ В СТЕКЕ.
; ПАРАМЕТРЫ: R1-ИМЯ СТАРШЕГО РЕГИСТРА РЕГИСТРОВОЙ ПАРЫ
; ( КРОМЕ H ), R2-ИМЯ МЛАДШЕГО РЕГИСТРА РЕГИСТРОВОЙ ПАРЫ
; ( КРОМЕ L ), N-НОМЕР СЛОВА В СТЕКЕ ОТНОСИТЕЛЬНО ТЕКУ-
; ЩЕГО ЗНАЧЕНИЯ SP.
; ОЦЕНКА:ВРЕМЯ-61 ТАКТ;ДЛИНА-9 БАЙТ;ГЛУБИНА СТЕКА-2 БАЙТА.
; *****
XTRN    MACRO    R1,R2,N
        PUSH     H

```



LXI	H, (N+1)*2
DAD	SP
RCHG	M, R2
INX	H
RCHG	M, R1
POP	H
ENDM	

```

; *****
; МАКРОКОМАНДА ЗАГРУЗКИ В РЕГИСТРОВУЮ ПАРУ D ИЛИ В СО-
; ДЕРЖИМОГО N-ГО СЛОВА СТЕКА.
; ПАРАМЕТРЫ: R1-ИМЯ СТАРШЕГО РЕГИСТРА РЕГИСТРОВОЙ ПАРЫ
; ( КРОМЕ H ), R2-ИМЯ МЛАДШЕГО РЕГИСТРА РЕГИСТРОВОЙ ПАРЫ
; ( КРОМЕ L ), N- НОМЕР СЛОВА В СТЕКЕ ОТНОСИТЕЛЬНО ТЕКУ-
; ЩЕГО ЗНАЧЕНИЯ SP.
; ОЦЕНКА: ВРЕМЯ-61 ТАКТ; ДЛИНА-7 БАЙТ; ГЛУБИНА СТЕКА-2 БАЙТА
; *****

```

LDSP	MACRO	R1, R2, N
	PUSH	H

LXI	H, (N+1)*2
DAD	SP
MOV	R2, H
INX	H
MOV	R1, H
POP	H
ENDM	

Макрокоманды RCHG и XCHR осуществляют обмен содержимым соответственно регистров и регистровых пар. В макрокоманде RCHG в качестве промежуточного запоминающего регистра используется аккумулятор, а в макрокоманде XCHR — вершина стека. Макрокоманда RCHG изменяет содержимое аккумулятора. Макрокоманды XTRR, XTRN и LDSP предназначены для работы с содержимым стека. Макрокоманда XTRR является аналогом процессорной команды XTHL и позволяет производить обмен между вершиной стека и регистровыми парами (B, C) и (D, E). Макрокоманда XTRN обеспечивает непосредственный доступ к указанному слову стека. Доступ к данным, содержащимся в стеке, осуществляется с помощью *базовой косвенной адресации*. В качестве базового адреса используется содержимое регистра указателя стека SP. В качестве операндов, определяющих имена регистров, нельзя указывать регистры (H) и (L), так как они используются для адресации содержимого стека. Действия макрокоманды LDSP аналогичны действиям макрокоманды XTRN, за исключением того, что содержимое указанных регистров теряется, а не помещается в стек, как

в XTRN. Следует отметить, что макрокоманды XTRN и LDSR изменяют содержимое аккумулятора, так как используют макрокоманду RCHG.

## 5.2. ФОРМИРОВАНИЕ МАССИВОВ

### 5.2.1. ПРОСТОЕ ФОРМИРОВАНИЕ МАССИВА

Формирование массива в МП системе, как правило, производится заполнением информацией, поступающей из внешних источников. Обслуживание этих внешних источников осуществляется специальными программами, которые назовем программами генерации кодов. При формировании массива может производиться некоторая дополнительная обработка поступающей информации.

Приведенная ниже программа выполняет простое формирование массива — без предварительной обработки (предполагается, что внешняя программа генерации кодов осуществляет при каждом своем вызове побайтную передачу очередного кода символа в регистр (A) и обеспечивает сохранение значений остальных регистров):

```

ЗАП1:
; *****
; ПОДПРОГРАММА ЗАПОЛНЕНИЯ МАССИВА КОДАМИ, ВЫРАБАТЫВАЕМЫМИ
; ВНЕШНЕЙ ПОДПРОГРАММОЙ ГЕНЕРАЦИИ КОДОВ (ПГК).
; ПАРАМЕТРЫ: (B,C) — АДРЕС НАЧАЛА МАССИВА, (D,E) — ДЛИНА МАСС-
; СИВА, (H,L) — АДРЕС ПГК.
; ОЦЕНКА: ВРЕМЯ-10+(82+T)*N, ГДЕ T — ВРЕМЯ ПГК, N — ДЛИНА
; МАССИВА; ДЛИНА-16 БАЙТ; ГЛУБИНА СТЕКА-4 БАЙТА.
; *****
; ВЫЗОВ ПОДПРОГРАММЫ ГЕНЕРАЦИИ КОДОВ
202B C5          PUSH    B
202C 013120      LXI     B, ПЕР1
202F C5          PUSH    B      ; АДРЕС ВОЗВРАТА ИЗ ПГК В СТЕК
2030 E9          PCHL
; ЗАПИСЬ ОЧЕРЕДНОГО БАЙТА И ПРОВЕРКА НА ОКОНЧАНИЕ МАССИВА
2031 C1          ПЕР1:    POP     B
2032 02          STAX    B
2033 03          INX     B
2034 1B          DCX     D
2035 7B          MOV     A, E
2036 B2          ORA     D
2037 C22B20      JNZ     ЗАП1
203A C9          RET

```

Структура программы представляет собой итеративный цикл. На каждой итерации выполняется запись в массив одного байта. Вызов подпрограммы генерации кодов выполняется командой PCHL: Так как эта команда

не обеспечивает сохранения адреса возврата в исходную программу, то он предварительно загружается в вершину стека командой PUSH B. После возврата из подпрограммы генерации кодов осуществляется запись полученного кода в формируемый массив по адресу (B, C), содержимое регистровой пары (B, C) увеличивается, а содержимое регистровой пары (D, E), равное количеству незаписанных байтов в формируемом массиве, уменьшается на единицу. Возврат из описываемой программы выполняется, когда количество незаписанных байтов становится равным нулю. Проверка содержимого регистровой пары (D, E) на нулевое значение производится путем логического сложения содержимого регистров этой пары.

### 5.2.2. ФОРМИРОВАНИЕ МАССИВА С КОНТРОЛЕМ КОДОВ ОКОНЧАНИЯ И ЗАБОЯ

Состав функций предварительной обработки данных, возлагаемых на программы формирования массива, определяется в основном спецификой работы внешнего источника информации. Рассмотрим в качестве примера программу формирования буфера консольного ввода, данные для которого вводятся со стандартной клавиатуры. Необходимо обеспечить ввод в буфер одной строки, представляющей цепочку символов, ограниченную кодом «ВК» — «Возврат каретки» (0DH), с возможностью коррекции вводимой строки посредством стирания последнего введенного символа консольной командой «ЗБ» — «Забой» (7FH). Программа имеет следующий вид:

ЗАП2:

```

;*****
; ПОДПРОГРАММА ЗАПОЛНЕНИЯ МАССИВА КОДАМИ, ВЫРАБАТЫВАЕМЫМИ
; ВНЕШНЕЙ ПОДПРОГРАММОЙ ГЕНЕРАЦИИ КОДОВ (ПГК) С КОНТРОЛЕМ
; КОДОВ ОДН(ЗАВЕРШЕНИЕ) И 7FH (ЗАБОЙ ПРЕДЫДУЩЕГО БАЙТА).
; ВХОДНЫЕ ПАРАМЕТРЫ: (B,C) - АДРЕС НАЧАЛА МАССИВА, (D,E) -
; ДЛИНА МАССИВА, (H,L) - АДРЕС ПГК.
; ВЫХОДНЫЕ ПАРАМЕТРЫ: (D,E) - КОЛИЧЕСТВО НЕЗАПИСАННЫХ БАЙТ,
; Z=1 - МАССИВ ЗАПОЛНЕН ПОЛНОСТЬЮ.
; ОЦЕНКА: ВРЕМЯ-42+122*N+346*M ТАКТОВ, ГДЕ N - КОЛИЧЕСТВО
; "ОБЫЧНЫХ" СИМВОЛОВ, M - КОЛИЧЕСТВО СИМВОЛОВ "ЗАБОЙ";
; ДЛИНА-65 БАЙТ; ГЛУБИНА СТЕКА-6 БАЙТ.
;*****

```

```

203B C5          PUSH      B
; ВЫЗОВ ПОДПРОГРАММЫ ГЕНЕРАЦИИ КОДОВ
203C C5          ЦИКЛ3: PUSH      B
203D 014220      LXI        B, PER2
2040 C5          PUSH      B          ; АДРЕС ВОЗВРАТА ИЗ ПГК В СТЕК
2041 E9          PCHL

```



вается с адресом начала буфера, который запоминается в стеке в начале работы программы. Коды «ВК» и «ЗБ» в буфер не записываются.

### 5.2.3. ФОРМИРОВАНИЕ МАССИВА С КОНТРОЛЕМ ПРОИЗВОЛЬНЫХ УПРАВЛЯЮЩИХ КОДОВ

Для унификации программ формирования массивов целесообразно предварительную обработку информации возложить на внешние подпрограммы, каждая из которых обрабатывает отдельный управляющий символ. Взаимодействие между программой формирования массива и подпрограммами обработки управляющих кодов можно осуществить с помощью специальной таблицы, структура которой приведена на рис. 5.2. Каждый элемент таблицы состоит из трех байтов и содержит значение управляющего кода, а также адрес подпрограммы его обработки. Последним байтом таблицы всегда является код 0FFH, что позволяет сделать таблицу переменной длины.

Ниже приведена программа формирования буферного массива, которая селектирует управляющие коды и в случае их обнаружения вызывает соответствующую подпрограмму обработки:

```

ЗАП3:
;*****
; ПОДПРОГРАММА ЗАПОЛНЕНИЯ МАССИВА КОДАМИ, ВЫРАБАТЫВАЕМЫМИ
; ВНЕШНЕЙ ПОДПРОГРАММОЙ ГЕНЕРАЦИИ КОДОВ (ПГК) С КОНТРОЛЕМ
; КОДОВ ПО ТАБЛИЦЕ УПРАВЛЯЮЩИХ КОДОВ (ТУК).
; ПАРАМЕТРЫ: (В,С)–АДРЕС НАЧАЛА МАССИВА, (D,E)–ДЛИНА МАСС-
; СИВА, (ST+1)–АДРЕС ПГК, (ST+2)–АДРЕС ТУК.
; ОЦЕНКА: ВРЕМЯ–31+(458+63*К)*N+(469+63*К)*M ТАКТОВ, ГДЕ
; N–КОЛИЧЕСТВО "ОБЫЧНЫХ" СИМВОЛОВ, M–КОЛИЧЕСТВО ВВЕДЕННЫХ
; "УПРАВЛЯЮЩИХ" СИМВОЛОВ, K– КОЛИЧЕСТВО ЭЛЕМЕНТОВ В ТУК;
; ДЛИНА–119 БАЙТ ; : ГЛУБИНА СТЕКА–4 БАЙТА.
;*****
207D C5          PUSH    B
; ПРОВЕРКА НА ДОСТИЖЕНИЕ ВЕРХНЕЙ ГРАНИЦЫ МАССИВА
207E 7A          ЦИКЛ4:  MOV    A,D
207F B3          ORA     E
2080 C28520      JNZ     PER5    ; ЕСЛИ НЕ ДОСТИГНУТА
2083 C1          POP     B
2084 C9          RET
; ОРГАНИЗАЦИЯ ВЫЗОВА ПГК
2085 219820      PER5:    LXI     H,PER6
2088 E5          PUSH    H        ; АДРЕС ВОЗВРАТА В СТЕК
                XTRN     D,E,3    ; (D,E)–АДРЕС ПГК
2096 EB          XCHG
2097 E9          PCHL
2098 F5          PER6:    PUSH    PSW    ; СОХРАНЕНИЕ ВВЕДЕННОГО СИМВОЛА
2099 EB          XCHG

```

	XTRN	D,E,3	
	XTRN	D,E,4	; (D,E) - АДРЕС ТУК
20B4 EB	XCHG		
20B5 F1	POP	PSW	; (A) - ВВЕДЕННЫЙ СИМВОЛ
			; ПРОВЕРКА КОДА НА ПРИНАДЛЕЖНОСТЬ К "УПРАВЛЯЮЩИМ"
20B6 5F	MOV	E,A	
20B7 E5	PUSH	H	; СОХРАНЕНИЕ АДРЕСА ТУК
20B8 7E	MOV	A,M	ЦИКЛ5:
20B9 BB	CMF	E	
20BA CAC820	JZ	PER7	; ЕСЛИ "УПРАВЛЯЮЩИЙ" СИМВОЛ
			; ПРОВЕРКА НА ОКОНЧАНИЕ ПРОСМОТРА ТУК
20BD FEFF	CFI	OFFH	
20BF CADF20	JZ	PER8	; ЕСЛИ КОНЕЦ ТУК
			; ПРОДОЛЖЕНИЕ ПРОСМОТРА ТУК
20C2 23	INX	H	
20C3 23	INX	H	
20C4 23	INX	H	
20C5 C3B820	JMP	ЦИКЛ5	
			; ВЫЗОВ ПОДПРОГРАММЫ ОБРАБОТКИ "УПРАВЛЯЮЩЕГО" СИМВОЛА
20C8 23	PER7:	INX	H
20C9 5E	MOV	E,M	
20CA 23	INX	H	
20CB 56	MOV	D,M	
20CC 217E20	LXI	H,ЦИКЛ4	; (H,L) - АДРЕС ВОЗВРАТА
20CF E3	XTHL		; (H,L) - АДРЕС ТУК
20D0 EB	XCHG		; (H,L) - АДРЕС ПОДПРОГРАММЫ
	XTRN	D,E,4	; (D,E) - ДЛИНА МАССИВА
20DE E9	RCHL		
			; ЗАПИСЬ СИМВОЛА В МАССИВ
20DF E1	PER8:	POP	H
20E0 7B	MOV	A,E	; (A) - КОД ВВЕДЕННОГО СИМВОЛА
20E1 EB	XCHG		; (D,E) - АДРЕС ТАБЛИЦЫ
20E2 02	STAX	B	
	XTRN	D,E,3	; (D,E) - ДЛИНА МАССИВА
20F0 03	INX	B	; МОДИФИКАЦИЯ АДРЕСА МАССИВА
20F1 1B	DCX	D	; МОДИФИКАЦИЯ ДЛИНЫ МАССИВА
20F2 C37E20	JMP	ЦИКЛ4	

В этой программе в отличие от предыдущих программ проверка на достижение границы заполняемого массива выполняется не в конце итерации, а в начале. Этот прием позволяет корректно анализировать ситуацию, когда во входном параметре задается длина заполняемого массива, равная нулю. После вызова подпрограммы генерации кода полученный код сравнивается со всеми управляющими кодами, содержащимися в таблице. Если очередное сравнение оказывается успешным, то следующие два байта таблицы помещаются в регистровую пару (H, L), в вершину стека загружается адрес начала итерации (адрес возврата) и выполняется команда RCHL. Тем самым осуществляется вызов подпрограммы обработки найденного управляющего символа. Параметры, передаваемые в подпрограммы обработки управляющих символов,

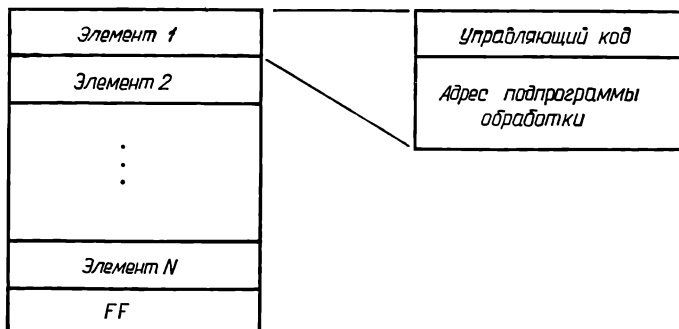


Рис. 5.2. Структура таблицы управляющих кодов

такие же, как и в программе ЗАПЗ, но в регистровых парах (В,С) и (D, E) содержатся не начальные, а текущие значения соответствующих переменных. Если ни одно из сравнений полученного кода с управляющими кодами таблицы не дает положительного результата, полученный код записывается в заполняемый массив, значения указателя и счетчика длины массива модифицируются и выполняется следующая итерация.

## 5.3. КОПИРОВАНИЕ МАССИВОВ

### 5.3.1. ПЕРЕСЫЛКА ИНФОРМАЦИИ С КОНЦА МАССИВА

Необходимость перемещения информации, начиная со старших адресов, т. е. с конца массива, возникает в том случае, когда адрес начала массива-приемника принадлежит области конца массива-источника (рис. 5.3, а). Программа пересылки имеет следующий вид:

```

КОПИ:
*****
; ПОДПРОГРАММА ПЕРЕСЫЛКИ МАССИВА С КОНЦА МАССИВА.
; ПАРАМЕТРЫ: (В,С) -АДРЕС МАССИВА-ПРИЕМНИКА (МНР), (D,E) -АД-
; РЕС МАССИВА-ИСТОЧНИКА (МИС), (H,L) -ДЛИНА МАССИВА.
; ОЦЕНКА: ВРЕМЯ-149+41*N ТАКОВ,ГДЕ N-ЧИСЛО ПЕРЕСЫЛАЕМЫХ
; БАЙТ; ДЛИНА-26 БАЙТ; ГЛУБИНА СТЕКА-6 БАЙТ.
*****
; СОХРАНЕНИЕ РЕГИСТРОВ В СТЕКЕ.

```

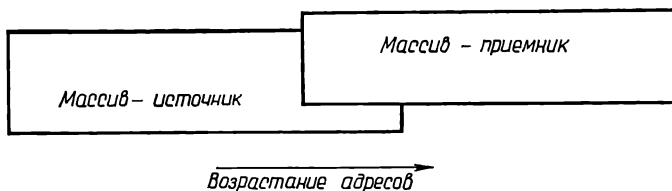
```

2000 C5          PUSH    B
2001 D5          PUSH    D
2002 E5          PUSH    H
                ; ОПРЕДЕЛЕНИЕ АДРЕСА КОНЦА МИС
2003 19          DAD     D

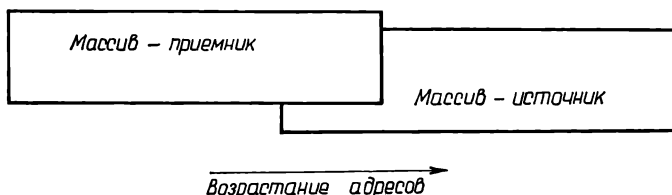
```

2004 2B	DCX	H	
2005 EB	XCHG		; (D,E) - АДРЕС КОНЦА МИС
	; ОПРЕДЕЛЕНИЕ АДРЕСА КОНЦА МПР		
2006 E1	POP	H	
2007 E5	PUSH	H	
2008 09	DAD	B	
2009 2B	DCX	H	; (H,L) - АДРЕС КОНЦА МПР
200A C1	POP	B	; (B,C) - ДЛИНА МАССИВА
200B C5	PUSH	B	
	; НЕПОСРЕДСТВЕННАЯ ПЕРЕСЫЛКА		
200C 1A	ЦИКЛ1: LDAX	D	
200D 77	MOV	M,A	
200E 0B	DCX	B	
200F 1B	DCX	D	
2010 2B	DCX	H	
2011 78	MOV	A,B	
2012 B1	ORA	C	
2013 C20C20	JNZ	ЦИКЛ1	
	; ВОССТАНОВЛЕНИЕ РЕГИСТРОВ		
2016 E1	POP	H	
2017 D1	POP	D	
2018 C1	POP	B	
2019 C9	RET		

*а*



*б*



**Рис. 5.3.** Схемы расположения массивов в памяти:

*а* — при пересылке с конца массива-источника; *б* — при пересылке с начала массива-источника

В программе предусмотрено сохранение значений входных параметров путем записи их в стек. Перед выполнением цикла непосредственного перемещения вычисляются адреса последних байтов массива-источника и



массива-приемника, что осуществляется путем сложения адреса начала массива и его длины командой DAD. Полученные адреса используются как начальные значения для указателей записи и чтения. Пересылка выполняется побайтно, после каждой пересылки значение указателей декрементируется. Таким образом, первым будет переписан последний байт заполняемого массива, затем — предпоследний и т. д.

### 5.3.2. ПЕРЕСЫЛКА ИНФОРМАЦИИ С НАЧАЛА МАССИВА

Если адрес начала массива-источника принадлежит области памяти массива-приемника, пересылку массива необходимо начинать с младших адресов (рис. 5.3, б). Программа пересылки имеет вид:

```

КОП2:
; *****
; ПОДПРОГРАММА ПЕРЕСЫЛКИ МАССИВА С НАЧАЛА МАССИВА.
; ПАРАМЕТРЫ: (B,C) -АДРЕС МПР, (D,E) -АДРЕС МИС, (H,L) -ДЛИНА
; МАССИВА.
; ОЦЕНКА: ВРЕМЯ-76+48*N ТАКТОВ, ГДЕ N-КОЛИЧЕСТВО ПЕРЕСЫ-
; ЛАЕМЫХ БАЙТ; ДЛИНА-17 БАЙТ; ГЛУБИНА СТЕКА-6 БАЙТ.
; *****
; СОХРАНЕНИЕ РЕГИСТРОВ В СТЕКЕ.
201A C5      PUSH    B
201B D5      PUSH    D
201C E5      PUSH    H
; НЕПОСРЕДСТВЕННАЯ ПЕРЕСЫЛКА
201D 1A      ЦИКЛ2: LDAX    D
201E 02      STAX    B
201F 03      INX     B
2020 13      INX     D
2021 2B      DCX     H
2022 7C      MOV     A,H
2023 B5      ORA     L
2024 C21D20  JNZ     ЦИКЛ2
; ВОССТАНОВЛЕНИЕ РЕГИСТРОВ
2027 E1      POP     H
2028 D1      POP     D
2029 C1      POP     B
202A C9      RET

```

Структура программы аналогична структуре программы КОП1, но отсутствуют фрагменты, в которых вычисляются адреса последних байтов массива-источника и массива-приемника, так как перемещение начинается с младших адресов и начальными значениями указателей являются значения соответствующих входных параметров. После каждой пересылки значения указателей инкрементируются. Если области памяти, занимаемые

массивом-источником и массивом-приемником, не пересекаются, перемещение выполняется корректно как программой КОП1, так и программой КОП2. Предпочтительнее пользоваться программой КОП2, так как она короче и выполняется быстрее.

## **5.4. ПОИСК В СТРУКТУРАХ**

### **5.4.1. ЗАДАЧИ ПОИСКА**

Под *поиском* понимается определение местоположения некоторой переменной или элемента данных в структуре. Практически любой обработке информации предшествуют процедуры поиска, с помощью которых осуществляется доступ к обрабатываемым данным. Процедуры поиска разнообразны. Ниже рассматриваются некоторые типичные задачи, решаемые в процессе поиска.

К поисковой процедуре сводится задача ассоциативного доступа к переменной, когда по известному значению переменной требуется определить ее адрес. При селективной обработке таблиц и списков поиск местоположения требуемых элементов осуществляется по известным значениям ключевых записей. При обработке символьных строк применяется поиск известной цепочки символов в заданной строке для обнаружения различных ключевых слов. Приведенные далее программы предназначены в основном для иллюстрации типичных поисковых процедур, но они могут быть использованы как элементарные при разработке реальных программных комплексов.

### **5.4.2. ПРОСТОЕ СРАВНЕНИЕ МАССИВОВ**

Эта операция выполняет побайтное сравнение заданных массивов, т. е. первый байт одного массива сравнивается с первым байтом второго массива, второй байт — со вторым и т. д. Если при очередном сравнении будет установлено неравенство байтов, дальнейшее сравнение прекращается, так как одного этого факта достаточно, чтобы сделать вывод о неравенстве массивов. Программа сравнения имеет следующий вид:

**СРВН1:**

```
*****  
; ПОДПРОГРАММА СРАВНЕНИЯ ДВУХ МАССИВОВ.  
; ВХОДНЫЕ ПАРАМЕТРЫ: (B,C) — АДРЕС НАЧАЛА ПЕРВОГО МАССИВА  
; (M1), (D,E)—АДРЕС НАЧАЛА ВТОРОГО МАССИВА (M2), (ST+1)—  
; ДЛИНА МАССИВОВ.
```

```

; ВЫХОДНОЙ ПАРАМЕТР: (A)=OFFH, ЕСЛИ МАССИВЫ РАВНЫ, (A)=0 В
; ПРОТИВНОМ СЛУЧАЕ.
; ОЦЕНКА: ВРЕМЯ - 136+82*N ТАКТОВ, ГДЕ N-КОЛИЧЕСТВО ЭЛЕ-
; МЕНТОР: ДЛИНА-36 БАЙТ; ГЛУБИНА СТЕКА - 6 БАЙТ.
;*****
; СОХРАНЕНИЕ ЗНАЧЕНИЙ ВХОДНЫХ ПАРАМЕТРОВ В СТЕКЕ
212A E5          PUSH    H
212B C5          PUSH    B
212C D5          PUSH    D
212D EB          XCHG     ; (H,L)- АДРЕС НАЧАЛА М2
                  LDSP    D,E,4 ; (D,E)- ДЛИНА МАССИВОВ
; ПРОВЕРКА ТЕКУЩЕЙ ДЛИНЫ МАССИВОВ НА НОЛЬ
2137 7B          ЦИКЛВ:  MOV    A,E
2138 B2          ORA     D
2139 3EFF         MVI     A,OFFH ; УСТАНОВКА ФЛАГА "РАВНЫ"
213B CA4B21       JZ      PER10 ; ЕСЛИ ДЛИНА - НОЛЬ
; СРАВНЕНИЕ ОЧЕРЕДНЫХ БАЙТОВ МАССИВОВ
213E 0A          LDAX    B
213F BE          CMP     H
2140 3E00         MVI     A,0 ; УСТАНОВКА ФЛАГА "НЕ РАВНЫ"
2142 C24B21       JNZ     PER10 ; ЕСЛИ БАЙТЫ НЕ РАВНЫ
; МОДИФИКАЦИЯ УКАЗАТЕЛЕЙ И ДЛИНЫ МАССИВОВ
2145 03          INC     B
2146 23          INC     H
2147 1B          DCR     D
2148 C33721       JMP     ЦИКЛВ
; ВОССТАНОВЛЕНИЕ РЕГИСТРОВ
214B D1          PER10:  POP    D
214C C1          POP     B
214D E1          POP     H
214E C9          RET

```

Непосредственное сравнение выполняется путем организации итерационного цикла. В начале каждой итерации проверяется значение счетчика текущей длины, и если оно равно нулю, то производится выход из цикла. Доступ к байтам, которые подлежат сравнению, осуществляется с помощью двух указателей, которые адресуют текущие байты соответственно первого и второго массивов. Если при очередном сравнении флаг Z равен нулю, т. е. байты не равны, выполнение цикла прекращается. В конце итерации модифицируются указатели массивов и счетчик текущей длины. Значение выходного параметра записывается в регистр (A) в теле цикла непосредственно перед командами условного перехода с помощью команды MVI, которая не изменяет состояние флагов микропроцессора. Этот прием позволяет достигнуть большей компактности программы, но увеличивает время ее выполнения.

В программе предусмотрено сохранение значений входных параметров. В некоторых случаях при выполнении

операции сравнения необходимо не только знать факт неравенства массивов, но и установить адреса байтов, по которым произошло несовпадение. Это можно обеспечить, если убрать из текста программы команды записи в стек и команды восстановления из стека, а также команду возврата, причем команды условного перехода, обеспечивающие выход из цикла, заменить командами условного возврата. Тогда в случае несовпадения массивов наряду с признаком в регистре (A) в регистровые пары (D, E) и (H, L) будут возвращены адреса несовпадающих байтов соответственно первого и второго массивов. Если необходимо обеспечить обработку несовпадающих байтов с помощью внешней подпрограммы, например для индикации их адресов и значений, следует команду условного перехода заменить командой условного вызова CNZ подпрограммы обработки.

#### 5.4.3. ПОИСК ОДНОБАЙТНОГО КОДА В МАССИВЕ

Задача состоит в определении адреса байта при известных его значениях и границах области памяти, в которой он может быть расположен. Программа поиска имеет вид:

```

ПСК1:
;*****
; ПОДПРОГРАММА ПОИСКА КОДА В МАССИВЕ.
; ВХОДНЫЕ ПАРАМЕТРЫ: (C) - ЗНАЧЕНИЕ КОДА, (D,E) - АДРЕС НАЧАЛА
; МАССИВА, (SI+1) - ДЛИНА МАССИВА.
; ВЫХОДНЫЕ ПАРАМЕТРЫ: (H,L) - АДРЕС НАЙДЕННОГО БАЙТА, ЕСЛИ
; ПОИСК УСПЕШНЫЙ, (H,L) = 0FFFFH В ПРОТИВНОМ СЛУЧАЕ.
; ОЦЕНКА: ВРЕМЯ = 24 + 96 * N ТАКТОВ, ГДЕ N - ДЛИНА МАССИВА;
; ДЛИНА - 30 БАЙТ.
;*****
; ПРОВЕРКА НА ДОСТИЖЕНИЕ ВЕРХНЕЙ ГРАНИЦЫ МАССИВА
20F5 EB          XCHG
                ЦИКЛ6: LDSP      D,E,1    ; (D,E) - ДЛИНА МАССИВА
20FF 7B          MOV      A,E
2100 B2          ORA      D
2101 CA0C21      JZ       PER9           ; ЕСЛИ КОНЕЦ МАССИВА
                ; СРАВНЕНИЕ ОЧЕРЕДНОГО БАЙТА
2104 7E          MOV      A,M
2105 B9          CMP      C
2106 C8          RZ              ; ПРИ УДАЧНОМ ПОИСКЕ
                ; ПРОДОЛЖЕНИЕ ПОИСКА
2107 23          INX      H
2108 1B          DCX      D
2109 C3F620      JMP      ЦИКЛ6
                ; ВОЗВРАТ ПРИ БЕЗУСПЕШНОМ ПОИСКЕ
210C 21FFFF      PER9: LXI      H, 0FFFFH
210F C9          RET

```

Поиск заданного байта производится с помощью итеративного цикла, в котором исходный байт последовательно сравнивается с каждым байтом массива. При первом же успешном сравнении осуществляется возврат из программы. Адрес байта, подлежащего сравнению на текущей итерации, располагается в регистровой паре (H, L), что автоматически обеспечивает требуемое значение выходного параметра при успешном поиске. Выход из цикла выполняется при нулевом значении счетчика текущей длины, контролируемого в начале каждой итерации. Программа позволяет обнаружить только первое вхождение искомого кода в заданный массив. Если требуется определить адрес всех байтов, содержащих заданный код, или подсчитать их количество, нужно циклически вызывать указанную программу до тех пор, пока в регистровую пару (H, L) не будет возвращен признак неудачного поиска.

#### 5.4.4. ПОИСК ПОСЛЕДОВАТЕЛЬНОСТИ КОДОВ В МАССИВЕ

Задача формулируется следующим образом: имеются два массива *A* и *B*, причем длина массива *A* больше длины массива *B*; необходимо определить местоположение копии массива *B* в массиве *A* (если она имеется). Задачу в такой постановке приходится решать, когда требуется отыскать в некоторой строке заданную цепочку символов.

Алгоритм выполнения данной поисковой процедуры заключается в скользящем сравнении массива *B* с массивом *A*: массив *B* сравнивается с начальным фрагментом массива *A*, начинающегося с первого байта массива *A*, длина которого равна длине массива *B*, затем сравнение производится с фрагментом массива *A*, начинающимся со второго байта этого массива, и т. д. (рис. 5.4). Программа поиска имеет вид:

```

СРВН2:
;*****
; ПОДПРОГРАММА ПОИСКА МАССИВА "B" В МАССИВЕ "A".
; ВХОДНЫЕ ПАРАМЕТРЫ: (P,C) - АДРЕС НАЧАЛА МАССИВА "A", (P,E)
; - АДРЕС НАЧАЛА МАССИВА "B", (ST+1) - ДЛИНА МАССИВА "B",
; (ST+2) - ДЛИНА МАССИВА "A".
; ВЫХОДНЫЕ ПАРАМЕТРЫ: (H,L) - АДРЕС ПЕРВОГО БАЙТА КОПИИ
; МАССИВА "B" В МАССИВЕ "A" ЕСЛИ ОНА СУЩЕСТВУЕТ, (H,L) - С-
; В ПРОТИВНОМ СЛУЧАЕ.
; ОЦЕНКА: ВРЕМЯ-429+(206+82*N)*M ТАКТОВ, ГДЕ N-КОЛИЧЕСТВО
; ЭЛЕМЕНТОВ В МАССИВЕ "B", M-КОЛИЧЕСТВО ЭЛЕМЕНТОВ В MAS-
; СИВЕ "A"; ДЛИНА-81 БАЙТ, ГЛУБИНА СТЕКА-6 БАЙТ.
;*****

```

```

; СОХРАНЕНИЕ ВХОДНЫХ ПАРАМЕТРОВ В СТЕКЕ
214F D5      PUSH    D
2150 C5      PUSH    B

; ОПРЕДЕЛЕНИЕ ЧИСЛА СРАВНЕНИЙ
                XTRN    D,E,3      ; (D,E)-ДЛИНА МАССИВА "А"
                XTRN    B,C,4      ; (B,C)-ДЛИНА МАССИВА "В"
216B 79      MOV     A,C
216C 93      SUB     E
216D 6F      MOV     L,A
216E 78      MOV     A,B
216F 9A      SBB     D
2170 67      MOV     H,A      ; (H,L)-ЧИСЛО СРАВНЕНИЙ
2171 D5      PUSH    D      ; СОХРАНЕНИЕ ДЛИНЫ МАССИВА "А"
                XTRN    D,E,4      ; (D,E)-АДРЕС МАССИВА "А"
                XTRN    B,C,5      ; (B,C)-АДРЕС МАССИВА "В"

; СКОЛЬЗЯЩЕЕ СРАВНЕНИЕ МАССИВОВ
218C CD2A21  ЦИКЛ9: CALL  CРВН1
218F B7      ORA     A
2190 C29D21  JNZ     ПЕР12      ; ЕСЛИ МАССИВЫ РАВНЫ
2193 03      INX     B
2194 2B      DCX     H
2195 7D      MOV     A,L
2196 B4      ORA     H
2197 C28C21  JNZ     ЦИКЛ9      ; ЕСЛИ СРАВНЕНИЯ НЕОКОНЧЕНО
219A C39F21  JMP     ПЕР11

; ЗАГРУЗКА ВЫХОДНОГО ПАРАМЕТРА ПРИ УДАЧНОМ ПОИСКЕ
219D 69      ПЕР12: MOV    L,C
219E 60      MOV     H,B

; ВОССТАНОВЛЕНИЕ ВХОДНЫХ ПАРАМЕТРОВ
219F D1      ПЕР11: POP    D
21A0 C1      POP     B
21A1 D1      POP     D
21A2 C9      RET

```

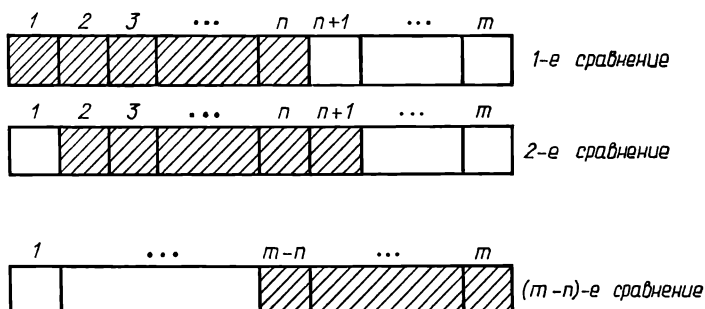


Рис. 5.4. Схема скользящего сравнения двух массивов.

Необходимое число сравнений для полного поиска определяется разностью длин массивов  $A$  и  $B$ , которая вычисляется в начальном фрагменте программы. При извлечении значений длин массивов из стека в макрокоман-

де XTRN номер слова в стеке указывается с учетом того, что в начале программы были произведены две загрузки в стек.

Скольльзящее сравнение массивов организовано в виде итеративного цикла. Сравнение массива *B* с фрагментом массива *A* выполняется с помощью подпрограммы SRBH1 (она описана на с. 225—226). После каждого вызова подпрограммы анализируется признак результата сравнения, который устанавливается подпрограммой SRBH1 в регистре (A). Если сравнение было успешным, т. е. копия массива *B* найдена в массиве *A*, формируется соответствующее значение выходного параметра и осуществляется возврат из программы. В противном случае поиск продолжается, адрес начала очередного фрагмента массива *A* инкрементируется, а в конце каждой итерации число сравнений декрементируется. Когда оно достигает нулевого значения, выполнение цикла скольльзящего сравнения прекращается и осуществляется возврат из программы. Полученное нулевое значение используется в качестве индикатора неудачного поиска без дополнительной установки выходного параметра. В программе обеспечено сохранение значений всех входных параметров, передаваемых через регистры и через стек.

#### 5.4.5. ПОИСК ЭЛЕМЕНТА ТАБЛИЦЫ ПО КЛЮЧУ

Ниже описывается процедура поиска элемента простой таблицы, формат которой приведен на рис. 5.5. В таблице, состоящей из *m* элементов длиной *n* байт, требуется определить адрес элемента, первый байт которого совпадает с заданным. Поиск заключается в просмотре первых байтов всех элементов и сравнении их значений с заданным. Программа поиска имеет вид:

```
ПСКЗ:
;*****
; ПОДПРОГРАММА ПОИСКА ЭЛЕМЕНТА ТАБЛИЦЫ ПО КЛЮЧУ.
; ВХОДНЫЕ ПАРАМЕТРЫ: (C) — КОД КЛЮЧА ИСКОМОГО ЭЛЕМЕНТА,
; (D) — КОЛИЧЕСТВО ЭЛЕМЕНТОВ В ТАБЛИЦЕ, (E) — ДЛИНА ЭЛЕМЕНТА,
; (ST+1) — АДРЕС НАЧАЛА ТАБЛИЦЫ.
; ВЫХОДНЫЕ ПАРАМЕТРЫ: (H,L) — АДРЕС ПЕРВОГО БАЙТА НАЙДЕН-
; НОГО ЭЛЕМЕНТА ПРИ УСПЕШНОМ ПОИСКЕ, (H,L)=OFFFH В ПРО-
; ТИВНОМ СЛУЧАЕ.
; ОЦЕНКА: ВРЕМЯ=101+41*N ТАКТОВ, ГДЕ N — КОЛИЧЕСТВО ЭЛЕ-
; МЕНТОВ В ТАБЛИЦЕ; ДЛИНА — 29 БАЙТ.
;*****
; СРАВНЕНИЕ КЛЮЧА ЭЛЕМЕНТА ТАБЛИЦЫ С ЗАДАНЫМ
```

		LDSP	D, E, 1	
211A EB		XCHG		; (H, L) — АДРЕС НАЧАЛА ЭЛЕМЕНТА
211B 42		MOV	B, D	; (B) — КОЛИЧЕСТВО ЭЛЕМЕНТОВ
211C 1600		MVI	D, 0	; (D, E) — ДЛИНА ЭЛЕМЕНТА
211E 7E	ЦИКЛ7:	MOV	A, M	
211F B9		CMF	C	
2120 C8		RZ		; ЕСЛИ КЛЮЧИ СОВПАЛИ
				; МОДИФИКАЦИЯ УКАЗАТЕЛЯ ТАБЛИЦЫ И СЧЕТЧИКА ЭЛЕМЕНТОВ.
2121 19		DAD	D	; (H, L) — АДРЕС СЛЕДУЮЩЕГО ЭЛЕМЕНТА
2122 05		DCR	B	
2123 C21E21		JNZ	ЦИКЛ7	; ЕСЛИ ЭЛЕМЕНТ НЕ ПОСЛЕДНИЙ
				; ВОЗВРАТ ПРИ НЕУДАЧНОМ ПОИСКЕ
2126 21FFFF		LXI	H, 0FFFFH	
2129 C9		RET		

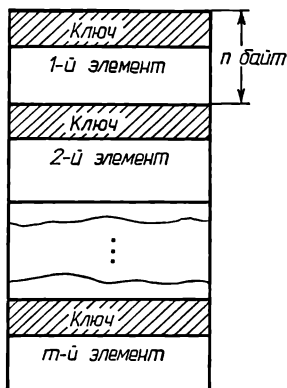


Рис. 5.5. Структура таблицы

Для организации цикла сравнения используются две рабочие переменные — указатель таблицы, который содержит адрес первого байта очередного элемента, и счетчик, содержащий количество непросмотренных элементов. На каждой итерации производится сравнение значения байта, адресуемого указателем, с заданным кодом. Если они равны, то поиск завершается и осуществляется возврат из программы. При этом текущее значение указателя таблицы возвращается в качестве выходного параметра. В противном случае вычисляется новое значение указателя путем добавления к его текущему значению длины элемента. Хотя величина длины элемента помещается в одном байте, для ее хранения выделяется регистровая пара. Это сделано для упрощения модификации указа-



теля, которая выполняется одной командой DAD. В конце итерации значение счетчика декрементируется. После просмотра всех элементов (факт окончания просмотра определяется нулевым значением счетчика) в выходной параметр записывается признак неудачного поиска и выполняется возврат из программы. В программе существуют ограничения на количество  $m$  элементов в структуре и длину  $n$  элемента ( $m, n \leq 256$ , поскольку для хранения как числа  $m$ , так и числа  $n$  выделяется по одному байту).

#### 5.4.6. ПОИСК ЭЛЕМЕНТА СПИСКА

Поиск в списке предполагает определение адреса элемента, содержимое поля ключа которого равно заданному. Структура списка приведена на рис. 5.1. Для решения задачи необходимо просмотреть весь список, сравнивая при этом содержимое поля ключа каждого элемента с заданным значением. Программа поиска имеет следующий вид:

```

ПСКСП:
;*****
; ПОДПРОГРАММА ПОИСКА ЭЛЕМЕНТА В СПИСКЕ ПО КЛЮЧУ.
; ВХОДНЫЕ ПАРАМЕТРЫ: (В,С) - АДРЕС НАЧАЛА СПИСКА, (Е) - ЗНА-
; ЧЕНИЕ КЛЮЧА.
; ВЫХОДНЫЕ ПАРАМЕТРЫ: (Н,Л) - АДРЕС НАЙДЕННОЙ ЗАПИСИ ПРИ
; УСПЕШНОМ ПОИСКЕ, (Н,Л)=OFFFH В ПРОТИВНОМ СЛУЧАЕ, (В,С)
; - АДРЕС ПРЕДЫДУЩЕЙ ЗАПИСИ В СПИСКЕ.
; ОЦЕНКА: ВРЕМЯ - 10+93*N ТАКОВ, ГДЕ N-КОЛИЧЕСТВО ЭЛЕ-
; МЕНТОВ В СПИСКЕ; ДЛИНА - 27 БАЙТ.
;*****
2290 210000      LXI      H,0
; СРАВНЕНИЕ ПОЛЯ КЛЮЧА С ЗАДАНЫМ ЗНАЧЕНИЕМ
2293 03          ЦИКЛ10: INX      B
2294 03          INX      B          ; (В,С) - АДРЕС ПОЛЯ КЛЮЧА
2295 0A          LDAX     B
2296 BB          CMP      E
2297 0B          DCS      B
2298 0B          DCS      B
                XCHR     H,B
229D C8          RZ              ; ЕСЛИ КЛЮЧ СОВПАЛ С ЗАДАНЫМ
; ПЕРЕХОД К СЛЕДУЮЩЕМУ ЭЛЕМЕНТУ
                MOV      C,H
229E 4E          INX      H
229F 23          MOV      B,H          ; (В,С)-АДРЕС СЛЕДУЮЩЕГО ЭЛЕМЕНТА
22A0 46          DCS      H
22A1 2B          ; ПРОВЕРКА НА ЗАВЕРШЕНИЕ ПРОСМОТРА СПИСКА
                MOV      A,C          ; ПРОВЕРКА ПОЛЯ СВЯЗИ НА НОЛЬ
22A2 79          ORA      B
22A3 B0          JNZ      ЦИКЛ10      ; ЕСЛИ ПРОСМОТР НЕ ОКОНЧЕН
22A4 C29322      ; ВОЗВРАТ ПРИ НЕУДАЧНОМ ПОИСКЕ
                LXI      H,OFFFH
22A7 21FFFF      RET
22AA C9

```

В программе просмотр элементов списка выполняется с помощью итеративного цикла. Для полного определения местоположения элемента в однонаправленном списке необходимо знать адрес не только данного элемента, но и предыдущего, так как в однонаправленных списках отсутствуют адресные ссылки назад. С этой целью в программу вводятся два указателя: один содержит адрес начала текущего элемента и расположен в регистровой паре (В, С), второй — адрес начала предыдущего элемента и помещается в регистровой паре (Н, L). Первоначально значение первого указателя равно адресу начала списка, а второго — нулю. На каждой итерации анализируется поле ключа одного элемента. После этого указатели меняются местами, и если обнаруживается, что поле ключа текущего элемента равно заданному, осуществляется возврат из программы. В противном случае в первый указатель загружается значение поля связи текущего элемента. Если новое значение первого указателя равно нулю (а это означает окончание списка), выполнение цикла прекращается, в регистровую пару (Н, L) загружается признак неудачного поиска и выполняется возврат из программы.

## **5.5. ПРЕОБРАЗОВАНИЯ СТРУКТУР**

### **5.5.1. ЗАДАЧИ ПРЕОБРАЗОВАНИЯ**

Рассмотрим ряд программ, реализующих процедуры простого преобразования структур данных. Эти преобразования касаются только количественных характеристик структур и формы представления элементов. Данные программы осуществляют перевод содержимого элементов некоторой структуры из одной системы кодирования в другую, удаление элемента из структуры и вставку его в структуру.

Необходимость перекодировки возникает в том случае, когда различные программы, входящие в один программный комплекс, по-разному интерпретируют данные одного содержания. Чаще всего такая потребность возникает при обработке символьной информации в операциях ввода-вывода, когда внутрисистемное представление символов определяется одними правилами, а некоторое внешнее устройство интерпретирует те же символы по другим правилам. Например, программы внутренней об-

работки символов требуют их представления в коде КОИ-7, а вывод этих символов на индикаторное табло необходимо осуществлять в сегментном коде. Аналогичная ситуация может возникнуть и при вводе информации с внешнего устройства. Процедуры вставки и удаления элементов рассматриваются не только для динамических структур (списков), но и для статических (массивов). К простому преобразованию массивов приходится чаще всего прибегать в том случае, когда в массиве содержится символьная информация, подлежащая редактированию.

### 5.5.2. ПРЯМАЯ ПЕРЕКОДИРОВКА СТРОКИ

Пусть имеется некоторая строка, представляющая собой массив символов, закодированный в системе кодов *A* (входная система кодирования). Необходимо получить строку, содержащую такую же последовательность символов, но закодированную в системе кодов *B* (выходная система кодирования). Если входная и выходная системы кодирования различаются незначительно (небольшим количеством символов), перекодировку можно осуществить путем последовательного просмотра входной строки и селекции различающихся массивов с их последующей заменой. В более общем случае перекодировка производится с помощью специальной таблицы. Таблица составляется следующим образом: в *i*-й элемент таблицы помещается код символа в выходной системе кодирования, соответствующий символу, который кодируется кодом *i* во входной системе кодирования. Таким образом, код символа во входной системе кодирования можно использовать как индекс для нахождения соответствующего кода выходной системы кодирования в таблице (схема перекодировки приведена на рис. 5.6). Программа перекодировки имеет вид:

```

ППКОД:
; *****
; ПОДПРОГРАММА ПРЯМОЙ ПЕРЕКОДИРОВКИ СТРОКИ.
; ПАРАМЕТРЫ: (B,C) - АДРЕС ВХОДНОЙ СТРОКИ, (D,E) - АДРЕС
; ТАБЛИЦЫ ПЕРЕКОДИРОВКИ, (ST+1) - АДРЕС ВЫХОДНОЙ СТРОКИ,
; (ST+2) - ДЛИНА ВХОДНОЙ СТРОКИ.
; ОЦЕНКА: ВРЕМЯ - 10+385*N ТАКТОВ, ГДЕ N - ДЛИНА СТРОКИ;
; ДЛИНА-67 БАЙТ.
; *****
; ФОРМИРОВАНИЕ ИНДЕКСА ПО КОДУ ВХОДНОЙ СТРОКИ

```

```

21A3 2600      MVI     H,0
21A5 0A        LDAX    B

```

```

21A6 6F      MOV     L,A      ; (H,L)-ИНДЕКС
                ; ПОЛУЧЕНИЕ СООТВЕТСТВУЮЩЕГО КОДА В ВЫХОДНОЙ СТРОКЕ
21A7 19      DAD     D
                XTRN    B,C,1  ; (B,C)-АДРЕС ВЫХОДНОЙ СТРОКИ
21B5 7E      MOV     A,M
21B6 02      STAX    B
                ; МОДИФИКАЦИЯ УКАЗАТЕЛЕЙ
21B7 03      INX     B
                XTRN    B,C,1  ; (B,C)-АДРЕС ВХОДНОЙ СТРОКИ
21C5 03      INX     B
                XTRN    B,C,2  ; (B,C)-ДЛИНА ВХОДНОЙ СТРОКИ
21D3 0B      DCX     B
                ; ПРОВЕРКА НА ОКОНЧАНИЕ ПРОСМОТРА ВХОДНОЙ СТРОКИ
21D4 79      MOV     A,C
21D5 B0      ORA     B
                XTRN    B,C,2  ; (B,C)-АДРЕС ВХОДНОЙ СТРОКИ
21E3 C2A321  JNZ     ППКОД    ; ЕСЛИ ВХОДНАЯ СТРОКА НЕ ИСЧЕРПАНА
21E6 C9      RET

```

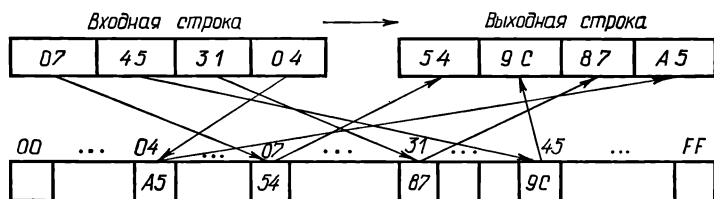


Таблица перекодировки

Рис. 5.6. Схема перекодировки символов входной и выходной строк

Программа представляет итеративный цикл, на каждой итерации которого перекодируется один байт. Используются два указателя — для входной и выходной строк, а также счетчик текущей длины для контроля окончания цикла. Значение очередного байта выходной строки определяется следующим образом: содержимое очередного байта входной строки добавляется к адресу начала таблицы, затем содержимое ячейки памяти с полученным адресом переписывается в текущий байт выходной строки.

### 5.5.3. ОБРАТНАЯ ПЕРЕКОДИРОВКА СТРОКИ

Эта задача отличается от предыдущей тем, что таблица перекодировки содержит коды символов не в выходной, а во входной системе кодирования. Обратную перекодировку можно свести к прямой, если составить новую таблицу. Как правило, в МП системах перекодировку

символов приходится проводить как в прямом, так и в обратном направлениях, и наличие двух таблиц, содержащих одну и ту же информацию, является избыточным. При обратной перекодировке значение каждого кода входной строки отыскивается в таблице, а в соответствующее место выходной строки записывается индекс найденного элемента таблицы. Программа перекодировки представлена ниже:

```

ОПКОД:
;*****
; ПОДПРОГРАММА ОБРАТНОЙ ПЕРЕКОДИРОВКИ СТРОКИ.
; ПАРАМЕТРЫ: (B,C) - АДРЕС ВХОДНОЙ СТРОКИ, (D,E) - АДРЕС
; ТАБЛИЦЫ ПЕРЕКОДИРОВКИ, (ST+1) - АДРЕС ВЫХОДНОЙ СТРОКИ,
; (ST+2) - ДЛИНА ВХОДНОЙ СТРОКИ.
; ОЦЕНКА: ВРЕМЯ-10+12685*М ТАКТОВ, ГДЕ М - ДЛИНА ВЫХОДНОЙ
; СТРОКИ; ДЛИНА - 79 БАЙТ; ГЛУБИНА СТЕКА - 8 БАЙТ.
;*****
; ПОИСК ЭЛЕМЕНТА ВХОДНОЙ СТРОКИ В ТАБЛИЦЕ ПЕРЕКОДИРОВКИ

21E7 C5      PUSH    B
21E8 69      MOV     L,C
21E9 60      MOV     H,B
21EA 4E      MOV     C,M      ; (C)-КОД ЭЛЕМЕНТА ВХОДНОЙ СТРОКИ
21EB D5      PUSH    D
21EC 217F00  LXI     H,7FH
21EF E5      PUSH    H
21F0 CDF520  CALL    ПСК1
21F3 C1      POP     B
21F4 D1      POP     D
21F5 7D      MOV     A,L
21F6 93      SUB     E
21F7 6F      MOV     L,A
21F8 7C      MOV     A,H
21F9 9A      SBB     D
21FA 67      MOV     H,A      ; (L)-КОД ЭЛЕМЕНТА ВЫХОДНОЙ СТРОКИ
; ЗАПИСЬ ЭЛЕМЕНТА В ВЫХОДНУЮ СТРОКУ
; LDSP B,C,2 ; (B,C)-АДРЕС ВЫХОДНОЙ СТРОКИ
2204 7D      MOV     A,L
2205 02      STAX    B
; МОДИФИКАЦИЯ УКАЗАТЕЛЕЙ
2206 03      INX     B
; XTRN B,C,2
2214 C1      POP     B      ; (B,C)-АДРЕС ВХОДНОЙ СТРОКИ
2215 03      INX     B
; ПРОВЕРКА НА ОКОНЧАНИЕ ПРОСМОТРА ВХОДНОЙ СТРОКИ
; XTRN B,C,2 ; (B,C)-ДЛИНА ВХОДНОЙ СТРОКИ
2223 0B      DCX     B
2224 79      MOV     A,C
2225 B0      ORA     B
; XTRN B,C,2 ; (B,C)-АДРЕС ВХОДНОЙ СТРОКИ
2233 C2E721  JNZ     ОПКОД ; ЕСЛИ ВХОДНАЯ СТРОКА НЕ ИСЧЕРПАНА
2236 C9      RET

```

Как и в предыдущей программе, для организации итеративного цикла используются два указателя —

для входной и выходной строк, а также счетчик текущей длины массива. Поиск элемента таблицы, содержащего очередной код входной строки, осуществляется с помощью подпрограммы ПСК1, которая находит заданный код в определенной области памяти и возвращает его абсолютный адрес в регистровую пару (H, L). Для получения индекса из этого абсолютного адреса вычитается адрес начала таблицы, затем значение индекса записывается в выходную строку.

#### 5.5.4. УДАЛЕНИЕ ФРАГМЕНТА МАССИВА

Под удалением фрагмента массива понимается удаление содержимого некоторой области памяти, входящей в данный массив, при этом длина массива уменьшается на длину удаляемого фрагмента. Пусть имеется массив с начальным адресом A и конечным адресом D, из которого необходимо удалить фрагмент, начинающийся с адреса B и оканчивающийся адресом C ( $A < B < C < D$ ). Как видно из рис. 5.7, задача удаления фрагмента сводится к перемещению содержимого области памяти, ограниченной адресами C и D (остаток массива), в область памяти, начинающуюся с адреса B (адрес первого удаляемого байта). Программа удаления имеет вид:

```

УДАЛ:
; *****
; ПОДПРОГРАММА УДАЛЕНИЯ ФРАГМЕНТА МАССИВА.
; ПАРАМЕТРЫ: (B,C) - АДРЕС НАЧАЛА УДАЛЯЕМОГО ФРАГМЕНТА (УФ)
; (D,E) - АДРЕС КОНЦА УФ, (ST+1) - АДРЕС КОНЦА МАССИВА.
; ОЦЕНКА: ВРЕМЯ-268+48*N ТАКОВ, ГДЕ N - ДЛИНА "ОСТАТКА"
; МАССИВА; ДЛИНА 21 БАЙТ.
; *****
; ПОДГОТОВКА ПАРАМЕТРОВ ДЛЯ ПОДПРОГРАММЫ ПЕРЕСЫЛКИ
                XTRN    B,C,1    ; (B,C) - АДРЕС КОНЦА МАССИВА
2244 79          MOV     A,C
2245 93          SUB     E
2246 6F          MOV     L,A
2247 78          MOV     A,B
2248 9A          SBB     D
2249 67          MOV     H,A      ; (H,L) - ДЛИНА "ОСТАТКА" МАССИВА
                XTRN    B,C,1    ; (B,C) - АДРЕС НАЧАЛА УФ
; ВЫЗОВ ПОДПРОГРАММЫ ПЕРЕСЫЛКИ
2257 C31A20     JMP     КОП2

```

В программе перемещение информации в массиве осуществляется с помощью подпрограммы КОП2. Команды, предшествующие обращению к этой подпрограмме, формируют необходимые значения входных параметров.

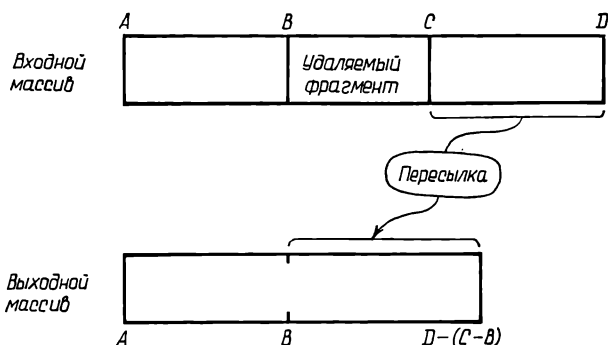


Рис. 5.7. Схема удаления фрагмента массива

Адресом начала массива-источника является адрес последнего удаляемого байта, адресом начала массива-приемника — адрес первого удаляемого байта. Длина перемещаемой области определяется как разность между адресами последнего байта массива и последнего удаляемого байта. Использование подпрограммы КОП2, которая выполняет перемещение с младших адресов в отличие от подпрограммы КОП1, выполняющей перемещение со старших адресов, объясняется необходимостью анализа ситуации, когда длина удаляемого фрагмента меньше длины остатка массива. При этом адрес конца массива-приемника будет больше адреса начала массива-источника.

### 5.5.5. ВСТАВКА ФРАГМЕНТА В МАССИВ

Пусть в некоторый массив, ограниченный адресами А и С, необходимо, начиная с адреса В, вставить содержимое области памяти, не принадлежащей этому массиву, при этом вся информация исходного массива сохраняется. Решение задачи осуществляется в два этапа. Сначала необходимо освободить в массиве место под вставляемый фрагмент. Для этого часть массива, ограниченная адресами В и С, сдвигается в памяти в сторону возрастания адресов на количество байтов, равное длине вставки, затем в освобожденную область копируется содержимое вставляемого фрагмента (рис. 5.8). Программа вставки имеет вид:

```

ВСТАВ:
;*****
; ПОДПРОГРАММА ВСТАВКИ ФРАГМЕНТА В МАССИВ.
; ПАРАМЕТРЫ: (В,С) - АДРЕС МЕСТА ВСТАВКИ, (D,E) - АДРЕС
; КОНЦА МАССИВА, (ST+1)-АДРЕС ПЕРВОГО ВСТАВЛЯЕМОГО БАЙТА,
; (ST+2) - ДЛИНА ВСТАВКИ.
; ОЦЕНКА: ВРЕМЯ ВЫПОЛНЕНИЯ -- 643+48*N+48*M ТАКТОВ, ГДЕ N-
; ДЛИНА "ОСТАТКА" МАССИВА, M-ДЛИНА ВСТАВКИ; ДЛИНА-36 БАЙТ;
; ГЛУБИНА СТЕКА-2 БАЙТА.
;*****
; ПЕРЕСЫЛКА ЧАСТИ МАССИВА ДЛЯ ОБЕСПЕЧЕНИЯ ВСТАВКИ
225A 7B      MOV     A,E
225B 91      SUB     C
225C 5F      MOV     E,A
225D 7A      MOV     A,D
225E 98      SUB     B
225F 3C      INR     A
2260 57      MOV     D,A      ; (D,E)-ДЛИНА "ОСТАТКА" МАССИВА
2261 EB      XCHG    LDSP    D,E,2 ; (D,E)-ДЛИНА ВСТАВКИ
226B EB      XCHG
226C D5      PUSH    D      ; (ST)-ДЛИНА "ОСТАТКА" МАССИВА
226D 09      DAD     B
226E 59      MOV     E,C
226F 50      MOV     D,B      ; (D,E)-АДРЕС ОБЛАСТИ ИСТОЧНИКА
2270 4D      MOV     C,L
2271 44      MOV     B,H      ; (B,C)-АДРЕС ОБЛАСТИ ПРИЕМНИКА
2272 E1      POP     H      ; (H,L)-ДЛИНА "ОСТАТКА" МАССИВА
2273 CD0020  CALL    КОП1
; ПЕРЕСЫЛКА ВСТАВКИ В ОСВОБОЖДЕННУЮ ОБЛАСТЬ
XCHR D,B      ; (B,C)-АДРЕС МЕСТА ВСТАВКИ
LDSP D,F,2
2283 EB      XCHG      ; (H,L)-ДЛИНА ВСТАВКИ
LDSP D,E,1    ; (D,E)-АДРЕС НАЧАЛА ВСТАВКИ
228D C30020  JMP     КОП1

```

Первый фрагмент программы обеспечивает «раздвижку» массива путем перемещения содержимого области памяти, ограниченной адресом вставки и адресом последнего байта массива, в область памяти, начинающуюся с адреса, значение которого равно сумме адреса вставки и длины вставляемого фрагмента. Перемещение выполняется подпрограммой КОП1. Во втором фрагменте обеспечивается заполнение освобожденной области содержимым вставляемого фрагмента. Здесь вторично вызывается подпрограмма КОП1, для которой массивом-источником является вставляемый фрагмент, а адресом начала массива-приемника — адрес вставки.

В данной программе в отличие от предыдущей перемещение информации выполняется с помощью подпрограммы КОП1, а не КОП2. Это объясняется тем, что если длина вставляемого фрагмента меньше длины остатка



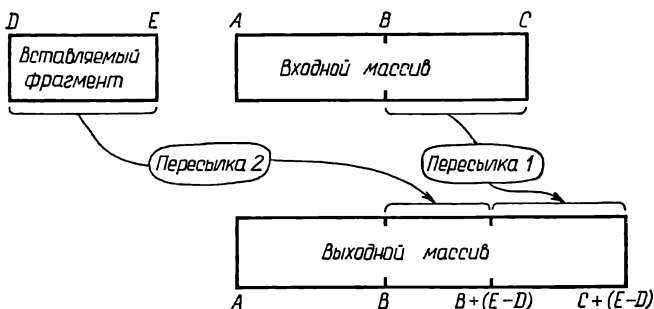


Рис. 5.8. Схема вставки фрагмента массива

массива (области памяти, ограниченной адресами В и С), то при выполнении «раздвижки» адрес начала массива-приемника оказывается меньше адреса конца массива-источника, и пересылку надо производить со старших адресов. Во втором фрагменте безразлично, какую подпрограмму применять: КОП1 или КОП2. Подпрограмма КОП1 используется только с целью унификации, хотя для уменьшения времени выполнения более целесообразно использовать подпрограмму КОП2.

### 5.5.6. ВСТАВКА ЭЛЕМЕНТА В СПИСОК

Модификация списка в отличие от модификации массива не требует физического перемещения информации; достаточно только изменить значение полей связи соответствующих элементов списка. Для выполнения вставки элемента в список необходимо указать точку входа в него, т. е. адрес начала списка и место вставки в список, идентифицируемое по заданному значению ключа того элемента, после которого необходимо произвести вставку.

Алгоритм решения этой задачи состоит в следующем. Последовательно просматривается весь список для нахождения элемента, значение поля ключа которого равно заданному, затем содержимое поля связи найденного элемента копируется в поле связи вставляемого элемента, а туда записывается адрес вставляемого элемента (рис. 5.9). Программа вставки имеет вид:

# ВСТСП:

```

;*****
; ПОДПРОГРАММА ВКЛЮЧЕНИЯ ЭЛЕМЕНТА В СПИСОК.
; ВХОДНЫЕ ПАРАМЕТРЫ: (B,C)-АДРЕС НАЧАЛА СПИСКА, (E)-ЗНА-
; ЧЕНИЕ КЛЮЧА ЭЛЕМЕНТА, ПОСЛЕ КОТОРОГО ПРОИЗВОДИТСЯ ВКЛЮ-
; ЧЕНИЕ, (ST+1)-АДРЕС ВКЛЮЧАЕМОГО ЭЛЕМЕНТА.
; ВЫХОДНЫЕ ПАРАМЕТРЫ: (A)=0-ВКЛЮЧЕНИЕ УДАЧНО, (A)=OFFH -
; НЕУДАЧНО.
; ОЦЕНКА: ВРЕМЯ-162+93*N ТАКОВ, ГДЕ N - КОЛИЧЕСТВО ЭЛЕ-
; МЕНТОВ В СПИСКЕ; ДЛИНА-33 БАЙТА; ГЛУБИНА СТЕКА-2 БАЙТА.
;*****
; ОРГАНИЗАЦИЯ ПОИСКА ЭЛЕМЕНТА, ПОСЛЕ КОТОРОГО
; ОСУЩЕСТВЛЯЕТСЯ ВКЛЮЧЕНИЕ
22AB CD9022      CALL    ПСКСП
; ПРОВЕРКА РЕЗУЛЬТАТА ПОИСКА
22AE 7D         MOV     A,L
22AF A4         ANA     H
22B0 3C         INR     A
22B1 C2B722     JNZ     PER14
; ВОЗВРАТ ПРИ НЕУДАЧНОМ ПОИСКЕ
22B4 3EFF     MVI     A,OFFH
22B6 C9         RET
; МОДИФИКАЦИЯ ПОЛЯ СВЯЗИ В ЭЛЕМЕНТЕ СПИСКА,
; ПОСЛЕ КОТОРОГО ПРОИЗВОДИТСЯ ВКЛЮЧЕНИЕ
PER14: LDSR     D,E+1    ; (D,E)-АДРЕС ВКЛЮЧАЕМОГО ЭЛЕМЕНТА
22C0 4E         MOV     C,M
22C1 73         MOV     M,E
22C2 23         INX     H
22C3 46         MOV     B,M
22C4 72         MOV     M,D    ; (B,C)-ЗНАЧЕНИЕ ПОЛЯ СВЯЗИ
; МОДИФИКАЦИЯ ПОЛЯ СВЯЗИ ВО ВКЛЮЧАЕМОМ ЭЛЕМЕНТЕ
; (H,L)-АДРЕС ВКЛЮЧАЕМОГО ЭЛЕМЕНТА
22C5 EB         XCHG
22C6 71         MOV     M,C
22C7 23         INX     H
22C8 70         MOV     M,B
; ВОЗВРАТ ПРИ УДАЧНОМ ЗАВЕРШЕНИИ
22C9 AF         XRA     A
22CA C9         RET

```

Поиск элемента по заданному значению ключа выполняется с помощью подпрограммы ПСКСП. После возврата из этой подпрограммы проверяется содержимое регистровой пары (H, L). Если оно равно OFFH (элемента с заданным ключом в списке не существует), в выходной параметр записывается признак неудачного выполнения и осуществляется возврат из программы. Проверка производится путем логического умножения содержимого регистров (H) и (L) и последующего их инкрементирования. Очевидно, что результат будет нулевой только в том случае, если в регистрах (H) и (L) содержался код OFFH. Если элемент найден, в его поле связи записывается адрес вставляемого элемента, причем старое содержимое поля связи сохраняется в регистровой паре (B, C).

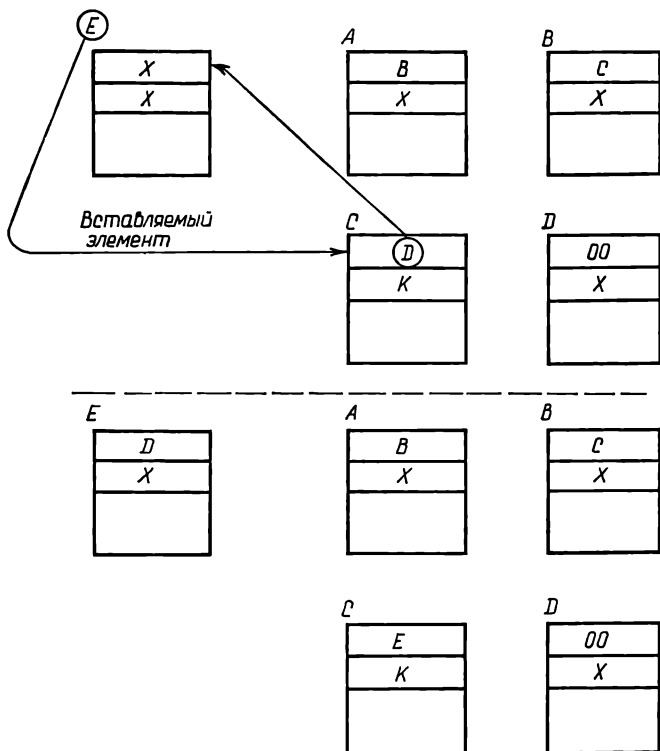


Рис. 5.9. Схема вставки элемента в список:

*A*, *B*, *C*, *D* — адреса элементов списка; *E* — адрес вставляемого элемента; *K* — значение ключа

В последнем фрагменте программы это сохраненное значение записывается в поле связи вставляемого элемента.

### 5.5.7. УДАЛЕНИЕ ЭЛЕМЕНТА ИЗ СПИСКА

Удаление элемента из списка заключается в том, что в поле связи элемента, стоящего перед удаляемым, записывается адрес элемента, стоящего в списке после удаляемого. Идентификацию удаляемого элемента списка можно произвести точно так же, как и ранее. Программа удаления имеет вид:

```
УДСП:
;*****
; ПОДПРОГРАММА УДАЛЕНИЯ ЭЛЕМЕНТА ИЗ СПИСКА.
; ВХОДНЫЕ ПАРАМЕТРЫ: (В,С) — АДРЕС НАЧАЛА СПИСКА, (Е) — КОД
```

```

; КЛЮЧА УДАЛЯЕМОГО ЭЛЕМЕНТА (УЗ).
; ВЫХОДНЫЕ ПАРАМЕТРЫ: (А)=0 - ТРЕБУЕМЫЙ ЭЛЕМЕНТ УДАЛЕН,
; (А)=OFFH - ТРЕБУЕМЫЙ ЭЛЕМЕНТ НЕ НАЙДЕН.
; ОЦЕНКА: ВРЕМЯ-143+93*N ТАКТОВ, ГДЕ N-КОЛИЧЕСТВО ЭЛЕМЕН-
; ТОВ В СПИСКЕ; ДЛИНА - 33 БАЙТА; ГЛУБИНА СТЕКА - 4 БАЙТА.
; *****
; ПОИСК УДАЛЯЕМОГО ЭЛЕМЕНТА
22CB C5      PUSH    B
22CC CD9022  CALL    ПСКСП
; ПРОВЕРКА РЕЗУЛЬТАТА ПОИСКА
22CF 7D      MOV     A,L
22D0 A4      ANA     H
22D1 3C      INR     A
22D2 C2D922  JNZ     ПЕР15 ; ЕСЛИ НАЙДЕН
; ВОЗВРАТ ПРИ НЕУДАЧНОМ ПОИСКЕ
22D3 C1      POP     B
22D6 3EFF    MVI     A,OFFH
22D8 C9      RET
; МОДИФИКАЦИЯ ЗНАЧЕНИЯ ПОЛЯ СВЯЗИ ЭЛЕМЕНТА.
22D9 79      ПЕР15: MOV    A,C
22DA B0      ORA     B
22DB C2E422  JNZ     ПЕР16 ; ЕСЛИ УЗ НЕ ПЕРВЫЙ В СПИСКЕ
22DE C1      POP     B
; МОДИФИКАЦИЯ АДРЕСА НАЧАЛА СПИСКА
22DF 4E      MOV     C,M
22E0 23      INX     H
22E1 46      MOV     B,M
22E2 AF      XRA     A
22E3 C9      RET
; МОДИФИКАЦИЯ ПОЛЯ СВЯЗИ ЭЛЕМЕНТА, СТОЯЩЕГО ПЕРЕД УЗ
22E4 7E      ПЕР16: MOV    A,M
22E5 02      STAX    B
22E6 23      INX     H
22E7 03      INX     B
22E8 7E      MOV     A,M
22E9 02      STAX    B
22EA AF      XRA     A
22EB C1      POP     B
22EC C9      RET

```

В первом фрагменте программы выполняется поиск удаляемого элемента. Структура этого фрагмента аналогична структуре первого фрагмента программы ВСТСП, рассмотренной на с. 241). Если элемент, подлежащий удалению, найден в списке, то содержимое его поля связи переписывается в поле связи предыдущего элемента. Адрес предыдущего элемента определяется подпрограммой ПСКСП и помещается ею в регистровую пару (В, С).

При удалении элемента из списка может возникнуть особая ситуация. Она заключается в том, что в качестве удаляемого элемента может быть задан первый элемент списка. При этом подпрограмма ПСКСП возвращает в регистровую пару (В, С) нулевое значение. В рассматри-

ваемой программе обработка этой ситуации сводится к тому, что изменяется адрес начала списка, который является входным параметром. Это выполняется путем записи содержимого поля связи удаляемого элемента в регистровую пару (В, С). Таким образом, изменение значения входного параметра, расположенного в регистровой паре (В, С), информирует вызывающую программу, что было произведено удаление первого элемента и дальнейшее обращение к списку должно производиться по новому адресу.

## 6. ПРОГРАММЫ СИСТЕМНОГО ОБЕСПЕЧЕНИЯ

### 6.1. ОБЩИЕ СВЕДЕНИЯ

В программном обеспечении МП систем можно выделить программы двух типов: *прикладные*, выполняющие числовую и символьную обработку структур данных на некоторой абстрактной (виртуальной) машине, и *системные*, превращающие реальные технические средства МП системы в абстрактную машину для работы прикладных программ. Эта классификация условна и зависит от уровня рассмотрения иерархии абстрактных машин. *Абстрактная машина* нижнего уровня, оснащенная базовыми прикладными программами и соответствующими системными средствами для их использования, может рассматриваться как машина следующего, более высокого уровня абстракции. При этом указанные прикладные программы теряют свое прикладное назначение и интерпретируются на более высоком уровне как системные. С точки зрения «голой» МП системы программы, приведенные ранее в книге, относятся к классу прикладных. В данной главе рассматриваются системные программы.

Совокупность системных программ образует *операционную систему*. Операционные системы микроконтроллеров и микроЭВМ подобны операционным системам больших и мини-ЭВМ, но значительно проще и компактнее. Функции, реализуемые системными программами, рассматриваемыми ниже, можно разделить на две группы. Первая группа обеспечивает выполнение в МП системе операции ввода-вывода, освобождая прикладные программы от выполнения специфических требований, предъявляемых к обмену информацией с различными *внешними устройствами* (ВУ). Эта группа функций реализуется специальными программами обмена — *драйверами*. Каждому ВУ соответствует свой драйвер, и прикладные программы взаимодействуют с ВУ посредством обращения к этим драйверам. Если в процессе эксплуатации МП системы возникает необходимость замены некоторого типа ВУ устройством другого типа, алгоритм обмена с ко-

торым отличается от данного, изменения в программном обеспечении сводятся, как правило, только к замене соответствующего драйвера. Вторая группа функций предоставляет пользователю возможность контролировать выполнение своих программ. Программы, реализующие эти функции, обеспечивают связь с пользователем посредством простого языка директив, с помощью которого можно запускать и останавливать программы, а также отображать и модифицировать содержимое памяти и регистров МП системы.

В данной главе рассматривается простейшая операционная система, которая называется *монитором*. Структурно монитор состоит из обработчика директив и программ управления вводом-выводом. Для иллюстрации построения драйверов выбраны типовые устройства. В их число входят *видеотерминалы* (дисплей) и *печатающие устройства*, имеющие выход на интерфейсы — *радиальный параллельный* (ИРПР) или *радиальный последовательный* (ИРПС), *накопители на гибких магнитных дисках* (НГМД) типа «Электроника ГМД-70/7012» и СМ616, а также *телетайп*. Некоторые драйверы ввода-вывода и реализации интерфейсов описаны в литературе [3, 22, 57, 62, 68, 69, 73]. Информацию об операционных системах микроЭВМ можно найти в книгах [26, 35, 38].

## 6.2. ДРАЙВЕР ОБМЕНА ПО ИРПР

ИРПР предназначен для радиального подключения ввода-вывода с параллельной передачей информации. По этому интерфейсу обмен данными производится побайтно, в асинхронном режиме, с использованием минимального количества сигналов состояния и управления. Передача данных осуществляется между одним источником и одним приемником, для организации *дуплексного обмена* (одновременного обмена в противоположных направлениях) требуются два сопряжения. Большинство серийно выпускаемых дисплеев и печатающих устройств имеет выход на этот интерфейс (например, дисплеи ВТА 2000-30, «Электроника 15ИЭ», печатающие устройства DZM-180 и DARO-1156). ИРПР ограничивает по стандарту расстояние между источником и приемником величиной 15 м.

Физически интерфейс образуют восемь линий ДАННЫЕ и четыре линии управления (рис. 6.1):

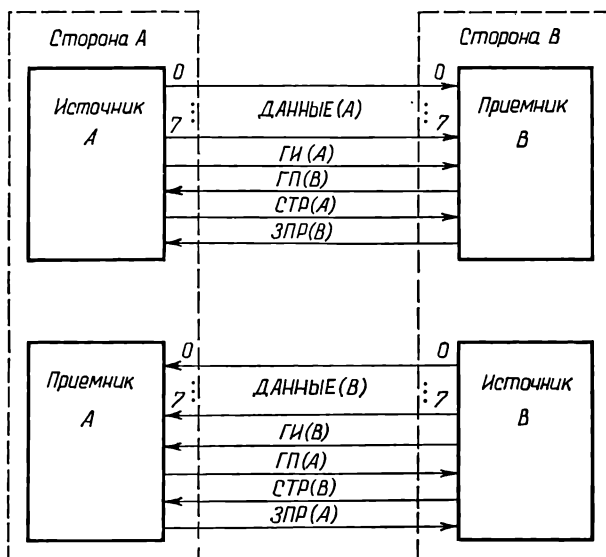


Рис. 6.1. Структура ИРПП

1) готовность источника (ГИ) — активное состояние (например, единичное) линии указывает, что источник физически существует и работоспособен (подключен к источнику питания);

2) готовность приемника (ГП) — линия предназначена для указания работоспособности приемника;

3) строб источника (СТР) — линия устанавливается источником в активное состояние в том случае, когда на линиях ДАННЫЕ появляется очередной байт;

4) запрос приемника (ЗПР) — активное состояние линии указывает на готовность приемника к приему очередного байта.

Обмен информацией регламентируется следующим алгоритмом. Работа по интерфейсу может осуществляться только при наличии сигналов ГИ и ГП, с помощью которых источник и приемник оповещают друг друга о работоспособности. Как только у источника появляется необходимость передать данные, он анализирует состояние линии ЗПР и в случае ее активности выдает байт на линии ДАННЫЕ. После этого источник устанавливает сигнал СТР, который записывает данные в приемник. Сигнал СТР снимается после перехода сигнала ЗПР приемника



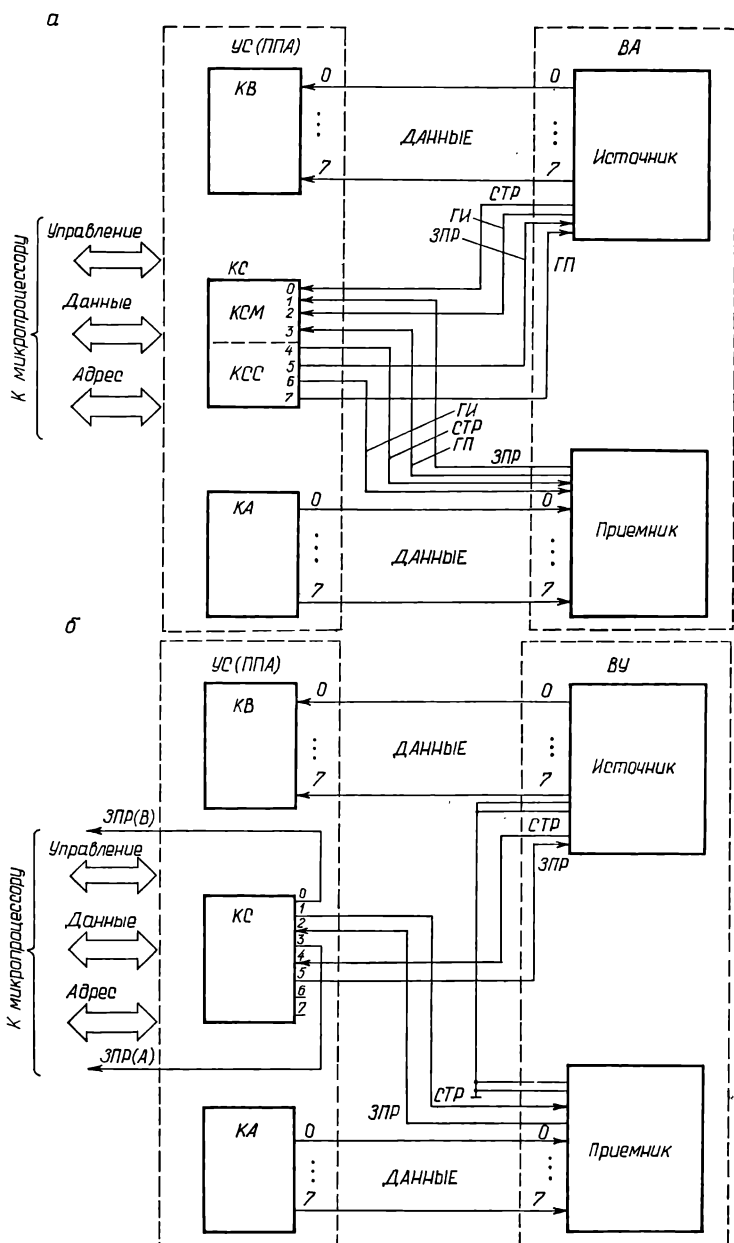


Рис. 6.2. Структуры сопряжения микросхемы ППА с источником и приемником данных

в пассивное (например, нулевое) состояние. В свою очередь приемник для получения байта должен установить сигнал ЗПР, дождаться активного состояния линии СТР и принять байт с линии ДАННЫЕ. На время обработки приемником полученного байта сигнал ЗПР снимается. Скорость обмена может задаваться как приемником, так и источником.

Любое ВУ подключается к МП системе с помощью специального *устройства сопряжения* (УС), которое преобразует физические сигналы интерфейса в логические, доступные программам. В МП системах, выполненных на базе БИС серии КР580, в устройствах сопряжения типа ИРПР чаще всего применяется микросхема периферийного параллельного адаптера (ППА) КР580ВВ55 (см. прил. 4) [2, 4, 12, 19, 31, 41]. На рис. 6.2 приведены структурные схемы устройства сопряжения типа ИРПР на базе ППА, работающего в режиме 0 (а) и в режиме 1 (б).

Рассмотрим работу драйвера ИРПР при использовании ППА в режиме 0. Драйвер обеспечивает выполнение функций инициализации, ввода и вывода символа:

```

; *****
; ПОДПРОГРАММЫ ДРАЙВЕРА ИНТЕРФЕЙСА И Р П Р ПРИ ИСПОЛЬ-
; ЗОВАНИИ БИС КР580ВВ55 В РЕЖИМЕ 0.
; *****

0000      КА      EQU      0      ; ПОРТ КАНАЛА А
0001      КВ      EQU      1      ; ПОРТ КАНАЛА В
0002      КС      EQU      2      ; ПОРТ КАНАЛА С
0003      РУС      EQU      3      ; ПОРТ УПРАВЛЯЮЩЕГО СЛОВА
ИРПРО:
; *****
; ПОДПРОГРАММА ИНИЦИАЛИЗАЦИИ.
; ВЫХОДНОЙ ПАРАМЕТР: (Z)=1, ЕСЛИ ИНИЦИАЛИЗАЦИЯ УСПЕШНАЯ,
; (Z)=0, В ПРОТИВНОМ СЛУЧАЕ.
; *****
; УСТАНОВКА РЕЖИМА

F400 3E83          MVI      A,10000011B
F402 D303          OUT      РУС
; УСТАНОВКА СИГНАЛА ГП

F404 3E0F          MVI      A,00001111B
F406 D303          OUT      РУС
; УСТАНОВКА СИГНАЛА ГИ

F408 3E0D          MVI      A,00001101B
F40A D303          OUT      РУС
; КОНТРОЛЬ ГИ, ГП ТЕРМИНАЛА

F40C DB02          IN       КС
F40E 2F            CMA
F40F E60C          ANI      00001100B
F411 C9            RET

```

# ВЫПРО:

```

; *****
; ПОДПРОГРАММА ВЫВОДА СИМВОЛА.
; ПАРАМЕТР: (С)-КОД ВЫВОДИМОГО СИМВОЛА.
; *****
; ОЖИДАНИЕ СИГНАЛА ЗПР ОТ ТЕРМИНАЛА

```

```

F412 DB02      IN      KC
F414 E602      ANI      00000010B
F416 CA12F4    JZ       ВЫПРО

```

; ВЫДАЧА ДАННЫХ

```

F419 79        MOV      А,С
F41A D300      OUT      КА

```

; ВЫДАЧА СИГНАЛА СТР

```

F41C 3E09      MVI      А,00001001B
F41E D303      OUT      РУС

```

; ОЖИДАНИЕ СНЯТИЯ СИГНАЛА ЗПР ОТ ТЕРМИНАЛА

```

F420 DB02      ЦИКЛ1: IN      KC
F422 E602      ANI      00000010B
F424 C220F4    JNZ      ЦИКЛ1

```

; СНЯТИЕ СИГНАЛА СТР

```

F427 3E08      MVI      А,00001000B
F429 D303      OUT      РУС
F42B C9        RET

```

## СТРПРО:

```

; *****
; ПОДПРОГРАММА ОПРЕДЕЛЕНИЯ СТАТУСА ВВОДА.
; ВЫХОДНОЙ ПАРАМЕТР: (А)=OFFH, ЕСЛИ ИНТЕРФЕЙС ГОТОВ ДЛЯ
; ВВОДА, (А)=0 - В ПРОТИВНОМ СЛУЧАЕ.
; *****
; ВЫДАЧА СИГНАЛА ЗПР

```

```

F42C 3E0B      MVI      А,00001011B
F42E D303      OUT      РУС

```

; ПРОВЕРКА СИГНАЛА СТР ОТ ТЕРМИНАЛА

```

F430 DB02      IN      KC
F432 E601      ANI      00000001B
F434 C8        RZ       ; ЕСЛИ НЕТ СТР

```

; УСТАНОВКА ПОЛОЖИТЕЛЬНОГО СТАТУСА

```

F435 3EFF      MVI      А,OFFH
F437 B7        ORA      А
F438 C9        RET

```

## ВВПРО:

```

; *****
; ПОДПРОГРАММА ВВОДА СИМВОЛА.
; ВЫХОДНОЙ ПАРАМЕТР: (А)- КОД ВВЕДЕННОГО СИМВОЛА.
; *****
; ОЖИДАНИЕ ПОЛОЖИТЕЛЬНОГО СТАТУСА ВВОДА

```

```

F439 CD2CF4    CALL     СТРПРО
F43C CA39F4    JZ       ВВПРО

```

; ВВОД ДАННЫХ

```

F43F DB01      IN      KB
F441 F5        PUSH     PSW

```

; СНЯТИЕ СИГНАЛА ЗПР

```

F442 3E0A      MVI      А,00001010B
F444 D303      OUT      РУС

```

; ОЖИДАНИЕ СНЯТИЯ СТР ОТ ТЕРМИНАЛА

```

F446 DB02      ЦИКЛ2: IN      KC
F448 E601      ANI      00000001B
F44A C246F4    JNZ      ЦИКЛ2
F44D F1        POP      PSW
F44E C9        RET

```

Драйвер состоит из четырех подпрограмм. Программа ИНРПРО производит инициализацию микросхемы и установку сигналов ГИ и ГП со стороны УС путем занесения информации в *регистр управляющего слова* (РУС). При инициализации устанавливается режим 0 для всех каналов, канал В (КВ) и младшая половина канала С (КСМ) программируются на ввод, а канал А (КА) и старшая половина канала С (КСС) — на вывод. Установка сигналов ГП и ГИ выполняется командами битовой установки. В конце программы ИНРПРО проверяется наличие сигналов ГП и ГИ со стороны ВУ. При их отсутствии в выходном параметре устанавливается признак неудачной инициализации  $Z=0$ . Программа ВЫРПРО осуществляет вывод символа из МП системы через УС в приемник ВУ (выводимый символ предварительно размещается в регистре (С)). В начале программы в цикле ожидания контролируется появление единицы во втором разряде КС. Эта единица интерпретируется как символ ЗПР от ВУ. Далее выводится символ и устанавливается сигнал СТР, после чего в цикле ожидания контролируется снятие сигнала ЗПР со стороны ВУ и, наконец, снимается сигнал СТР со стороны УС.

Программа СТРПРО выполняет проверку готовности источника ВУ к передаче символа в МП через УС. Говорят, что если ВУ готово к передаче, то оно имеет *положительный статус ввода*, а если не готово, — *отрицательный*. В программе СТРПРО статус ввода фиксируется в выходном параметре, передаваемом через аккумулятор МП. В начале программы устанавливается сигнал ЗПР со стороны УС и проверяется наличие ответного сигнала СТР со стороны ВУ. Если этот сигнал установлен, то в выходной параметр записывается положительный статус ввода (код 0FFH), в противном случае — отрицательный (код 00). Программа ВВРПРО осуществляет ввод символа в МП через УС от источника ВУ. В начале программы вызывается в цикле подпрограмма СТРПРО до тех пор, пока она не передаст в выходном параметре положительный статус ввода. Затем из КВ вводится символ и снимается сигнал ЗПР со стороны УС. Возврат из этой программы выполняется только после снятия сигнала СТР со стороны ВУ.

Драйвер ИРПР при использовании ППА в режиме 1 имеет аналогичную структуру:

```

; *****
; ПОДПРОГРАММЫ ДРАЙВЕРА ИНТЕРФЕЙСА И Р П Р ПРИ ИСПОЛЬ-
; ЗОВАНИИ БИС КР580ВВ55 В РЕЖИМЕ 1.
; *****
ИНРПР1:
; *****
; ПОДПРОГРАММА ИНИЦИАЛИЗАЦИИ ИНТЕРФЕЙСА
; *****
; УСТАНОВКА РЕЖИМА
F44F 3EA6      MVI      A,10100110B
F451 D303      OUT      PUC
; СНЯТИЕ МАСКИ ИСТОЧНИКА
F453 3E0C      MVI      A,00001100B
F455 D303      OUT      PUC
; СНЯТИЕ МАСКИ ПРИЕМНИКА
F457 3E04      MVI      A,00000100B
F459 D303      OUT      PUC
F45B AF        XRA      A          ; СБРОС ФЛАГА Z
F45C C9        RET

ВЫРПР1:
; *****
; ПОДПРОГРАММА ВЫВОДА СИМВОЛА.
; ПАРАМЕТР: (C) - КОД ВЫВОДИМОГО СИМВОЛА.
; *****
; ОЖИДАНИЕ ГОТОВНОСТИ ИСТОЧНИКА
F45D DB02      IN        KC
F45F E608      ANI      00001000B
F461 CA5DF4     JZ        ВЫРПР1
; ВЫВОД ДАННЫХ
F464 79        MOV      A,C
F465 D300      OUT      KA
F467 C9        RET

СТРПР1:
; *****
; ПОДПРОГРАММА ОПРЕДЕЛЕНИЯ СТАТУСА ВВОДА.
; ВЫХОДНОЙ ПАРАМЕТР: (A)=OFFH, ЕСЛИ ИНТЕРФЕЙС ГОТОВ ДЛЯ
; ВВОДА, И (A)=0 В ПРОТИВНОМ СЛУЧАЕ.
; *****
; ПРОВЕРКА СИГНАЛА ГП
F468 DB02      IN        KC
F46A E601      ANI      00000001B
F46C C8        RZ          ; ЕСЛИ НЕТ ГОТОВНОСТИ
; УСТАНОВКА ПОЛОЖИТЕЛЬНОГО СТАТУСА
F46D 3EFF      MVI      A,OFFH
F46F B7        ORA      A
F470 C9        RET

ВВРПР1:
; *****
; ПОДПРОГРАММА ВВОДА СИМВОЛА.
; ВЫХОДНОЙ ПАРАМЕТР: (A) - КОД ВВЕДЕННОГО СИМВОЛА.
; *****
F471 CD68F4     CALL     СТРПР1
F474 CA71F4     JZ        ВВРПР1
; ВВОД ДАННЫХ
F477 DB01      IN        KB
F479 C9        RET

```

При инициализации, которую выполняет подпрограмма ИНРПР1, микросхема программируется в режим 1. КА настраивается на режим вывода, а КВ — ввода. Кроме того, при выполнении программы размаскируются сигналы ЗПР по КВ и КА путем установки единиц в 6-м и 2-м разрядах КС. При выводе символа (подпрограмма ВЫРПР1) циклически проверяется состояние 3-го разряда КС. Как только в этом разряде устанавливается единица, что свидетельствует о готовности ППА вывести символ, символ выводится в КА. Готовность ППА к вводу определяется подпрограммой СТРПР1, которая возвращает в качестве выходного параметра значение статуса ввода, исходя из текущего состояния 0-го разряда. КС — сигнала ЗПР (В). Подпрограмма ВВРПР1 выполняет ввод символа. Перед чтением символа из КВ выполняется в цикле вызов подпрограммы СТРПР1 до тех пор, пока не будет получен положительный статус ввода.

При использовании ППА в режиме 1 драйвер управления вводом-выводом получается более компактным, так как все манипуляции сигналами управления выполняет сама микросхема. Заметим, что драйвер режима 0 можно использовать для УС, выполненных без применения ППА. Этот драйвер требует наличия только трех 8-разрядных программно-доступных регистров.

### **6.3. ДРАЙВЕР ОБМЕНА ПО ИРПС**

ИРПС предназначен для асинхронной передачи информации в виде импульсов постоянного тока по четырехпроводной дуплексной линии связи. Обмен производится последовательным двоичным кодом с регулярной скоростью по двум цепям: «Передаваемые данные» и «Принимаемые данные». ИРПС использует меньше линий связи, чем ИРПР, и позволяет обмениваться данными с ВУ, удаленными на расстояние до 500 м, но при этом скорость обмена существенно уменьшается из-за последовательного характера передачи. Применять ИРПС рекомендуется для связи с удаленными ВУ, когда явно проявляется его преимущество в экономии линий связи. Устройство сопряжения типа ИРПС удобно выполнять на базе микросхемы *универсального синхронно-асинхронного приемопередатчика* (УСАПП) КР580ВВ51 (см. прил. 5) [2, 4, 14, 19, 36, 41].

Драйвер ИРПС, так же как и драйвер ИРПР, состоит из четырех подпрограмм, которые обеспечивают инициа-

лизацию УС, ввод, вывод символа и определение статуса ввода:

```

;*****
; ПОДПРОГРАММЫ ЛРАЙВЕРА ИНТЕРФЕЙСА И Р П С ПРИ ИСПОЛЬ-
; ЗОВАНИИ БИС КР580ИК51.
;*****
0010 ДАН51 EQU 10H ; ПОРТ РЕГИСТРА ДАННЫХ
0011 РУС51 EQU 11H ; ПОРТ УПРАВЛЯЮЩЕГО СЛОВА
ИНРПС:
;*****
; ПОДПРОГРАММА ИНИЦИАЛИЗАЦИИ ИНТЕРФЕЙСА.
;*****
; ЗАГРУЗКА В БИС ИНСТРУКЦИИ РЕЖИМА
F47A 3E4F MVI A,01001111B
F47C D311 OUT РУС51
; ЗАГРУЗКА В БИС ИНСТРУКЦИИ КОМАНДЫ
F47E 3E05 MVI A,00000101B
F480 D311 OUT РУС51
F482 AF XRA A ; СБРОС ФЛАГА Z
F483 C9 RET
ВЫРПС:
;*****
; ПОДПРОГРАММА ВЫВОДА СИМВОЛА.
; ПАРАМЕТР: (С) - КОД ВЫВОДИМОГО СИМВОЛА.
;*****
; ОЖИДАНИЕ ГОТОВНОСТИ ПЕРЕДАТЧИКА
F484 DD11 IN РУС51
F486 E601 ANI 00000001B
F488 CA84F4 JZ ВЫРПС
; ВЫВОД ДАННЫХ
F48B 79 MOV A,C
F48C D310 OUT ДАН51
F48E C9 RET
СТРПС:
;*****
; ПОДПРОГРАММА ОПРЕДЕЛЕНИЯ СТАТУСА ВВОДА.
; ВЫХОДНОЙ ПАРАМЕТР: (А) = OFFH, ЕСЛИ ИНТЕРФЕЙС ГОТОВ ДЛЯ
; ВВОДА И (А) = 0, В ПРОТИВНОМ СЛУЧАЕ.
;*****
; ПРОВЕРКА ГОТОВНОСТИ ПРИЕМНИКА
F48F DB11 IN РУС51
F491 E602 ANI 00000010B
F493 CB RZ ; ЕСЛИ НЕТ ГОТОВНОСТИ
; УСТАНОВКА ПОЛОЖИТЕЛЬНОГО СТАТУСА
F494 3EFF MVI A,OFFH
F496 B7 ORA A
F497 C9 RET
ВВРПС:
;*****
; ПОДПРОГРАММА ВВОДА СИМВОЛА.
; ВЫХОДНОЙ ПАРАМЕТР: (А) - КОД ВВЕДЕННОГО СИМВОЛА.
;*****
F498 CD8FF4 CALL СТРПС
F49B CA98F4 JZ ВВРПС
; ВВОД ДАННЫХ
F49E DB10 IN ДАН51
F4A0 C9 RET

```

Инициализацию выполняет программа ИНРПС, которая загружает в РУС *инструкцию режима* и *инструкцию команды*. Инструкция режима предписывает УСАПП работать в асинхронном режиме восьмибитными послылками с одним стоп-битом и с коэффициентом деления частоты синхронизации, равным 1:64. Инструкция команды разрешает прием и передачу данных. Программа ВЬРПС осуществляет вывод символа. В начале программы организован цикл ожидания сигнала готовности передатчика (ГПД). При установленном сигнале ГПД выводимый символ записывается в регистр данных УСАПП. Программа СТРПС определяет статус ввода УС в зависимости от состояния сигнала готовности приемника (ГПР). Значение статуса ввода возвращается в качестве выходного параметра. Ввод символа производит программа ВВРПС. Вводимый символ читается из регистра ввода УСАПП только при наличии положительного статуса ввода, который предварительно определяется подпрограммой СТРПС.

#### **6.4. ДРАЙВЕР ОБМЕНА С ТЕЛТАЙПОМ**

*Консоль*, или пульт оператора, в современных МП системах реализуется, как правило, на базе алфавитно-цифрового или графического дисплея. Тем не менее применение в качестве консоли телетайпа имеет ряд достоинств. Во-первых, в телетайпе наряду с клавиатурой и печатающим устройством имеются устройства ввода и вывода на пятидорожечную перфоленту (трансмиссер и перфоратор), что делает телетайп более универсальным, чем дисплей; во-вторых, телетайпы в настоящее время широко распространены и более доступны, чем дисплеи.

При проектировании драйвера телетайпа возникает проблема, связанная с тем, что телетайп воспринимает символы, представленные в системе кодирования МТК-2 (международный телеграфный код), а для внутренних обменов в МП системе, как правило, используется система кодирования КОИ-7 (код обмена информацией). Для дисплеев этой проблемы нет, так как большинство дисплеев «понимает» КОИ-7. Решить эту проблему с помощью простого перекодирования, программа которого приведена в гл. 5, не удастся, так как системы кодирования МТК-2 и КОИ-7 различаются не только значениями кодов одинаковых символов, но и структурой. В системе КОИ-7



а					б				
+	1	2	3	4	+	1	2	3	4
0	Е	ПС*	А	—*	0	Е	ПС*	А	—*
4	Ѕ	І	U	ВК*	4	С	И	У	ВК*
8	Д	Р	Ј	Н	8	Д	Р	Й	Н
12	Ѓ	С	К	Т	12	Ф	Ц	К	Т
16	Ѕ	Л	W	Н	16	З	Л	В	Х
20	У	Р	Q	О	20	Ы	П	Я	О
24	В	Г	ЦИФ*	М	24	Б	Г	ЦИФ*	М
28	Х	У	ЛАТ*	РУС*	28	Ь	Ж	ЛАТ*	РУС*

в									
+	1	2	3	4	5	6	7	8	
0	3	ПС*	—	—*	'	8	7	ВК*	
8	КТМ	4	Ю*	,	Э*	:	(	5	
16	+	)	2	Щ*	6	0	1	9	
24	?	Ш*	ЦИФ*	.	/	=	ЛАТ*	РУС*	

Рис. 6.3. Таблицы перекодировки:  
а — ЛАТ; б — РУС; в — ЦИФ (символ КТМ означает «Кто там?»)

каждый символ кодируется семибитным кодом, который однозначно определяет этот символ. В системе же МТК-2 символы кодируются пятибитными кодами, причем один код может соответствовать сразу трем символам, а конкретный символ определяется по текущему значению специального признака — *управляющего регистра*. Выделяют три управляющих регистра: *цифровой* (ЦИФ), *латинский* (ЛАТ) и *русский* (РУС). Каждый регистр в системе МТК-2 имеет свой уникальный десятичный код: (ЦИФ) = 27, (ЛАТ) = 31, (РУС) = 32. Кроме регистров, уникальные коды имеют *управляющие символы* «Перевод строки» (ПС) = 02, «Возврат каретки» (ВК) = 08 и «Пробел» (—) = 04.

Для перекодировки символов из системы КОИ-7 в систему МТК-2 и обратно необходимо иметь три таблицы (рис. 6.3). Каждая таблица содержит 32 элемента (клет-

ки) с обозначениями в них символов соответствующего регистра. Значение любого символа в таблицах, выраженное в системе МТК-2, равно сумме соответствующих координатных индексов, расположенных в верхней строке и крайнем левом столбце каждой таблицы. Например, значение латинского символа Q равно  $20 + 3 = 23$ . На языке ассемблера таблицы перекодировки имеют вид:

#### ЦТАБ:

\*\*\*\*\*  
; ТАБЛИЦА КОДОВ КОИ-7 В СООТВЕТСТВИИ С ТАБЛИЦЕЙ ЦИФРОВО-  
; ГО РЕГИСТРА КОДА МТК-2.  
\*\*\*\*\*

F655 00330A2D	DB 000, 'З', 'ОАН', '-', ' ', '27H', '8', '7'
F659 20273837	
F65D 0D053460	DB 0DH, 'ОН', '4', 'Ю', ' ', ' ', '3', ' ', ' '
F661 2C7C3A28	
F665 352B2932	DB '5', '+', ' )', ' ', '2', ' ', 'Ш', ' ', '6', ' ', '0', ' ', '1'
F669 7D363031	
F66D 393F7B1B	DB '9', '?', 'Ш', ' ', '027', ' ', ' ', ' ', ' ', ' ', '031
F671 2E2F3D1F	

#### ЛТАБ:

\*\*\*\*\*  
; ТАБЛИЦА КОДОВ КОИ-7 В СООТВЕТСТВИИ С ТАБЛИЦЕЙ ЛАТИН-  
; СКОГО РЕГИСТРА КОДА МТК-2.  
\*\*\*\*\*

F675 00450A41	DB 000, 'Е', 'ОАН', 'А', ' ', ' ', 'S', ' ', 'I', ' ', 'U'
F679 20534955	
F67D 0D44524A	DB 0DH, 'D', ' ', 'R', ' ', 'J', ' ', 'N', ' ', 'F', ' ', 'C', ' ', 'K'
F681 4E46434B	
F685 545A4C57	DB 'T', 'Z', 'L', ' ', 'W', ' ', 'H', ' ', 'Y', ' ', 'G', ' ', 'Q'
F689 4B595051	
F68D 4F42471B	DB 'O', 'B', ' ', 'G', ' ', '027', ' ', 'M', ' ', 'X', ' ', 'U', ' ', '031
F691 4D5B561F	

#### РТАБ:

\*\*\*\*\*  
; ТАБЛИЦА КОДОВ КОИ-7 В СООТВЕТСТВИИ С ТАБЛИЦЕЙ РУССКО-  
; ГО РЕГИСТРА КОДА МТК-2.  
\*\*\*\*\*

F695 00650A61	DB 000, 'Е', 'ОАН', 'А', ' ', ' ', 'C', ' ', 'И', ' ', 'У'
F699 20736975	
F69D 0D64726A	DB 0DH, 'Д', ' ', 'P', ' ', 'И', ' ', 'H', ' ', 'Ф', ' ', 'Ц', ' ', 'K'
F6A1 6E66636B	
F6A5 747A6C77	DB 'T', 'Z', ' ', 'L', ' ', 'B', ' ', 'X', ' ', 'Ы', ' ', 'П', ' ', 'Я'
F6A9 6B797071	
F6AD 6F62671B	DB 'O', 'B', ' ', 'G', ' ', '027', ' ', 'H', ' ', 'Ь', ' ', 'Ж', ' ', '031
F6B1 6D7B761F	

В этих таблицах значение символов представлено в системе КОИ-7, а значение символов в системе МТК-2 определяется порядковым номером соответствующего символа внутри таблицы.

Для управления обменом с телетайпом (как вводом, так и выводом) необходимо запоминать коды последнего



Рис. 6.4. Структура драйвера телетайпа

принимаемого и передаваемого регистра. Драйвер обмена с телетайпом состоит из пяти подпрограмм и имеет структуру, показанную на рис. 6.4. Как вводимые, так и выводимые символы подлежат перекодировке, которая выполняется соответствующими программами. Перекодировку символов из системы МТК-2 в систему КОИ-7 выполняет программа МТККОИ:

#### МТККОИ:

```

; *****
; ПОДПРОГРАММА ПЕРЕКОДИРОВКИ СИМВОЛА ИЗ КОДА МТК-2 В
; КОД КОИ-7.
; ВХОДНЫЕ ПАРАМЕТРЫ: (С)-КОД МТК-2, (D,E)-АДРЕС ТЕКУЩЕГО
; РЕГИСТРА КОДА МТК-2.
; ВЫХОДНЫЕ ПАРАМЕТРЫ: (A)- КОД КОИ-7, (A) = OFFH, ЕСЛИ
; НЕСУЩЕСТВУЮЩИЙ КОД МТК-2, (A) = 0, ЕСЛИ ПЕРЕКОДИРУЕМЫЙ
; СИМВОЛ-КОД РЕГИСТРА МТК-2.
; *****
; ПРОВЕРКА КОДА ТЕКУЩЕГО РЕГИСТРА

```

```

F61F 1A          LDAX  D      ; ТЕКУЩИЙ РЕГИСТР МТК-2
F620 2155F6      LXI   H,CTAB
F623 FE1B        CPI    27    ; КОД ЦИФРОВОГО РЕГИСТРА
F625 CA3BF6      JZ     PER16
F628 2175F6      LXI   H,LTAB
F62B FE1F        CPI    31    ; КОД ЛАТИНСКОГО РЕГИСТРА
F62D CA3BF6      JZ     PER16
F630 2195F6      LXI   H,RTAB
F633 FE00        CPI    00    ; КОД РУССКОГО РЕГИСТРА
F635 CA3BF6      JZ     PER16

```

```

; ВОЗВРАТ ПРИ ОШИБКЕ КОДА РЕГИСТРА ИЛИ ПРИ ОБНАРУЖЕНИИ
; СИМВОЛА, НЕ ПРИНАДЛЕЖАЩЕГО КОДУ МТК-2

```

```

F638 3EFF        PER15: MVI   A,OFFH
F63A C9          RET

```

```

; ПРОВЕРКА ПРИНАДЛЕЖНОСТИ КОДА СИМВОЛА К МТК-2

```

```

F63B 79          PER16: MOV   A,C
F63C FE20        CPI    32
F63E D23BF6      JNC    PER15 ; ЕСЛИ НЕ ПРИНАДЛЕЖИТ
; ВЫБОР АНАЛОГА ИЗ ТАБЛИЦЫ

```

```

F641 0600        MVI    B,0
F643 09          DAD    B
F644 7E          MOV    A,M

```

	; ПРОВЕРКА НА КОД РЕГИСТРА		
F645 FE1B	CPI	27	
F647 CA52F6	JZ	ПЕР17	; ЕСЛИ КОД "ЦИФ"
F64A FE1F	CPI	31	
F64C CA52F6	JZ	ПЕР17	; ЕСЛИ КОД "ЛАТ"
F64F FE00	CPI	00	
F651 C0	RNZ		; ЕСЛИ НЕ КОД "РУС"
	; ВОЗВРАТ, ЕСЛИ ПЕРЕКОДИРУЕМЫЙ СИМВОЛ ЯВЛЯЕТСЯ		
	; КОДОМ РЕГИСТРА МТК-2		
F652 12	ПЕР17:	STAX	D
F653 97		SUB	A
F654 C9		RET	

Вторым входным параметром программы является адрес ячейки памяти, в которой хранится код последнего принятого регистра. В начале программы по значению этого кода устанавливается в регистровой паре (H, L) адрес соответствующей таблицы перекодировки (если задается некорректное исходное значение кода регистра или кода перекодируемого символа, то выполняется возврат из программы с установкой выходного параметра (A) = 0FFH). После этого значение кода перекодируемого символа добавляется командой DAD B к адресу начала таблицы в регистровой паре (H, L). Полученная сумма является адресом ячейки памяти, где хранится код КОИ-7 перекодируемого символа. Если выбранное из таблицы значение кода является кодом управляющего регистра, оно записывается по адресу текущего регистра (второй входной параметр), и выполняется возврат из программы со значением выходного параметра (A) = 00. В противном случае в выходной параметр записывается код КОИ-7 перекодируемого символа.

При перекодировке из системы КОИ-7 в систему МТК-2 для выбора таблицы перекодировки используется следующее правило: коды символов КОИ-7, значения которых лежат в диапазоне 41H — 5AH, принадлежат таблице латинского регистра МТК-2, коды в диапазоне 61H — 4AH — русского, коды в диапазоне 20H — 39H — цифрового. Исключение составляют управляющие символы ПС, ВК и пробела, которые присутствуют во всех таблицах, а также особые символы «Ю», «Э», «Ш», «Щ». На рис. 6.3 элементы, содержащие эти символы, отмечены звездочкой. Перекодировку из системы КОИ-7 в систему МТК-2 выполняет программа КОИМТК:

```

КОИМТК:
; *****
; ПОДПРОГРАММА ПЕРЕКОДИРОВКИ СИМВОЛОВ ИЗ КОДА КОИ-7 В
; КОД МТК-2.

```

; ВХОДНЫЕ ПАРАМЕТРЫ: (С)-КОД КОИ-7, (D,E)-АДРЕС УСТАНОВ-  
 ; ЛЯЕМОГО РЕГИСТРА КОДА МТК-2.  
 ; ВЫХОДНЫЕ ПАРАМЕТРЫ: (А)- КОД МТК-2, (А)=OFFH, ЕСЛИ НЕ-  
 ; СУЩЕСТВУЮЩИЙ КОД.

\*\*\*\*\*

```

F5A8 79          MOV     A,C      ; ПЕРЕКОДИРУЕМЫЙ СИМВОЛ
; СЕЛЕКЦИЯ БЕЗРЕГИСТРОВЫХ СИМВОЛОВ
F5A9 0E02        MVI     C,02
F5AB FE0A        CPI     0AH
F5AD CADD5       JZ      PER8     ; ЕСЛИ (ПС)
F5B0 0E04        MVI     C,04
F5B2 FE20        CPI     ' '
F5B4 CADD5       JZ      PER8     ; ЕСЛИ ПРОБЕЛ
F5B7 0E08        MVI     C,08
F5B9 FE0D        CPI     0DH
F5BB CADD5       JZ      PER8     ; ЕСЛИ (BK)
; СЕЛЕКЦИЯ "ОСОБЫХ" СИМВОЛОВ
F5BE 0E0B        MVI     C,11
F5C0 FE60        CPI     'Ю'
F5C2 CADA5       JZ      PER7     ; ЕСЛИ "Ю"
F5C5 0E0D        MVI     C,13
F5C7 FE7C        CPI     'Э'
F5C9 CADA5       JZ      PER7     ; ЕСЛИ "Э"
F5CC 0E14        MVI     C,20
F5CE FE7D        CPI     'Щ'
F5D0 CADA5       JZ      PER7     ; ЕСЛИ "Щ"
F5D3 0E1A        MVI     C,26
F5D5 FE7B        CPI     'Ш'
F5D7 C2DFF5      JNZ      PER9     ; ЕСЛИ НЕ "Ш"
; ВОЗВРАТ ПРИ ВЫЯВЛЕНИИ БЕЗРЕГИСТРОВЫХ
; ИЛИ "ОСОБЫХ" СИМВОЛОВ
F5DA 3E1B        MVI     A,27     ; КОД РЕГИСТРА "ЦИФР"
F5DC 12          STAX     D        ; УСТАНОВКА РЕГИСТРА
F5DD 79          PER8: MOV     A,C      ; УСТАНОВКА СИМВОЛА
F5DE C9          RET
; ПРОВЕРКА ПРИНАДЛЕЖНОСТИ К ТАБЛИЦАМ ПЕРЕКОДИРОВКИ
F5DF 0600        PER9: MVI     B,0
F5E1 4F          MOV     C,A      ; ПЕРЕКОДИРУЕМЫЙ СИМВОЛ
F5E2 FE41        CPI     41H
F5E4 D2EFF5      JNC     PER10    ; ЕСЛИ НЕ НА "ЦИФ"
F5E7 3E1B        MVI     A,27     ; КОД РЕГИСТРА "ЦИФ"
F5E9 2155F6      LXI     H,CTAB   ; ТАБЛИЦА РЕГИСТРА "ЦИФ"
F5EC C30CF6      JMP     PER13
F5EF FE61        PER10: CPI     61H
F5F1 D2FCF5      JNC     PER11    ; ЕСЛИ НЕ НА "ЛАТ"
F5F4 3E1F        MVI     A,31     ; КОД РЕГИСТРА "ЛАТ"
F5F6 2175F6      LXI     H,LTAB   ; ТАБЛИЦА РЕГИСТРА "ЛАТ"
F5F9 C30CF6      JMP     PER13
F5FC FE7B        PER11: CPI     7BH
F5FE D209F6      JNC     PER12    ; ЕСЛИ НЕ НА "РУС"
F601 3E00        MVI     A,00     ; КОД РЕГИСТРА "РУС"
F603 2195F6      LXI     H,RTAB   ; ТАБЛИЦА РЕГИСТРА "РУС"
F606 C30CF6      JMP     PER13
; ВОЗВРАТ ПРИ ОТСУТСТВИИ АНАЛОГА В КОДЕ МТК-2
F609 3EFF        PER12: MVI     A,OFFH
F60B C9          RET
; ВЫБОР АНАЛОГА ИЗ СООТВЕТСТВУЮЩЕЙ ТАБЛИЦЫ
F60C 12          PER13: STAX     D        ; УСТАНОВКА РЕГИСТРА
  
```

F60D 79		MOV	A,C	; ПЕРЕКОДИРУЕМЫЙ СИМВОЛ
F60E 0E20		MVI	C,32	; КОЛИЧЕСТВО СИМВОЛОВ В ТАБЛИЦЕ
F610 BE	ЦИКЛ8:	CMR	M	
F611 C216F6		JNZ	ПЕР14	
F614 7B		MOV	A,B	; АНАЛОГ В КОДЕ МТК-2

F615 C9		RET		
				; ПРОВЕРКА НА ЗАВЕРШЕНИЕ ПРОСМОТРА ТАБЛИЦЫ
F616 04	ПЕР14:	INR	B	
F617 23		INX	H	
F618 0D		DCR	C	
F619 C210F6		JNZ	ЦИКЛ8	; ЕСЛИ ПРОСМОТР НЕ ЗАВЕРШЕН
F61C C309F6		JMP	ПЕР12	; АНАЛОГ В ТАБЛИЦАХ НЕ НАЙДЕН

Программа, помимо перекодировки, записывает по адресу, содержащемуся во втором входном параметре, код соответствующего регистра МТК-2. В начале программы осуществляется селекция управляющих и особых символов, о которых речь шла выше. Селекция производится путем прямого сравнения. Если обнаруживается управляющий (безрегистровый) символ, в выходной параметр записывается код этого символа в системе МТК-2 и осуществляется возврат из программы. Если же обнаруживается особый символ «Ю», «Э», «Ш» или «Щ», то перед возвратом по адресу регистра (второй входной параметр) записывается код цифрового регистра. Затем, исходя из принадлежности кода символа к соответствующему диапазону, определяются адрес таблицы перекодировки и код регистра. Если код перекодированного символа не принадлежит ни одному из диапазонов значений, то в выходной параметр записывается признак несуществующего кода. В противном случае код регистра записывается по адресу, определяемому вторым входным параметром, а в соответствующей таблице путем последовательного сравнения производится поиск символа. При обнаружении элемента, содержащего код перекодированного символа, индекс элемента, являющийся кодом этого символа в системе МТК-2, записывается в выходной параметр и выполняется возврат из программы.

Устройство сопряжения и алгоритм обмена с телеграфом по своей структуре аналогичны УС типа ИРПС (различаются только уровни напряжения в приемной и передающей цепях). Поэтому для ввода-вывода телеграфных посылок можно использовать подпрограммы драйвера ИРПС (ВВРПС и ВЫРПС):

## ИНТЛП:

```

; *****
; ПОДПРОГРАММА ИНИЦИАЛИЗАЦИИ ИНТЕРФЕЙСА ИРПС ДЛЯ РАБОТЫ
; С ТЕЛЕТАЙПОМ.

```

```

; *****

```

```

; ВЫДАЧА ИНСТРУКЦИИ РЕЖИМА

```

```

F559 3E83      MVI      A,10000011B
F55B D311      OUT      RUSC1

```

```

; ВЫДАЧА ИНСТРУКЦИИ КОМАНДЫ

```

```

F55D 3E05      MVI      A,00000101B
F55F D311      OUT      RUSC1

```

```

; НАЧАЛЬНАЯ УСТАНОВКА КОДОВ РЕГИСТРОВ

```

```

F561 97        SUB      A
F562 32A5F5     STA      TRBY
F565 32A6F5     STA      PRBY
F568 32A7F5     STA      RBV
F56B C9        RET

```

## ВВТЛП:

```

; *****

```

```

; ПОДПРОГРАММА ВВОДА СИМВОЛА.

```

```

; ВЫХОДНОЙ ПАРАМЕТР: (А) - КОД КОИ-7 ВВЕДЕННОГО СИМВОЛА.

```

```

; *****

```

```

; СОХРАНЕНИЕ РЕГИСТРОВ

```

```

F56C E5        PUSH     H
F56D D5        PUSH     D
F56E C5        PUSH     B

```

```

; ВВОД СИМВОЛА

```

```

F56F CD98F4    CALL     BVRPC
F572 E61F      ANI      1FH

```

```

; ПЕРЕКОДИРОВКА СИМВОЛА

```

```

F574 4F        MOV      C,A
F575 11A7F5     LXI      D,RBV ; АДРЕС КОДА РЕГИСТРА
F578 CD1FF6     CALL     KTKKOI
F57B B7        ORA      A
F57C CA6FF5     JZ       ЦИКЛ7 ; ЕСЛИ ВВЕДЕН КОД РЕГИСТРА

```

```

; ВОССТАНОВЛЕНИЕ РЕГИСТРОВ

```

```

F57F C1        POP      B
F580 D1        POP      D
F581 E1        POP      H
F582 C9        RET

```

## ВЫТЛП:

```

; *****

```

```

; ПОДПРОГРАММА ВЫВОДА СИМВОЛА.

```

```

; ПАРАМЕТР: (С) - КОД КОИ-7 ВЫВОДИМОГО СИМВОЛА.

```

```

; *****

```

```

; СОХРАНЕНИЕ РЕГИСТРОВ

```

```

F583 E5        PUSH     H
F584 D5        PUSH     D
F585 C5        PUSH     B

```

```

; ПЕРЕКОДИРОВКА СИМВОЛА

```

```

F586 E67F      ANI      7FH
F588 11A5F5     LXI      D,TRBY ; АДРЕС КОДА ТЕКУЩЕГО РЕГИСТРА
F58B CDA8F5     CALL     KOIMTK

```

```

; СРАВНЕНИЕ КОДОВ ТЕКУЩЕГО И ПРЕДЫДУЩЕГО РЕГИСТРОВ

```

```

F58E 4F        MOV      C,A
F58F 1A        LDAX     D
F590 21A6F5     LXI      H,PRBY ; АДРЕС КОДА ПРЕДЫДУЩЕГО РЕГИСТРА

```

F593 BE	CMP	X	
F594 CA9EF5	JZ	PER6	; ЕСЛИ КОД РЕГИСТРА НЕ ИЗМЕНИЛСЯ
			; ВЫДАЧА КОДА РЕГИСТРА
F597 77	MOV	M, A	
F598 C5	PUSH	B	
F599 4F	MOV	C, A	
F59A CD84F4	CALL	ВЫПТС	
F59D C1	POP	B	
			; ВЫДАЧА СИМВОЛА
F59E CD84F4	PER6:	CALL	ВЫПТС
			; ВОССТАНОВЛЕНИЕ РЕГИСТРОВ
F5A1 C1	POP	B	
F5A2 D1	POP	D	
F5A3 E1	POP	H	
F5A4 C9	RET		
			; РАБОЧИЕ ЯЧЕЙКИ
F5A5	TPBV:	DS	1 ; ТЕКУЩИЙ КОД РЕГИСТРА ДЛЯ ВЫВОДА
F5A6	IPBV:	DS	1 ; ПРЕДЫДУЩИЙ КОД РЕГИСТРА ДЛЯ ВЫВОДА
F5A7	PBV:	DS	1 ; КОД РЕГИСТРА ДЛЯ ВВОДА

Инициализация выполняется программой ИНТЛП, которая программирует УСАПП и обнуляет содержимое рабочих ячеек памяти, предназначенных для хранения кодов регистров. В этой программе инструкция режима отличается от инструкции режима в подпрограмме ИНРПС тем, что длина посылки устанавливается равной 5 битам, а длина стоп-бита — 1,5 бита, что соответствует правилам кодирования в системе МТК-2.

Процедура ввода символа с телетайпа, которую реализует программа ВВТЛП, заключается в следующем. Ввод телеграфной посылки осуществляется с помощью подпрограммы ВВРПС. После этого следует обращение к подпрограмме МТККОИ, которая перекодирует введенный символ в систему КОИ-7 и записывает код соответствующего регистра в рабочую ячейку РВВ. Если введенная посылка является кодом регистра, возврат из программы не производится и процедура ввода повторяется; в противном случае выполняется возврат из программы.

Вывод символа на телетайп выполняет подпрограмма ВЫТЛП. Символ, представленный в системе КОИ-7, с помощью подпрограммы КОИМТК перекодируется в систему МТК-2. Если в результате перекодировки образуется код текущего регистра, отличный от кода предыдущего регистра, соответствующего предыдущему выведенному символу, необходимо до вывода текущего символа выполнить вывод нового кода текущего ре-



гистра. С этой целью в рабочей ячейке ПРВЫ сохраняется код предыдущего регистра, а в ячейку ТРВЫ записывается код регистра текущего символа. Если после возврата из подпрограммы КОИМТК содержимое указанных ячеек различается, ячейка ТРВЫ копируется в ячейку ПРВЫ, а затем ее содержимое с помощью подпрограммы ВЫРПС выводится на телетайп. Аналогичным образом вслед выводится сам символ. Если же содержимое ячеек ПРВЫ и ТРВЫ одинаково, вывод кода регистра не производится. Программы ВЫТЛП и ВВТЛП обеспечивают сохранение всех регистров МП, кроме аккумулятора.

## 6.5. ДРАЙВЕР ОБМЕНА С ДИСКОВОДОМ

Широко распространенным устройством внешней памяти, применяемым в МП системах, является НГМД. Носителем информации в нем служит полимерный диск стандартного диаметра (например, 203 мм, или 8 дюймов), покрытый с обеих сторон ферромагнитным материалом и заключенный в плотный конверт с отверстиями, через которые производятся чтение и запись информации (рис. 6.5). Поверхность диска разбита на 77 концентрических окружностей — дорожек. Каждая дорожка содержит 26 секторов. Один сектор может хранить 128 байт полезной информации, а диск в целом — 256 К байт. Рассматриваемый ниже драйвер ориентирован на управление обменом с накопителем типа «Электроника ГМД-7012», который имеет два устройства для установки дисков и взаимодействует с МП системой посредством *интерфейсных команд*. Эти команды позволяют выполнять обмен данными между внутренним буфером НГМД и заданным сектором на диске, а также между МП системой и внутренним буфером НГМД.

Основные линии физического взаимодействия НГМД с МП системой показаны на рис. 6.6. Данные и команды передаются последовательным кодом по линии ДАННЫЕ. Направление обмена по этой линии определяется состоянием линии ВЫВОД. Линия СДВИГ служит для стробирования битов символа, передаваемых по линии ДАННЫЕ. По линии ПУСК передаются сигналы, информирующие накопитель, что УС готово принять или передать очередной символ. Аналогичное назначение имеет линия ЗАПРОС ПЕРЕДАЧИ (только сигналы передаются со стороны накопителя). По линии ЗАВЕРШЕНО

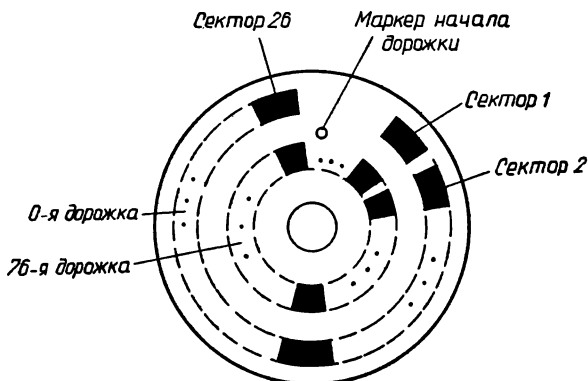


Рис. 6.5. Структура размещения информации на ГМД

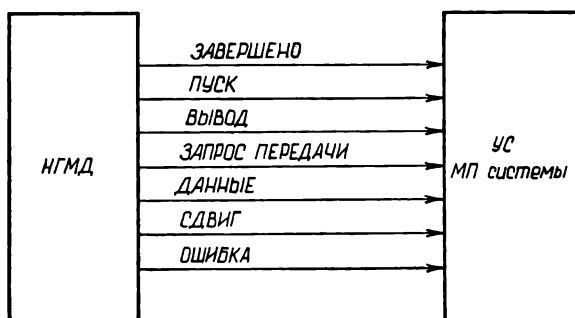


Рис. 6.6. Внешний интерфейс НГМД

накопитель информирует устройство сопряжения об окончании выполнения команды. По линии ОШИБКА фиксируется факт удачного или неудачного выполнения команды.

Взаимодействие драйвера с УС осуществляется через порты ввода-вывода данных. Порт вывода (ПВЫ) предназначен для передачи байта информации в накопитель, а порт ввода (ПВВ) — для его приема. Порт вывода ПУСК служит для генерации сигнала «Пуск». Порт ввода СОСТ, формат которого приведен на рис. 6.7, служит для фиксации сигналов, поступающих из накопителя. Драйвер обеспечивает выполнение двух функций: запись информации из памяти МП системы в указанный сектор

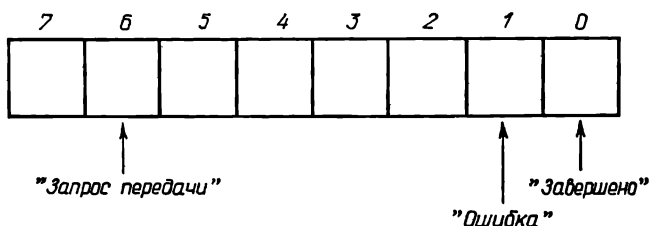


Рис. 6.7. Формат регистра состояния НГМД

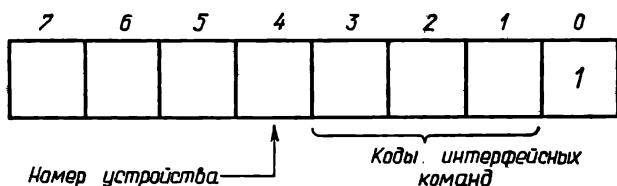


Рис. 6.8. Форматы команд НГМД:

000 — «Запись в буфер»; 001 — «Чтение буфера»; 010 — «Запись сектора»;  
011 — «Чтение сектора»

на диске и чтение информации из сектора на диске в память МП системы.

Для выполнения записи на диск необходимо предварительно записать информацию из памяти во внутренний буфер НГМД. Это делается с помощью интерфейсной команды «Запись буфера», которая состоит из 129 байт: первый байт содержит код команды (рис. 6.8), а остальные 128 байт — данные, предназначенные для записи в сектор. Каждый байт команды загружается в порт ПВЫ, после чего генерируется сигнал «Пуск». Очередной байт можно загружать только после появления сигнала «Запрос передачи». После вывода 129-го байта вместо сигнала «Запрос передачи» накопитель передает сигнал «Завершено». После записи информации в буфер НГМД необходимо выполнить перезапись из буфера на диск командой «Запись сектора». Для реализации этой команды следует загрузить в накопитель 3 байта. Первый байт содержит код команды, второй и третий — соответственно номер сектора (1—26) и номер дорожки (0—76).

Чтение сектора диска в память МП системы осуществляется также в два этапа — с диска во внутренний буфер и из буфера НГМД в память МП системы. Чтение сектора диска во внутренний буфер НГМД выполняется

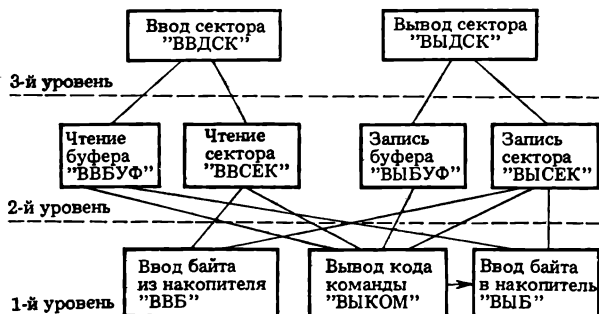


Рис. 6.9. Структура драйвера НГМД

командой «Чтение сектора», которая задается аналогично команде «Запись сектора» (но с другим кодом). Для перемещения информации из буфера накопителя в память МП системы служит команда «Чтение буфера», для выполнения которой необходимо в порт ПВЫ выдать код команды, а затем из порта ПВВ по каждому сигналу «Запрос передачи» прочесть 128 байт данных. После последнего байта вместо сигнала «Запрос передачи» передается сигнал «Завершено».

Структура драйвера НГМД приведена на рис. 6.9. Драйвер состоит из 9 программ. Внешние программы взаимодействуют с драйвером только через программы 3-го уровня. Побайтный обмен с накопителем выполнят программы 1-го уровня:

```

0004          ; *****
0004          ПВВ      EQU      4          ; ПОРТ ВВОДА ДАННЫХ
0005          ПВЫ      EQU      5          ; ПОРТ ВЫВОДА ДАННЫХ
0004          ПУСК     EQU      4          ; ПОРТ СИГНАЛА "ПУСК"
0005          СОСТ     EQU      5          ; ПОРТ РЕГИСТРА СОСТОЯНИЯ
ВЫБ:
; *****
; ПОДПРОГРАММА ВЫВОДА БАЙТА В НАКОПИТЕЛЬ.
; ВХОДНОЙ ПАРАМЕТР: (А) - ВЫВОДИМЫЙ БАЙТ.
; ВЫХОДНЫЙ ПАРАМЕТР: (СУ)=1-ПОЛУЧЕН СИГНАЛ "ЗАВЕРШЕНО",
; (СУ)=0 - ПОЛУЧЕН СИГНАЛ "ЗАПРОС ПЕРЕДАЧИ".
; *****
F4A1 D305          OUT      ПВЫ          ; ЗАГРУЗКА БАЙТА
F4A3 D304          OUT      ПУСК
; ОЖИДАНИЕ ОТВЕТНЫХ СИГНАЛОВ
F4A5 DB05          PER1:   IN      СОСТ
F4A7 0F            RRC
F4A8 DB            RC          ; ЕСЛИ "ЗАВЕРШЕНО"
F4A9 E620          ANI      20H

```

```

F4A8 C0          RNZ          ; ЕСЛИ "ЗАПРОС ПЕРЕДАЧИ"
F4AC C3A5F4      JMP          ПЕР1

ВВБ:
; *****
; ПОДПРОГРАММА ПРИЕМА БАЙТА ИЗ НАКОПИТЕЛЯ.
; ВЫХОДНЫЕ ПАРАМЕТРЫ: (А) - ПРИНЯТЫЙ БАЙТ, (СУ) = 1, ЕСЛИ
; ПОЛУЧЕН СИГНАЛ "ЗАВЕРШЕНО", (СУ) = 0, ЕСЛИ - "ЗАПРОС
; ПЕРЕДАЧИ".
; *****
; ВВОД БАЙТА ИЗ НАКОПИТЕЛЯ
F4AF DB04        IN          ПВВ
F4B1 D304        OUT         ПУСК
F4B3 5F          MOV         Е, А
; ОЖИДАНИЕ СИГНАЛОВ "ЗАВЕРШЕНО" ИЛИ "ЗАПРОС ПЕРЕДАЧИ"
F4B4 CDA5F4      CALL        ПЕР1
F4B7 7B          MOV         А, Е
F4B8 C9          RET

ВЫКОМ:
; *****
; ПОДПРОГРАММА ВЫДАЧИ КОМАНДЫ В НАКОПИТЕЛЬ.
; ПАРАМЕТРЫ: (С) - КОД КОМАНДЫ, НУСТ - НОМЕР ДИСКОВОГО
; УСТРОЙСТВА.
; *****
; ФОРМИРОВАНИЕ БАЙТА С КОДОМ КОМАНДЫ
F4B9 3A54F5      LDA         НУСТ
F4BC 07          RLC
F4BD 07          RLC
F4BE 07          RLC
F4BF 07          RLC
F4C0 B1          ORA         С
; ВЫДАЧА В НАКОПИТЕЛЬ
F4C1 C3A1F4      JMP         ВЫБ

```

Программа ВЫБ управляет выводом байта в накопитель. Байт загружается в порт ПВЫ, после чего генерируется сигнал «Пуск» путем выполнения команды OUT ПУСК. Возврат из программы выполняется только при наличии сигналов «Завершено» или «Запрос передачи». Программа ВВБ производит ввод байта из накопителя. Сначала выдается сигнал «Пуск», а затем происходит ожидание сигналов «Завершено» или «Запрос передачи». Для организации этого ожидания с помощью команды CALL вызывается соответствующий фрагмент подпрограммы ВЫБ. Затем из порта ПВВ вводится байт. Программа вывода команды (ВЫКОМ) обеспечивает установку в заданной команде бита номера устройства в соответствии с содержимым ячейки НУСТ.

Выполнение основных функций накопителя обеспечивают подпрограммы 2-го уровня драйвера:

# ВВЕУФ:

```

; *****
; ПОДПРОГРАММА ЧТЕНИЯ БУФЕРА НАКОПИТЕЛЯ В ПАМЯТЬ.
; ВХОДНОЙ ПАРАМЕТР: АБУФ-АДРЕС БУФЕРА В ПАМЯТИ.
; ВЫХОДНОЙ ПАРАМЕТР: (А) - СТАТУС ЗАВЕРШЕНИЯ.
; *****
; ВЫДАЧА БАЙТА КОМАНДЫ

```

```

F4C4 0E02      MVI    C,2      ; (C)-КОД КОМАНДЫ ЧТЕНИЯ БУФЕРА
F4C6 CDB9F4    CALL   ВЫКОМ
F4C9 DAEDF4    JC     ПЕР2      ; ЕСЛИ "ЗАВЕРШЕНО"
; ПРИЕМ 128 БАЙТ ИЗ БУФЕРА НАКОПИТЕЛЯ В ПАМЯТЬ
F4CC 2A57F5    LHLD   АБУФ
F4CF 067F      MVI     В,127
F4D1 CDAFF4    ЦИКЛ3: CALL  ВВБ
F4D4 DAEDF4    JC     ПЕР2      ; ЕСЛИ "ЗАВЕРШЕНО"
F4D7 77        MOV     М,А
F4D8 23        INK     Н
F4D9 05        DCR     В
F4DA C2D1F4    JNZ    ЦИКЛ3
F4DD CDAFF4    CALL   ВВБ
F4E0 D2EDF4    JNC     ПЕР2      ; ЕСЛИ НЕТ "ЗАВЕРШЕНО"
F4E3 77        MOV     М,А
; ПРОВЕРКА СИГНАЛА "ОШИБКА"
F4E4 DB05      PROВ: IN     СОСТ
F4E6 E602      ANI     2
F4E8 C2EDF4    JNZ    ПЕР2      ; ЕСЛИ "ОШИБКА"
; ВОЗВРАТ ПРИ УСПЕШНОМ ЗАВЕРШЕНИИ
F4EB AF        XRA     А
F4EC C9        RET
; ВОЗВРАТ ПРИ АВАРИЙНОМ ЗАВЕРШЕНИИ
F4ED 3EFF      ПЕР2: MVI   А,OFFH
F4EF B7        ORA     А
F4F0 C9        RET

```

# ВВЕУФ:

```

; *****
; ПОДПРОГРАММА ЗАПИСИ ИНФОРМАЦИИ В БУФЕР НАКОПИТЕЛЯ.
; ВХОДНОЙ ПАРАМЕТР: АБУФ-АДРЕС БУФЕРА В ПАМЯТИ.
; ВЫХОДНОЙ ПАРАМЕТР: (А)-СТАТУС ЗАВЕРШЕНИЯ.
; *****
; ВЫДАЧА БАЙТА КОМАНДЫ

```

```

F4F1 0E00      MVI     C,0      ; (C)-КОД КОМАНДЫ ЗАПИСИ БУФЕРА
F4F3 CDB9F4    CALL   ВЫКОМ
F4F6 DAEDF4    JC     ПЕР2      ; ЕСЛИ "ЗАВЕРШЕНО"
; ПЕРЕДАЧА 128 БАЙТ ИЗ ПАМЯТИ В БУФЕР НАКОПИТЕЛЯ
F4F9 2A57F5    LHLD   АБУФ
F4FC 067F      MVI     В,127
F4FE 7E        ЦИКЛ4: MOV    А,М
F4FF CDA1F4    CALL   ВЫБ
F502 DAEDF4    JC     ПЕР2      ; ЕСЛИ "ЗАВЕРШЕНО"
F505 23        INX     Н
F506 05        DCR     В
F507 C2FEF4    JNZ    ЦИКЛ4
F50A 7E        MOV     А,М
F50B CDA1F4    CALL   ВЫБ
F50E D2EDF4    JNC     ПЕР2      ; ЕСЛИ НЕТ "ЗАВЕРШЕНО"
F511 C3E4F4    JMP     ПРОВ

```

```

ВВСЕК:
; *****
; ПОДПРОГРАММА ЧТЕНИЯ СЕКТОРА С ДИСКА В БУФЕР НАКОПИТЕЛЯ.
; ВХОДНЫЕ ПАРАМЕТРЫ: НСЕК - НОМЕР СЕКТОРА, НАОР - НОМЕР
; ДОРОЖКИ.
; ВЫХОДНОЙ ПАРАМЕТР: (А) - СТАТУС ЗАВЕРШЕНИЯ.
; *****
; ВЫДАЧА КОДА КОМАНДЫ
F514 0E06             MVI     C,6       ; (C)-КОД КОМАНДЫ ЧТЕНИЯ СЕКТОРА
F516 CDB9F4          PER3:  CALL    ВЫКОМ
F519 DAEDF4           JC      PER2      ; ЕСЛИ "ЗАВЕРШЕНО".
; ВЫДАЧА НОМЕРА СЕКТОРА
F51C 3A56F5           LDA      НСЕК
F51F CDA1F4           CALL     ВЫБ
F522 DAEDF4           JC      PER2      ; ЕСЛИ "ЗАВЕРШЕНО"
; ВЫДАЧА НОМЕРА ДОРОЖКИ
F525 3A55F5           LDA      НАОР
F528 CDA1F4           CALL     ВЫБ
F52B D2EDF4           JNC      PER2      ; ЕСЛИ НЕТ "ЗАВЕРШЕНО"
F52E C3E4F4           JMP      ПРОВ

ВВСЕК:
; *****
; ПОДПРОГРАММА ЗАПИСИ СЕКТОРА ДИСКА ИЗ БУФЕРА НАКОПИТЕЛЯ.
; ВХОДНЫЕ ПАРАМЕТРЫ: НСЕК - НОМЕР СЕКТОРА, НАОР - НОМЕР
; ДОРОЖКИ.
; ВЫХОДНОЙ ПАРАМЕТР: (А) - СТАТУС ЗАВЕРШЕНИЯ.
; *****
; ВЫДАЧА КОДА КОМАНДЫ
F531 0E04             MVI     C,4       ; (C)-КОД КОМАНДЫ ЗАПИСИ СЕКТОРА
F533 C316F5           JMP      PER3

```

Программа ВВБУФ реализует чтение 128 байт из буфера НГМД в область памяти, адрес которой записан в ячейке АБУФ. Вначале подпрограммой ВЫКОМ выводится байт с кодом команды «Чтение буфера», а затем следует цикл приема 127 байт информации. На каждой итерации цикла посредством обращения к подпрограмме ВВБ вводится из НГМД один байт и записывается в память. Для адресации очередного байта используется указатель в регистровой паре (H, L). Контроль окончания цикла производится с помощью счетчика, который располагается в регистре (B) и декрементируется на каждой итерации. После каждого вызова программы ВВБ проверяется отсутствие сигнала «Завершено». Последний, 128-й, байт вводится вне цикла, так как после него проверяется наличие сигнала «Завершено». Отрицательный статус завершения операции устанавливается в трех случаях: 1) сигнал «Завершено» появляется в процессе выполнения команды; 2) сигнал «Завершено» отсутствует после выполнения команды; 3) появляется сигнал «Ошибка».

Структура программы ВЫБУФ, выполняющей запись

128 байт информации из памяти МП системы в буфер НГМД, аналогична структуре программы ВВБУФ. Отличие лишь в том, что в первом фрагменте выводится код команды «Запись буфера» и для вывода информации используется программа ВЫБ. Программа ВВСЕК обеспечивает чтение информации из сектора на диске во внутренний буфер НГМД: в накопитель последовательно выводятся код команды «Чтение сектора» и содержимое ячеек НСЕК и НДОР, в которых записаны номера соответственно сектора и дорожки. Отрицательный статус завершения операции устанавливается при наличии сигнала «Завершено» во время выполнения команды или его отсутствии в момент завершения команды. Запись информации из буфера НГМД в сектор на диске выполняет программа ВЫСЕК. Она отличается от программы ВВСЕК только одной командой, которая определяет код команды накопителя. Поэтому после загрузки кода команды в регистр (С) управление передается соответствующему фрагменту программы ВВСЕК.

Программы 3-го уровня драйвера обеспечивают с помощью программ 2-го уровня передачу информации из памяти МП системы в заданный сектор на диске и обратно. Внешние программы, которым необходимо осуществить операцию обмена с накопителем, должны записать в ячейки памяти НУСТ, НДОР и НСЕК соответственно номера нужных устоя, дорожки и сектора и вызвать необходимую подпрограмму 3-го уровня:

#### ВВАСК:

```

; *****
; ПОДПРОГРАММА ЧТЕНИЯ СЕКТОРА С ДИСКА В ПАМЯТЬ.
; ВХОДНЫЕ ПАРАМЕТРЫ: НСЕК - НОМЕР СЕКТОРА, НДОР - НОМЕР
; ДОРОЖКИ, АБУФ - АДРЕС БУФЕРА, НУСТ - НОМЕР ДИСКОВОГО
; УСТРОЙСТВА.
; ВЫХОДНОЙ ПАРАМЕТР: (А) - СТАТУС ЗАВЕРШЕНИЯ.
; *****
F536 1620      MVI     D,32      ; УСТАНОВКА СЧЕТЧИКА ПОВТОРЕНИЙ
; ЧТЕНИЕ СЕКТОРА В БУФЕР НАКОПИТЕЛЯ
F538 CD14F5    CALL    ВВСЕК
; ЧТЕНИЕ БУФЕРА НАКОПИТЕЛЯ В ПАМЯТЬ
F53B CCC4F4    CZ      ВВБУФ
F53E C8        RZ          ; ЕСЛИ УСПЕШНО
; ПОВТОРНАЯ ПОПЫТКА
F53F 15        DCR      D
F540 C238F5    JNZ     ЦИКЛ5    ; ЕСЛИ НЕ ВСЕ ПОПЫТКИ ВЫПОЛНЕНЫ
; АВАРИЙНЫЙ ВОЗВРАТ
F543 B7        ORA      А
F544 C9        RET

```



#### ВЫДСК:

```
*****
; ПОДПРОГРАММА ЗАПИСИ СЕКТОРА ИЗ ПАМЯТИ НА ДИСК.
; ВХОДНЫЕ ПАРАМЕТРЫ: НСЕК - НОМЕР СЕКТОРА, НДОР - НОМЕР
; ДОРОЖКИ, АБУФ - АДРЕС БУФЕРА, НУСТ - НОМЕР ДИСКОВОГО
; УСТРОЙСТВА.
; ВЫХОДНОЙ ПАРАМЕТР: (А) - СТАТУС ЗАВЕРШЕНИЯ.
*****
F545 1620          MVI    D,32      ; УСТАНОВКА СЧЕТЧИКА ПОВТОРЕНИЙ
; ЗАПИСЬ В БУФЕР НАКОПИТЕЛЯ ИЗ ПАМЯТИ
F547 CDF1F4        ЦИКЛ6: CALL  ВЫБУФ
; ЗАПИСЬ СЕКТОРА ИЗ БУФЕРА НАКОПИТЕЛЯ
F54A CC31F5          CZ      ВЫСЕК
F54D C8             RZ              ; ЕСЛИ УСПЕШНО
; ПОВТОРНАЯ ПОПЫТКА
F54E 15             DCR      D
F54F C247F5          JNZ     ЦИКЛ6   ; ЕСЛИ НЕ ВСЕ ПОПЫТКИ ВЫПОЛНЕНЫ
; АВАРИЙНЫЙ ВОЗВРАТ
F552 B7             ORA      А
F553 C9             RET
*****
; РАБОЧИЕ ЯЧЕЙКИ ПАМЯТИ.
*****
F554      НУСТ: DS      1          ; НОМЕР УСТРОЙСТВА
F555      НДОР: DS      1          ; НОМЕР ДОРОЖКИ
F556      НСЕК: DS      1          ; НОМЕР СЕКТОРА
F557      АБУФ: DS      2          ; АДРЕС БУФЕРА
```

В программе ВВДСК, которая выполняет чтение содержимого сектора в память, сначала с помощью подпрограммы ВВСЕК информация из сектора считывается в буфер накопителя, а затем посредством подпрограммы ВВБУФ — в память МП системы. Если хотя бы одна из этих подпрограмм возвращает в вызывающую программу отрицательный статус своего завершения, описанная процедура повторяется до 32 раз. Если все попытки оказываются неудачными, программа ВВДСК устанавливает отрицательный статус своего завершения. Запись информации из памяти МП системы в сектор на диске реализует подпрограмма ВЫДСК. В этой программе с помощью подпрограмм ВЫБУФ и ВЫСЕК информация записывается сначала в буфер накопителя, а затем на диск. Так же как и в программе ВВДСК, предусмотрен 32-кратный повтор в случае получения от вызываемых подпрограмм отрицательного статуса их завершения.

## 6.6. СИСТЕМНЫЙ МОНИТОР

### 6.6.1. ФУНКЦИИ И СТРУКТУРА

Системное программное обеспечение не только организует управление вводом-выводом, но также должно

предоставлять пользователю элементарные возможности для отладки и контроля выполнения своих программ. В связи с этим в состав монитора, кроме драйверов, включается обработчик директив (ОД) пользователя. Язык директив позволяет отображать и модифицировать содержимое ячеек памяти и регистров МП системы, а также организовывать выполнение программ с точками останова. Для работы ОД необходимо наличие в составе МП системы консоли. В рассматриваемом случае предполагается, что консоль подключена к ИРПР и работает в системе КОИ-7. Кроме того, имеется еще печатающее устройство, подключенное к МП системе по ИРПС. Перечень *директив монитора* приведен в табл. 6.1.

Табл. 6.1. Директивы монитора

Формат	Выполняемое действие	Назначение параметров
D <П1>, <П2>	Отображение содержимого области памяти на консоль	П1 — адрес начала отображаемой области; П2 — адрес ее конца
F <П1>, <П2>, <П3>	Заполнение области памяти	П1, П2 — адреса первого и последнего байтов заполняемой области; П3 — значение байта этой области
M <П1>, <П2>, <П3>	Перемещение области памяти	П1, П2 — адреса первого и последнего байтов области источника; П3 — адрес первого байта области приемника
S <П1>	Отображение и модификация содержимого области памяти	П1 — адрес первого байта
X <П1>	Отображение и модификация содержимого регистров МП	П1 — имя регистра: А, В, С, D, Е, F, Н, L, Р или S
G <П1>, <П2>, <П3>	Запуск программы пользователя с указанием точек разрыва	П1 — адрес старта программы; П2, П3 — адреса точек разрыва
H	Установка (сброс) режима копирования консольного вывода на устройство памяти	—

Монитор — неотъемлемая часть МП системы и размещается, как правило, в постоянной памяти. В связи с этим одним из основных требований, предъявляемых к программе монитора, является компактность, которая

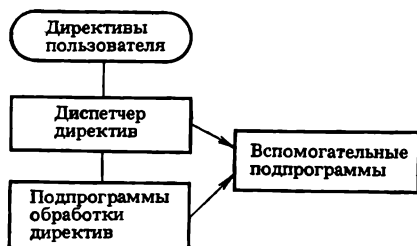


Рис. 6.10. Структура ОД монитора

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
EO					флаг печати	C	B	E	D	PSW	A	SP		INX SP	POP B	POP D
FO	POP PSW	POP H	SPHL	LXI H	L	H	EI	JMP	PC		Точка 1				Точка 2	

Рис. 6.11. Структура рабочей области монитора

достигается за счет широкого применения подпрограмм. Структура ОД представлена на рис. 6.10. Непосредственно с пользователем взаимодействует программа *диспетчера директив*, распознающая директиву по первому символу. Диспетчер вызывает соответствующую подпрограмму обработки директивы, которая вводит параметры директивы и выполняет предписанные действия, обращаясь к вспомогательным подпрограммам.

Монитор в процессе своей работы использует определенное пространство оперативной памяти — *рабочую область монитора*. В первых пяти ячейках оперативной памяти (начиная с нулевого адреса) размещаются трехбайтная команда перехода на подпрограмму обработки точек останова и двухбайтный адрес последней ячейки оперативной памяти. В старших 28 ячейках оперативной памяти располагаются флаг печати (МЛБ адреса этой ячейки равен 0E4H), область сохранения содержимого регистров программы пользователя (МЛБ адресов: 0E5H...0ECH), последовательность команд для запуска программы пользователя (МЛБ адресов: 0EDH...0F9H) и две трехбайтные структуры, в которых хранятся содержимое и адреса двух точек останова (рис. 6.11). Так как в разных МП системах объем оперативной памяти может быть различным, то для рабочих переменных фиксированными являются значения только МЛБ адресов

(указаны на рис. 6.11). СТБ адреса определяется по содержимому четвертой ячейки памяти (равен СТБ адреса верхней границы памяти).

## 6.6.2. ВСПОМОГАТЕЛЬНЫЕ ПРОГРАММЫ

Вспомогательные программы ОД разделим условно на три группы: 1) программы ввода-вывода символов; 2) программы преобразования шестнадцатеричных чисел из символьного представления в двоичное (и наоборот); 3) программы ввода-вывода шестнадцатеричных чисел. Программы 1-й группы обеспечивают диспетчеру и подпрограммам ОД возможность ввода-вывода различной символьной информации:

### ВЫВС:

```

; *****
; ПОДПРОГРАММА ВЫВОДА СИМВОЛА НА КОНСОЛЬ.
; ПАРАМЕТР: (C) - КОД ВЫВОДИМОГО СИМВОЛА.
; *****

```

FA69 CD84F4	CALL	ВЫРПС	; ВЫВОД НА КОНСОЛЬ
FA6C E5	PUSH	H	
FA6D 2A0300	LHLD	3	
FA70 2EE4	MVI	L, 0E4H	
FA72 7E	MOV	A, H	
FA73 B7	ORA	A	
FA74 E1	POP	H	
FA75 C25DF4	JNZ	ВЫРП	; ВЫВОД НА ПЕЧАТЬ
FA78 C9	RET		

### ВЫВС1:

```

; *****
; ПОДПРОГРАММА ВЫВОДА СИМВОЛА НА КОНСОЛЬ.
; ПАРАМЕТРЫ: (SP) - АДРЕС ЯЧЕЙКИ ПАМЯТИ, СОДЕРЖАЩЕЙ КОД
; КОИ-7 ВЫВОДИМОГО СИМВОЛА.
; ПРИМЕЧАНИЕ: ВОЗВРАТ ИЗ ЭТОЙ ПОДПРОГРАММЫ ПРОИСХОДИТ ПО
; АДРЕСУ (SP)+1.
; *****

```

FA79 E3	XTHL		; (H, L) - АДРЕС СИМВОЛА
FA7A 4E	MOV	C, H	
FA7B 23	INX	H	
FA7C E3	XTHL		; (SP) - АДРЕС ВОЗВРАТА
FA7D C369FA	JMP	ВЫВС	

### КВВЫ:

```

; *****
; ПОДПРОГРАММА ВВОДА СИМВОЛА С КОНСОЛИ С ЭХОМ.
; ВЫХОДНОЙ ПАРАМЕТР: (A) - КОД ВВЕДЕННОГО СИМВОЛА.
; *****

```

FA80 C5	PUSH	B	
FA81 CD71F4	CALL	ВВРП	; ВВОД СИМВОЛА
FA84 E67F	ANI	7FH	
FA86 4F	MOV	C, A	
FA87 CD69FA	CALL	ВЫВС	; ВЫВОД СИМВОЛА
FA8A 79	MOV	A, C	

```

FAB8 C1          POP      B
FAB8 C9          RET

      КОНС:
; *****
; ПОДПРОГРАММА УСТАНОВКИ НОВОЙ СТРОКИ НА КОНСОЛИ.
; *****

FA8D CD79FA      CALL     ВЫВС1  ; ВЫВОД ВК
FA90 0D          DB       ODH
FA91 CD79FA      CALL     ВЫВС1  ; ВЫВОД ПС
FA94 0A          DB       OAH
FA95 C9          RET

      ВЫВП:
; *****
; ПОДПРОГРАММА ВЫВОДА ПРОБЕЛА НА КОНСОЛЬ.
; *****

FA96 0E20        MVI      C, ' '
FA98 6369FA      JMP      ВЫВС

      ТЕКСТ:
; *****
; ПОДПРОГРАММА ВЫВОДА НА КОНСОЛЬ ЦЕПОЧКИ СИМВОЛОВ.
; ПАРАМЕТР: (H,L) - АДРЕС НАЧАЛА ЦЕПОЧКИ СИМВОЛОВ.
; *****

FA9B 7E          MOV      A,M
FA9C B7          ORA      A
FA9D C8          RZ              ; ЕСЛИ ОЧЕРЕДНОЙ КОД РАВЕН 0
FA9E 4F          MOV      C,A
FA9F CD69FA      CALL     ВЫВС   ; ВЫВОД СИМВОЛА
FAA2 23          INX      H
FAA3 C39BFA      JMP      ТЕКСТ

```

Программы ВЫВС и ВЫВС1 реализуют вывод на консоль одного символа. Программа ВЫВС после вывода символа на консоль драйвером ИРПР проверяет состояние флага печати. Если он установлен, вывод символа дублируется на печатающее устройство через драйвер ИРПС. Программа ВЫВС1 выводит на консоль символ, код которого расположен непосредственно за командой вызова данной программы, а возврат осуществляется по адресу, следующему за символом. Такой механизм передачи параметра позволяет вызывающей программе организовать вызов в четырех байтах вместо пяти при использовании программы ВЫВС. Ввод символа с консоли выполняется программой КВВЫ. Эта программа осуществляет также вывод введенного символа — «эхо» — на устройство отображения консоли (после нажатия клавиши). Программы КОНС и ВЫВП предназначены для вывода на консоль наиболее часто встречающихся символов. Первая выполняет переход на новую строку устройства отображения посредством выдачи ко-

дов ВК и ПС, вторая — кода пробела. Программа ТЕКСТ выводит последовательность символов, коды которых расположены в памяти по заданному адресу. Последний байт этой последовательности должен содержать признак конца текста — код 00H.

ОД оперирует только с шестнадцатеричными числами. Необходимые операции по преобразованию шестнадцатеричных цифр из символьного представления в двоичное и обратно выполняют программы 2-й группы:

#### ПРК16:

```

; *****
; ПОДПРОГРАММА ПРЕОБРАЗОВАНИЯ ШЕСТНАДЦАТЕРИЧНОЙ ЦИФРЫ
; ИЗ СИМВОЛЬНОГО ПРЕДСТАВЛЕНИЯ В ДВОИЧНОЕ.
; ВХОДНЫЕ ПАРАМЕТРЫ: (А) - КОД СИМВОЛА В КОИ-7.
; ВЫХОДНЫЕ ПАРАМЕТРЫ: (А) - ЗНАЧЕНИЕ ЦИФРЫ ( В МЛАДШЕМ
; ПОЛУБАЙТЕ); (СУ)=1, ЕСЛИ ВВЕДЕННЫЙ СИМВОЛ НЕ ЯВЛЯЕТСЯ
; ШЕСТНАДЦАТЕРИЧНОЙ ЦИФРОЙ.
; *****

```

FAA6 D630	SUI	30H	
FAA8 D8	RC		; ЕСЛИ КОД < 30H
FAA9 C6E9	ADI	0E9H	
FAAB D8	RC		; ЕСЛИ КОД > 46H
FAAC C606	ADI	06H	
FAAE F2B4FA	JP	ПЕР14	; ЕСЛИ КОД > 40H
FAB1 C607	ADI	07H	
FAB3 D8	RC		; ЕСЛИ КОД > 3AH
FAB4 C60A	ПЕР14: ADI	0AH	
FAB6 B7	ORA	A	; (СУ) = 0
FAB7 C9	RET		

#### ПР16К:

```

; *****
; ПОДПРОГРАММА ПРЕОБРАЗОВАНИЯ ЦИФРЫ ИЗ ДВОИЧНОГО ПРЕД-
; СТАВЛЕНИЯ В СИМВОЛЬНОЕ.
; ВХОДНОЙ ПАРАМЕТР: (А) - ЧИСЛО (В МЛАДШЕМ ПОЛУБАЙТЕ).
; ВЫХОДНОЙ ПАРАМЕТР: (С) - КОД ШЕСТНАДЦАТЕРИЧНОЙ ЦИФРЫ В
; КОДЕ КОИ-7.
; *****

```

FAB8 E60F	ANI	0FH	
FABA C690.	ADI	90H	
FABC 27	DAA		; УСТАНОВКА (СУ)=1 ДЛЯ ЧИСЕЛ > 9
FABD CE40	ACI	40H	
FABF 27	DAA		
FAC0 4F	MOV	C, A	
FAC1 C9	RET		

Программа ПРК16 преобразует цифру в системе КОИ-7 в двоичное число (шестнадцатеричную цифру). Шестнадцатеричным цифрам 0, 1, ..., 9, A, ..., F в системе КОИ-7 соответствуют коды 30H, 31H, ..., 39H, 41H, ..., 46H. Следовательно, для получения двоичных кодов цифр 0, ..., 9 достаточно вычесть из кода символа константу 30H,

а для кодов шестнадцатеричных цифр А,..., F — константу 37Н. Программа, кроме преобразования, производит проверку принадлежности значения входного параметра к множеству шестнадцатеричных цифр. В начале программы проверяется принадлежность кода символа к диапазону 30Н...46Н путем вычитания и сложения с соответствующими константами. При этом в аккумуляторе получается значение, соответствующее двоичному числу со смещением: 10Н — для шестнадцатеричных цифр А, ..., F и 17Н — для цифр 0, ..., 9. Для селекции этих двух типов цифр к содержимому аккумулятора добавляется 6. При этом для цифр 1-го типа возникает переполнение, и флаг S устанавливается в нуль. Для цифр 2-го типа выполняется дополнительная проверка превышения числа 39Н, после чего производится дополнительная коррекция для получения двоичного числа.

Обратное преобразование выполняет программа ПР16К. Для получения кода символа преобразуемой шестнадцатеричной цифры к ее значению добавляется константа 30Н, но по правилам десятичной арифметики. При этом для шестнадцатеричных цифр А, ..., F необходимо к полученной сумме добавить еще единицу. В программе указанное преобразование выполняется по-иному: к двоичным числам добавляется константа 90Н, что позволяет для данных цифр установить флаг CY в единицу, а затем с помощью команды ACI — константа 40Н.

Программы 3-й группы предназначены для ввода-вывода двух- и четырехразрядных шестнадцатеричных чисел:

#### ПРВВ:

```

; *****
; ПОДПРОГРАММА ВВОДА СИМВОЛА С КОНСОЛИ С СЕЛЕКЦИИ ТЕР-
; МИНАТОРА.
; ВЫХОДНЫЕ ПАРАМЕТРЫ: (A)—ВВЕДЕННЫЙ СИМВОЛ; (Z)=1, (CY)=0,
; ЕСЛИ ВВЕДЕН ПРОБЕЛ ИЛИ ", "; (Z)=1, (CY)=1, ЕСЛИ ВВЕДЕН ВК;
; (Z)=0, (CY)=0 В ПРОТИВНОМ СЛУЧАЕ.
; *****

```

FAC2 CD80FA	CALL	KBBW	; ВВОД СИМВОЛА
FAC5 FE20	ПРА: CPI	' '	
FAC7 C8	RZ		; ЕСЛИ ВВЕДЕН ПРОБЕЛ
FAC8 FE2C	CPI	','	
FACA C8	RZ		; ЕСЛИ ВВЕДЕНА ",",
FACB FE0D	CPI	ODH	
FACD 37	STC		
FACE C8	RZ		; ЕСЛИ ВВЕДЕН ВК
FACF 3F	CMC		
FAD0 C9	RET		

ВВ2ВН:

```

; *****
; ПОДПРОГРАММА ВВОДА С КОНСОЛИ ЧЕТЫРЕХРАЗЯДНОГО ШЕСТ-
; НАДЦАТЕРИЧНОГО ЧИСЛА.
; ВЫХОДНЫЕ ПАРАМЕТРЫ: (Н, L) - ВВЕДЕННОЕ ЧИСЛО; (CY) = 1, ЕСЛИ
; ПОСЛЕ ЧИСЛА ВВЕДЕН ВК.
; *****

```

```

FAD1 CDC2FA      CALL   PRVB   ; ВВОД СИМВОЛА
FAD4 CA66F8      JZ       ВОП    ; ЕСЛИ ВВЕДЕН ТЕРМИНАТОР
FAD7 210000      ВВ2ВН1: LXI   Н, О  ; ОБНУЛЕНИЕ НАКОПИТЕЛЯ
FADA 47          ЦИКЛ17: MOV    В, А
FADB CDA6FA      CALL   PRK16   ; ПРЕОБРАЗОВАНИЕ ВВЕДЕННОГО ЧИСЛА
FADE DAEDFA      JC       ПЕР16  ; ЕСЛИ НЕ ШЕСТНАДЦАТЕРИЧНАЯ ЦИФРА
; СДВИГ НАКОПИТЕЛЯ НА 4 РАЗРЯДА ВЛЕВО
FAE1 29          DAD      Н
FAE2 29          DAD      Н
FAE3 29          DAD      Н
FAE4 29          DAD      Н
; ДОБАВЛЕНИЕ ВВЕДЕННОЙ ЦИФРЫ К НАКОПИТЕЛЮ
FAE5 B5          ORA      L
FAE6 6F          MOV      L, А
; ВВОД СИМВОЛА СЛЕДУЮЩЕЙ ЦИФРЫ
FAE7 CD80FA      CALL   KBWЫ
FAEA C3DAFA      JMP      ЦИКЛ17
; ЗАВЕРШЕНИЕ ВВОДА
FAED 7B          ПЕР16: MOV    А, В
FAEE CDC5FA      CALL   ПРА     ; ПРОВЕРКА НА ТЕРМИНАТОР
FAF1 C266F8      JNZ     ВОП    ; ЕСЛИ НЕ ТЕРМИНАТОР
FAF4 C9          RET

```

ВВ2Б:

```

; *****
; ПОДПРОГРАММА ВВОДА С КОНСОЛИ ЧЕТЫРЕХРАЗЯДНЫХ ШЕСТ-
; НАДЦАТЕРИЧНЫХ ЧИСЕЛ.
; ВХОДНОЙ ПАРАМЕТР: (С) - КОЛИЧЕСТВО ВВОДИМЫХ ЧИСЕЛ.
; ВЫХОДНЫЕ ПАРАМЕТРЫ: (SP), (SP+1), ..., (SP+N) - ВВЕДЕННЫЕ
; ЧИСЛА (ПОСЛЕДНЕЕ ВВЕДЕННОЕ ЧИСЛО НАХОДИТСЯ В ВЕРШИНЕ
; СТЕКА).
; *****

```

```

FAF5 CDD1FA      CALL   ВВ2ВН   ; (Н, L) - ВВЕДЕННОЕ ЧИСЛО
; ЗАПИСЬ ВВЕДЕННОГО ЗНАЧЕНИЯ В СТЕК
FAF8 E3          XTHL
FAF9 E5          PUSH    Н
; ПРОВЕРКА НА ЗАВЕРШЕНИЕ ВВОДА
FAFA 0D          DCR      С
FAFB D202FB      JNC     ПЕР17   ; ЕСЛИ НЕ ВВЕДЕН ВК
FAFE C266F8      JNZ     ВОП    ; ЕСЛИ ВВЕДЕНЫ НЕ ВСЕ ЧИСЛА
FB01 C9          RET
FB02 C2F5FA      ПЕР17: JNZ     ВВ2Б

```

```

FB05 C366F8      JMP      ВОП    ; ЕСЛИ В КОНЦЕ НЕ ВВЕДЕН ВК
ВЫБА:

```

```

; *****
; ПОДПРОГРАММА ВЫВОДА НА КОНСОЛЬ ДВУХРАЗЯДНОГО ШЕСТНАД-
; ЦАТЕРИЧНОГО ЧИСЛА.
; ПАРАМЕТР: (А) - ВЫВОДИМОЕ ЧИСЛО.
; *****

```



```

FB08 F5          PUSH    PSW
FB09 0F          RRC
FB0A 0F          RRC
FB0B 0F          RRC
FB0C 0F          RRC
FB0D CD11FB      CALL    PER18 ; ВЫВОД СТАРШЕГО РАЗРЯДА
FB10 F1          POP     PSW
                ; ВЫВОД НА КОНСОЛЬ ШЕСТНАДЦАТЕРИЧНОЙ ЦИФРЫ ИЗ
                ; МЛАДШЕГО ПОЛУБАЙТА РЕГИСТРА A
FB11 CDB8FA      PER18: CALL PR16K ; ПРЕОБРАЗОВАНИЕ В КОД КОИ-7
FB14 C369FA      JMP     ВЫВС    ; ВЫВОД СИМВОЛА

                ВЫВНЛ:
                *****
                ; ПОДПРОГРАММА ВЫВОДА НА КОНСОЛЬ ЧЕТЫРЕХРАЗЯДНОГО ШЕСТ-
                ; НАДЦАТЕРИЧНОГО ЧИСЛА.
                ; ПАРАМЕТР: (H,L) - ВЫВОДИМОЕ ЧИСЛО.
                *****

FB17 7C          MOV     A,H
FB18 CD08FB      CALL    ВЫВА    ; ВЫВОД СТАРШЕГО БАЙТА
FB1B 7D          MOV     A,L
FB1C C308FB      JMP     ВЫВА    ; ВЫВОД МЛАДШЕГО БАЙТА

```

При вводе с консоли последовательности шестнадцатеричных чисел в качестве разделителей используются символы «—» и «,». Признаком конца ввода последовательности является символ ВК. Указанные символы называются *терминаторами*. Программа ПРВВ выполняет ввод символа с консоли, и если этот символ является терминатором, во флагах процессора Z и CY устанавливаются соответствующие значения. Вторая точка входа ПРА обеспечивает только проверку на терминатор символа, код которого записан в регистре (A).

Программа ВВ2БН производит ввод 4-разрядного числа с консоли в регистровую пару (H, L). Первые две команды выполняют ввод символа, и если он оказывается терминатором, управление передается в диспетчер по входу «Ошибка» (ВОП). Если ввод и проверку выполнять не обязательно, к данной программе необходимо обращаться по точке входа ВВ2БН1. Накопление вводимого числа в регистровой паре (H, L) осуществляется с помощью итеративного цикла. Перед началом цикла содержимое регистровой пары (H, L) обнуляется. На каждой итерации введенный символ с помощью программы ПРК16 преобразуется в двоичное представление шестнадцатеричной цифры и добавляется к предварительно сдвинутому на четыре разряда влево содержимому регистровой пары (H, L). Сдвиг влево производится с помощью четырех команд DAD H. Затем посредством

подпрограммы KBVI вводится следующий символ, и итерация повторяется. Работа цикла продолжается до тех пор, пока программа ПРК16 не обнаружит, что преобразуемый символ не является шестнадцатеричной цифрой. Если данный символ — терминатор, осуществляется возврат из программы. В противном случае управление передается в диспетчер на вход «Ошибка». Таким образом, эта программа вводит любое количество цифр, но число формируется только по последним четырем введенным цифрам.

Подпрограмма BV2B предназначена для ввода заданного количества 4-разрядных чисел, которые помещаются в стек (в вершине стека хранится последнее число). Если количество введенных чисел меньше заданного или за последним числом нет символа VK, возврата из программы не происходит, а управление передается в диспетчер на вход «Ошибка». Вывод 2-разрядного числа выполняется подпрограммой ВYBA: вначале старший полубайт выводимого числа преобразуется подпрограммой ПР16K в символьное представление и выводится на консоль, а затем — младший. В программе с целью экономии памяти последний фрагмент, который преобразует и выводит одну цифру числа, выполняется дважды, причем первый раз в качестве подпрограммы. Подпрограмма ВYBNL обеспечивает вывод на консоль 4-разрядного числа: подпрограммой ВYBA сначала выводятся старшие разряды, а затем — младшие.

### 6.6.3. ДИРЕКТИВА ВЫВОДА СОДЕРЖИМОГО ПАМЯТИ

Директива D осуществляет построчный вывод содержимого заданной области памяти. В каждой строке выводится 17 значений: первое является адресом, а остальные — содержимым последовательных ячеек памяти, начиная с данного адреса (значения разделяются символами «:» и «—»). Программа, выполняющая директиву, имеет вид:

```

KMD:
; *****
; ПОДПРОГРАММА ОБРАБОТКИ ДИРЕКТИВЫ D.
; *****
; ВВОД С ПУЛЬТА ПАРАМЕТРОВ ДИРЕКТИВЫ

F8B4 0E02      MVI      C,2
F8B6 CDF5FA    CALL     BV2B
F8B9 D1        POP      D          ; (D,E) - АДРЕС ВЕРХНЕЙ ГРАНИЦЫ
F8BA E1        POP      H          ; (H,L) - АДРЕС НИЖНЕЙ ГРАНИЦЫ

```

```

; ВЫВОД СОДЕРЖИМОГО ПАМЯТИ НА КОНСОЛЬ
F8BB CD8DFA ЦИКЛ5: CALL КОНС ; ВЫВОД ВК,ПС
F8BE CD17FB CALL ВЫВНЛ ; ВЫВОД НА КОНСОЛЬ (Н,Л)

; ВЫВОД НА КОНСОЛЬ СОДЕРЖИМОГО ЯЧЕЙКИ ПАМЯТИ
F8C1 CD79FA ЦИКЛ6: CALL ВЫВС1 ; ВЫВОД ":"
F8C4 3A DB ":"
F8C5 CD96FA CALL ВЫВП ; ВЫВОД ПРОБЕЛА
F8C8 7E MOV A,M
F8C9 CD08FB CALL ВЫВА ; ВЫВОД (A)

; ПРОВЕРКА НА ОКОНЧАНИЕ ВЫВОДА ОБЛАСТИ ПАМЯТИ
F8CC CDDFF8 CALL ИНКНД
F8CF DADBF8 JC ПЕР2 ; ЕСЛИ ВЫВОД ОКОНЧЕН

; ПРОВЕРКА НА ОКОНЧАНИЕ ВЫВОДА СТРОКИ
F8D2 7D MOV A,L
F8D3 E60F ANI OFH
F8D5 C2C1F8 JNZ ЦИКЛ6 ; ЕСЛИ ВЫВОД СТРОКИ НЕ ОКОНЧЕН
F8D8 C3BBF8 JMP ЦИКЛ5 ; НА ВЫВОД НОВОЙ СТРОКИ
F8DB CD8DFA ПЕР2: CALL КОНС ; ВЫВОД ВК,ПС
F8DE C9 RET

ИНКНД:
; *****
; ПОДПРОГРАММА СРАВНЕНИЯ РЕГИСТРОВЫХ ПАР (Н,Л) И (D,E).
; ВЫХОДНЫЕ ПАРАМЕТРЫ: ЕСЛИ (Н,Л)=0, ТО (CY)=1, (Z)=1; ЕСЛИ
; (Н,Л)=(D,E), ТО (CY)=0, (Z)=1; ЕСЛИ (Н,Л)>(D,E), ТО (CY)=1,
; (Z)=0; ЕСЛИ (Н,Л)<(D,E), ТО (CY)=0, (Z)=0.
; *****
F8DF 23 INX H
; ПРОВЕРКА (Н,Л) НА 0
F8E0 7C MOV A,H
F8E1 B5 ORA L
F8E2 37 STC
F8E3 C8 RZ ; ЕСЛИ (Н,Л) = 0

; СРАВНЕНИЕ (Н,Л) И (D,E)
F8E4 7B MOV A,E
F8E5 95 SUB L
F8E6 7A MOV A,D
F8E7 9C SBB H
F8E8 C9 RET

```

Вначале с помощью программы ВВ2Б производится прием двух параметров: адресов нижней и верхней границ отображаемой области памяти. Дальнейшая работа программы организована в виде двух вложенных итеративных циклов. Вывод одной строки обеспечивает внутренний цикл, на каждой итерации которого с помощью соответствующих подпрограмм выводятся разделители и содержимое очередного байта. Завершение работы внутреннего цикла происходит тогда, когда содержимое младшего полубайта адреса текущей выводимой ячейки памяти становится равным нулю. Во внешнем цикле выполняются переход на новую строку и вывод текущего значения адреса отображаемого байта. Выход из внешнего цикла осуществляется в том случае, когда значение текущего

адреса достигнет верхней границы отображаемой области памяти. Подпрограмма ИНКНД выполняет операции, необходимые для ведения адресных указателей. В этой подпрограмме содержимое регистровой пары (Н, L) инкрементируется и сравнивается с содержимым регистровой пары (D, E) и нулевым значением. В результате сравнения вырабатываются соответствующие значения выходных параметров.

#### 6.6.4. ДИРЕКТИВА ЗАПОЛНЕНИЯ ОБЛАСТИ ПАМЯТИ

Директива F заполняет ячейки памяти, адреса нижней и верхней границы которой задаются первым и вторым параметрами директивы, константой, значение которой указывается в третьем параметре:

```

КМФ:
; *****
; ПОДПРОГРАММА ОБРАБОТКИ ДИРЕКТИВЫ F.
; *****
; ВВОД С КОНСОЛИ ПАРАМЕТРОВ ДИРЕКТИВЫ
F8E9 0E03          MVI     C,3      ; (C)--КОЛИЧЕСТВО ПАРАМЕТРОВ
F8EB CDF5FA        CALL    BB2E
F8EE C1            POP     B        ; (C)--КОД ЗАПОЛНЕНИЯ
F8EF D1            POP     D        ; (D,E)--ВЕРХНЯЯ ГРАНИЦА ПАМЯТИ
F8F0 E1            POP     H        ; (H,L)--НИЖНЯЯ ГРАНИЦА ПАМЯТИ
; ЗАПОЛНЕНИЕ ПАМЯТИ
F8F1 71            ЦИКЛ7: MOV     M,C
F8F2 CDDFF8        CALL    ИНКНД   ; ПРОВЕРКА НА ОКОНЧАНИЕ РАБОТЫ
F8F5 D2F1F8        JNC     ЦИКЛ7   ; ЕСЛИ НЕ КОНЕЦ
F8F8 C9            RET

```

Вначале программа принимает три параметра, а затем с помощью итеративного цикла организуется заполнение памяти. Благодаря использованию подпрограммы ИНКНД тело цикла содержит всего три команды.

#### 6.6.5. ДИРЕКТИВА ПЕРЕМЕЩЕНИЯ СОДЕРЖИМОГО ОБЛАСТИ ПАМЯТИ

Директива M обеспечивает перемещение содержимого области памяти, границы которой определяются первым и вторым параметрами, в другую область памяти, начиная с адреса, задаваемого третьим параметром:

```

КММ:
; *****
; ПОДПРОГРАММА ОБРАБОТКИ ДИРЕКТИВЫ М.
; *****
; ВВОД С КОНСОЛИ ПАРАМЕТРОВ ДИРЕКТИВЫ

F8F9 0E03      MVI     C,3
F8FB CDF5FA     CALL    BB2B
F8FE C1         POP     B      ; (B,C) - НИЖНЯЯ ГРАНИЦА ПРИЕМНИКА
F8FF D1         POP     D      ; (D,E) - ВЕРХНЯЯ ГРАНИЦА ИСТОЧНИКА
F900 E1         POP     H      ; (H,L) - НИЖНЯЯ ГРАНИЦА ИСТОЧНИКА

; НЕПОСРЕДСТВЕННАЯ ПЕРЕСЫЛКА
F901 7E        MOV     A,M
F902 02         STAX    B
F903 03         INX     B
F904 CDDFF8     CALL    ИНКНД   ; ПРОВЕРКА НА ОКОНЧАНИЕ
F907 D201F9     JNC     ЦИКЛВ   ; ЕСЛИ НЕ ОКОНЧЕНО
F90A C9         RET

```

Структура программы аналогична структуре предыдущей программы. В теле цикла содержатся команды, осуществляющие пересылку текущего байта и модификацию области памяти приемника. Модификация области памяти источника и проверка на достижение им верхней границы выполняются посредством обращения к программе ИНКНД.

#### 6.6.6. ДИРЕКТИВА МОДИФИКАЦИИ СОДЕРЖИМОГО ОБЛАСТИ ПАМЯТИ

Директива S выполняется в диалоговом режиме и совмещает функции отображения и модификации содержимого области памяти. После ввода единственного параметра, который является адресом начала области памяти, на консоль выдаются содержимое первой ячейки этой области и символ «-». В ответ пользователь может ввести число, последние две цифры которого будут записаны в эту ячейку. Ввод числа должен заканчиваться вводом терминатора. Если запись в данную ячейку не требуется, необходимо сразу после вывода символа «-» ввести терминатор. В обоих случаях после ввода терминатора на консоль выводятся содержимое следующего байта и символ «-», затем описанные действия пользователь может повторить. Для прекращения работы директивы S необходимо в качестве терминатора использовать символ ВК. Программа, выполняющая директиву, имеет следующий вид:

KMS:

```
; *****
; ПОДПРОГРАММА ОБРАБОТКИ ДИРЕКТИВЫ S.
; *****
; ВВОД АДРЕСА ПЕРВОЙ ЯЧЕЙКИ ПАМЯТИ В (H,L)
```

```
F90B CDD1FA      CALL    BB2BH
F90E D8          RC          ; ЕСЛИ ПОСЛЕ АДРЕСА ВВЕДЕН ВК
                  ; ОТОБРАЖЕНИЕ СОДЕРЖИМОГО ЯЧЕЙКИ ПАМЯТИ
F90F 7E          ЦИКЛ9: MOV    A,M
F910 CD08FB      CALL    BABA
F913 CD79FA      CALL    BABC1 ; ВЫВОД "-"
F916 2D          DB          '-'
                  ; АНАЛИЗ СИМВОЛА, ВВЕДЕННОГО ПОСЛЕ ОТОБРАЖЕНИЯ
F917 CDC2FA      CALL    PRVB   ; ВВОД ОДНОБАЙТНОГО ЧИСЛА
F91A D8          RC          ; ЕСЛИ ВВЕДЕН ВК
F91B CA25F9      JZ      PER3   ; ЕСЛИ ВВЕДЕН ПРОБЕЛ
                  ; МОДИФИКАЦИЯ СОДЕРЖИМОГО ЯЧЕЙКИ
F91E EB          XCHG
F91F CDD7FA      CALL    BB2BH1 ; ВВОД ШЕСТНАДЦАТЕРИЧНОГО ЧИСЛА
F922 EB          XCHG
F923 73          MOV     M,E     ; ЗАПИСЬ ЧИСЛА В ПАМЯТЬ
F924 D8          RC          ; ЕСЛИ ПОСЛЕ ЧИСЛА ВВЕДЕН ВК
                  ; ПЕРЕХОД НА ОТОБРАЖЕНИЕ СОДЕРЖИМОГО СЛЕДУЮЩЕЙ ЯЧЕЙКИ
F925 23          PER3: INX     H
F926 C30FF9      JMP      ЦИКЛ9
```

Ввод параметра осуществляется подпрограммой BB2BH. Если сразу после параметра введен символ ВК, выполняется возврат из программы. Дальнейшая работа программы организована в виде цикла. На каждой итерации выводятся содержимое очередной ячейки памяти и символ «-», затем в зависимости от ответа пользователя выполняется (или не выполняется) запись нового содержимого. Для ввода ответа используются вспомогательные подпрограммы PRVB, с помощью которых выполняется ввод первого символа ответа, и BB2BH1 (дополнительная точка входа в подпрограмму BB2BH), которая вызывается, если первый введенный символ не является терминатором, и обеспечивает ввод нового содержимого в текущую ячейку памяти. В конце итерации значение указателя текущей ячейки инкрементируется. Возврат из программы производится в том случае, если в качестве терминатора введен символ ВК, что фиксируется установкой в единицу флага CY после возврата из подпрограммы PRVB или BB2BH1.

#### 6.6.7. ДИРЕКТИВА ОТОБРАЖЕНИЯ И МОДИФИКАЦИИ РЕГИСТРОВ

При обработке точек разрыва программы пользователя состояние регистров записывается в область сохранения, а при последующем запуске программы восстанавли-

вается. Поэтому с точки зрения пользователя модификация содержимого регистров МП является не чем иным, как модификацией содержимого соответствующих ячеек области сохранения. Если сразу после директивы X вводится символ ВК, выполняется только отображение содержимого регистров: на консоль выводится строка, в которой вместе с общепринятыми мнемоническими обозначениями регистров приводится их содержимое. Если после директивы X вводится мнемоническое имя регистра, на консоль выводятся его содержимое и символ «-». Дальнейшая работа данной директивы аналогична работе директивы S (только доступ организуется не к последовательным ячейкам памяти, а к ячейкам области сохранения, соответствующим регистрам, имена которых упорядочены по алфавиту). Программы, выполняющие директиву, имеют вид:

```

КМХ:
; *****
; ПОДПРОГРАММА ОБРАБОТКИ ДИРЕКТИВЫ X.
; *****
; ВВОД СИМВОЛА, СЛЕДУЮЩЕГО ЗА ДИРЕКТИВОЙ

F929 2186F9      LXI      H, ТРЕГ
F92C CDC2FA      CALL     PRBB      ; ВВОД СИМВОЛА
F92F DA6FF9      JC       PER8      ; ЕСЛИ ВВЕДЕН ВК
F932 0E0C      MVI       C, 0CH      ; (C) - ЧИСЛО ЭЛЕМЕНТОВ ТАБЛИЦЫ
; ПОИСК В ТАБЛИЦЕ ОБЛАСТИ СОХРАНЕНИЯ (ТРЕГ) ЭЛЕМЕНТА,
; СООТВЕТСТВУЮЩЕГО ВВЕДЕННОМУ СИМВОЛУ

F934 BE      ЦИКЛ10: CMP      H
F935 CA42F9      JZ       PER4      ; ЕСЛИ ЭЛЕМЕНТ НАЙДЕН
F938 23      INX      H
F939 23      INX      H
F93A 23      INX      H
F93B 0D      DCR      C
F93C C234F9      JNZ      ЦИКЛ10    ; ЕСЛИ НЕ ВСЯ ТРЕГ ПРОСМОТРЕНА.
F93F C366F8      JMP      ВОП      ; ВОЗРАТ В ДИСПЕТЧЕР
; ОРГАНИЗАЦИЯ РЕЖИМА МОДИФИКАЦИИ ОБЛАСТИ СОХРАНЕНИЯ
; РЕГИСТРОВ ПРОЦЕССОРА ДЛЯ ПРОГРАММ ПОЛЬЗОВАТЕЛЯ (ОСР)

F942 CD96FA      PER4: CALL     ВЫВП      ; ВЫВОД ПРОБЕЛА
F945 CDA5F9      ЦИКЛ11: CALL    ВЫВР      ; ВЫВОД ЭЛЕМЕНТА ОСР
F948 CD79FA      CALL     ВЫВС1     ; ВЫВОД "-"
F94B 2D      DB      '...'
F94C CDC2FA      CALL     PRBB
F94F DB      RC      ; ЕСЛИ ВВЕДЕН ВК
F950 CA66F9      JZ       PER7      ; ЕСЛИ ВВЕДЕН РАЗДЕЛИТЕЛЬ
; МОДИФИКАЦИЯ ЭЛЕМЕНТА ОСР

F953 E5      PUSH     H
F954 C5      PUSH     B
F955 CDD7FA      CALL     ВВ2БН1     ; ВВОД ЧИСЛА В (H,L)
F958 C1      POP      B
F959 F5      PUSH     PSW      ; СОХРАНЕНИЕ РАЗДЕЛИТЕЛЯ
F95A 7D      MOV      A, L
F95B 12      STAX     D      ; ЗАПИСЬ ЧИСЛА В ОСР

```

```

F95C 7B      MOV     A,B      ; (A) - АТТРИБУТ
F95D C1      POP     B        ; (C)-ПРИЗНАК РАЗДЕЛИТЕЛЯ
F95E B7      ORA     A
F95F FA65F9  JM      PER6     ; ЕСЛИ ЭЛЕМЕНТ ОДНОВАЙТНЫЙ
F962 13      INX     D
F963 7C      MOV     A,H
F964 12      STAX    D        ; ЗАПИСЬ СТАРШЕГО БАЙТА
F965 E1      PER6:  POP     H
                ; ПРОВЕРКА НА ЗАВЕРШЕНИЕ ПРОСМОТРА ТРЕГ
F966 AF      PER7:  XRA     A
F967 B6      ORA     H
F968 F8      RM                ; ЕСЛИ ПРОСМОТР ЗАВЕРШЕН
F969 C5      PUSH    B
F96A F1      POP     PSW     ; ВОССТАНОВЛЕНИЕ РАЗДЕЛИТЕЛЯ
F96B D8      RC                ; ЕСЛИ БЫЛ ВВЕДЕН ВК
F96C C345F9  JMP     ЦИКЛ11   ; НА ПРОДОЛЖЕНИЕ
                ; ОРГАНИЗАЦИЯ РЕЖИМА ОТОБРАЖЕНИЯ ОСР
F96F C8D5FA  PER8:  CALL    КОНС   ; ВЫВОД ВК,ПС
F972 CD96FA  ЦИКЛ12: CALL    ВВВР
                ; ПРОВЕРКА НА ОКОНЧАНИЕ ПРОСМОТРА ТРЕГ
F975 AF      XRA     A
F976 B6      ORA     H
F977 F8      RM                ; ЕСЛИ ПРОСМОТР ЗАВЕРШЕН
F978 4E      MOV     C,M      ; (C) - КОД ИМЕНИ РЕГИСТРА
F979 CD69FA  CALL    ВВВС     ; ВЫВОД ИМЕНИ РЕГИСТРА
F97C CD79FA  CALL    ВВВС1    ; ВЫВОД "="
F97F 3D      DB      '='
F980 CDA5F9  CALL    ВВВР
F983 C372F9  JMP     ЦИКЛ12   ; НА ПРОДОЛЖЕНИЕ

```

#### ТРЕГ:

; ТАБЛИЦА ОБЛАСТИ СОХРАНЕНИЯ

```

F986 41EA00  DB      'A',0EAH,0
F989 42E600  DB      'B',0E6H,0
F98C 43E500  DB      'C',0E5H,0
F98F 44E800  DB      'D',0E8H,0
F992 45E700  DB      'E',0E7H,0
F995 46E900  DB      'F',0E9H,0
F998 48Г500  DB      'H',0F5H,0
F99B 4CF400  DB      'L',0F4H,0
F99E 50F900  DB      'P',0F9H,1
F9A1 53EC01  DB      'S',0ECH,1
F9A4 FF      DB      OFFH    ; КОНЦЕВИК ТАБЛИЦЫ

```

#### ВВВР:

```

; *****
; ПОДПРОГРАММА ВЫВОДА НА КОНСОЛЬ ЭЛЕМЕНТА ОСР.
; ПАРАМЕТР: (H,L) - АДРЕС ЭЛЕМЕНТА ТРЕГ.
; *****
; ОПРЕДЕЛЕНИЕ АДРЕСА ОСР

```

```

F9A5 23      INX     H
F9A6 5E      MOV     E,M
F9A7 3A0400  LDA     4
F9AA 57      MOV     D,A      ; (D,E) - АДРЕС ОСР
F9AB 23      INX     H
F9AC 46      MOV     B,M      ; (B) - ПРИЗНАК ЭЛЕМЕНТА
F9AD 23      INX     H
                ; ВЫВОД СТАРШЕГО БАЙТА ЭЛЕМЕНТА
F9AE 1A      LDAX    D

```



F9AF C008FB	CALL	ВЫВА	
	; ВЫВОД	МЛАДШЕГО БАЙТА ЭЛЕМЕНТА	
F9B2 05	DCR	B	; ПРОВЕРКА ПРИЗНАКА
F9B3 F8	RM		; ЕСЛИ ЭЛЕМЕНТ ОДНОБАЙТНЫЙ
F9B4 1B	DCX	D	
F9B5 1A	LDAX	D	
F9B6 C308FB	JMP	ВЫВА	

Для обозначения доступа к произвольным ячейкам области сохранения существует специальная таблица ТРЕГ, с помощью которой по мнемоническому имени регистра определяется адрес ячейки области сохранения, где размещается его содержимое. Подпрограмма ВЫВР обеспечивает вывод содержимого ячейки (или ячеек) области сохранения по заданному адресу элемента таблицы. В таблице хранится только значение МЛБ адреса ячейки области сохранения. СТБ адреса потребуется из четвертой ячейки памяти (в ней хранится СТБ адреса верхней границы памяти). После формирования полного адреса на консоль с помощью программы ВЫВА выводится содержимое соответствующей ячейки. Если в третьем байте заданного элемента таблицы установлен признак двухбайтного значения (единица), дополнительно выводится содержимое предыдущей ячейки.

Программа КМХ обработки директивы X начинает работу с ввода параметра. Если в качестве параметра введен символ ВК, выполняется цикл отображений. На каждой итерации этого цикла обрабатывается один элемент таблицы области сохранения ТРЕГ. Вначале первый байт элемента проверяется на код «концевика» таблицы. Если «концевик» обнаруживается, выполняется возврат. В противном случае из таблицы выбирается и выводится на консоль символ мнемонического обозначения регистра, а затем после вывода символа «-» с помощью подпрограммы ВЫВР выводится содержимое этого регистра. Указатель таблицы размещается в регистровой паре (H, L). После возврата из программы ВЫВР указатель адресуется начало следующего элемента, поэтому отдельная модификация указателя не производится.

Если в качестве параметра вводится символ мнемонического обозначения регистра, выполняется фрагмент программы КМХ, реализующий режим модификации области сохранения. Вначале введенный код отыскивается среди первых байтов элементов таблицы. Если поиск неудачен, управление передается на вход ВОП программы

диспетчера. В противном случае выполняется цикл модификации, начиная с того регистра, символ которого введен в качестве параметра. Каждая итерация производит обработку одного элемента области сохранения. После вывода содержимого элемента и символа «-» выполняется ввод первого символа ответа пользователя. Если это символ ВК, осуществляется возврат из программы; если другой терминатор, выполняется переход на окончание цикла итерации; если символ шестнадцатеричной цифры, то подпрограммой ВВ2БН1 с консоли вводится число и в зависимости от содержимого третьего байта элемента таблицы в область сохранения записываются один или два байта введенного числа. В конце итерации первый байт следующего элемента проверяется на код «концевика» таблицы. В зависимости от результата проверки выполняется либо возврат из программы, либо следующая итерация цикла.

#### 6.6.8. ДИРЕКТИВА ЗАПУСКА ПРОГРАММЫ ПОЛЬЗОВАТЕЛЯ

Директива G позволяет выполнить программу пользователя, начиная с заданного адреса, который является первым параметром директивы. Второй и третий параметры определяют адреса точек останова, после достижения которых выполнение программы пользователя прекращается и управление передается программе диспетчера. Все параметры являются необязательными. Если опущен первый параметр, адресом старта является содержимое области сохранения, соответствующее программному счетчику РС. Программа, выполняющая директиву, имеет следующий вид:

```

KMG:
; *****
; ПОДПРОГРАММА ОБРАБОТКИ ДИРЕКТИВЫ G.
; *****
; ЗАГРУЗКА В СТЕК АДРЕСА ОБЛАСТИ СОХРАНЕНИЯ

F9B9 2A0300      LHL D      3
F9BC 2EED        MVI      L,0EDH
F9BE E3          XTHL      ; АДРЕС ЗАПУСКА В СТЕК

; ВВОД 1-ГО ПАРАМЕТРА

F9BF CDC2FA      CALL     PRVB ; КОНТРОЛЬ ТЕРМИНАТОРА
F9C2 CAD1F9      JZ       PER9 ; ЕСЛИ ВВЕДЕН ТЕРМИНАТОР
F9C5 CDD7FA      CALL     BB2BH1

; ЗАПИСЬ ТОЧКИ СТАРТА В ОБЛАСТЬ СОХРАНЕНИЯ

F9C8 EB          XCHG      ; (D,E) - АДРЕС СТАРТА
F9C9 2A0300      LHL D      3
F9CC 2EF9        MVI      L,0F9H
F9CE 72          MOV      M,D

```

F9CF 2B	DCX	H	
F9D0 73	MOV	H,E	
	; ВВОД 2-ГО И 3-ГО ПАРАМЕТРОВ		
F9D1 DAFEF9	PER9: JC	PER11	; ЕСЛИ ПОСЛЕ ДИРЕКТИВЫ ВВЕДЕН ВК
	; УСТАНОВКА СЧЕТЧИКОВ: (D)-ВВЕДЕННЫХ ПАРАМЕТРОВ,		
	; (E)-ДОПУСТИМОЕ ЧИСЛО ПАРАМЕТРОВ		
F9D4 110200	LXI	D,0002	
	; ЦИКЛ ВВОДА ПАРАМЕТРОВ		
F9D7 CD79FA	ЦИКЛ13: CALL	ВВВС1	; ВЫВОД "--"
F9DA 2D	DB	'--'	
F9DB CDD1FA	CALL	ВВ2БН	; ВВОД ПАРАМЕТРА В (H,L)
F9DE E5	PUSH	H	
F9DF 14	INR	D	
F9E0 DAEAF9	JC	PER10	; ЕСЛИ ПОСЛЕ ПАРАМЕТРА ВВЕДЕН ВК
	; ПРОВЕРКА НА ОКОНЧАНИЕ ВВОДА ПАРАМЕТРОВ		
F9E3 1D	DCR	E	
F9E4 C2D7F9	JNZ	ЦИКЛ13	
F9E7 C366F8	JMP	ВОП	; ВОЗВРАТ В МОНИТОР
	; ЗАПИСЬ ВВЕДЕННЫХ ПАРАМЕТРОВ В ОСР		
F9EA 2A0300	PER10: LHLD	3	
F9ED 2EFA	MVI	L,OFAN	
F9EF C1	ЦИКЛ14: POP	B	; (B,C)-ЗНАЧЕНИЕ ПАРАМЕТРА
F9F0 71	MOV	H,C	
F9F1 23	INX	H	
F9F2 70	MOV	H,B	
F9F3 23	INX	H	
	; УСТАНОВКА ТОЧКИ РАЗРЫВА		
F9F4 0A	LDAX	B	
F9F5 77	MOV	H,A	; СОХРАНЕНИЕ СОДЕРЖИМОГО
F9F6 23	INX	H	
F9F7 3EC7	MVI	A,0C7H	
F9F9 02	STAX	B	; ЗАПИСЬ RST 0 В ТОЧКУ РАЗРЫВА
F9FA 15	DCR	D	
F9FB C2EFF9	JNZ	ЦИКЛ14	
	; ПЕРЕХОД НА ТОЧКУ СТАРТА		
F9FE CDBDFA	PER11: CALL	КОНС	; ВЫВОД ВК,ПС
FA01 C9	RET		

В начале программы в вершину стека загружается адрес пусковой последовательности, размещаемой в рабочей области монитора. Это делается для того, чтобы по команде возврата RET управление было передано не программе диспетчера, а на начало пусковой последовательности (см. рис. 6.11), команды которой обеспечивают запись содержимого области сохранения в соответствующие регистры и передачу управления по адресу старта программы пользователя. После этого с помощью подпрограммы ПРВВ вводится первый символ первого параметра. Если символ является шестнадцатеричной цифрой, подпрограммой ВВ2БН1 вводится значение параметра и записывается в область сохранения. Если вводится терминатор, он игнорируется; если же в качестве терминатора вводится символ ВК, выполняется команда RET,

которая передает управление на начало пусковой последовательности. После ввода первого параметра выполняется цикл ввода второго и третьего параметров.

Так как эти параметры могут быть опущены, то для подсчета введенных параметров в регистре (D) организуется специальный счетчик. Вторым счетчик в регистре (E) служит для ограничения числа итераций цикла до двух (количество допустимых точек разрыва). На каждой итерации этого цикла через программу ВYBC1 производится вывод символа «-», а затем с помощью программы ВВ2БН вводится значение параметра и записывается в стек. После этого соответствующим образом модифицируются оба счетчика. Если обнаруживается, что допустимое число параметров исчерпано, управление передается на вход ВОП программы диспетчера. Выход из данного цикла производится в том случае, если после очередного параметра введен символ ВК, что фиксируется установкой в единицу флага CY после возврата из подпрограммы ВВ2БН. В момент выхода из цикла введенные параметры размещаются в вершине стека, а их количество — в регистре (D).

Следующий цикл обеспечивает формирование точек останова в программе пользователя. На каждой итерации этого цикла значение параметра извлекается из стека и записывается в область сохранения. Затем байт, который адресует данный параметр, также записывается в область сохранения, а вместо него записывается код команды рестарта RST 0 (0C7H). Количество итераций цикла определяется содержимым регистра (D). Заключительный фрагмент программы выполняет возврат, перед которым с помощью программы КОНС производится переход на новую строку консоли, что информирует пользователя о запуске его программы.

#### **6.6.9. ОБРАБОТКА ТОЧЕК РАЗРЫВА**

При достижении программой пользователя адреса останова, по которому записан код команды RST 0, управление передается программе PRST 0, обеспечивающей запись содержимого регистров в область сохранения и восстановление содержимого точек разрыва:

PRST0:

\*\*\*\*\*  
 ; ПОДПРОГРАММА ОБРАБОТКИ ТОЧЕК РАЗРЫВА.  
 \*\*\*\*\*  
 ; СОХРАНЕНИЕ РЕГИСТРОВ В СТЕКЕ ПРЕРВАННОЙ ПРОГРАММЫ

FA02	F3	DI	
FA03	E5	PUSH	H
FA04	D5	PUSH	D
FA05	C5	PUSH	B
FA06	F5	PUSH	PSW

; СОХРАНЕНИЕ РЕГИСТРОВ A,F,D,E,B И C В OCP

FA07	0603	MVI	B,3
FA09	2A0300	LHLD	3
FA0C	2EEB	MVI	L,0EBH
FA0E	2B	DCX	H
FA0F	D1	POP	D
FA10	72	MOV	M,D
FA11	2B	DCX	H
FA12	73	MOV	M,E
FA13	05	DCR	B
FA14	C20EFA	JNZ	ЦИКЛ15

; ЗАПИСЬ В OCP (H,L)

FA17	D1	POP	D
FA18	2EF5	MVI	L,0F5H
FA1A	72	MOV	M,D
FA1B	2B	DCX	H
FA1C	73	MOV	M,E

; ЗАПИСЬ В OCP РЕГИСТРА PC (АДРЕСА ТОЧКИ ОСТАНОВА)

FA1D	C1	POP	B
FA1E	0B	DCX	B ; (B,C) - АДРЕС ТОЧКИ ОСТАНОВА
FA1F	2EF9	MVI	L,0F9H
FA21	70	MOV	M,B
FA22	2B	DCX	H
FA23	71	MOV	M,C

; ЗАПИСЬ В OCP РЕГИСТРА SP

FA24	EB	XCHG	
FA25	210000	LXI	H,0
FA28	39	DAD	SP
FA29	EB	XCHG	
FA2A	2EEC	MVI	L,0ECH
FA2C	72	MOV	M,D
FA2D	2B	DCX	H
FA2E	73	MOV	M,E

; ИНИЦИАЛИЗАЦИЯ СТЕКА МОНИТОРА

FA2F	2EE4	MVI	L,0E4H
FA31	F9	SPHL	

; СРАВНЕНИЕ ТОЧКИ ОСТАНОВА С ПЕРВОЙ ТОЧКОЙ РАЗРЫВА

FA32	2EFA	MVI	L,0FAH
FA34	7E	MOV	A,M
FA35	91	SUB	C
FA36	23	INX	H
FA37	7E	MOV	A,M
FA38	98	SBB	B
FA39	CA46FA	JZ	ПЕР12 ; ЕСЛИ СОВПАДАЕТ

; СРАВНЕНИЕ ТОЧКИ ОСТАНОВА СО ВТОРОЙ ТОЧКОЙ РАЗРЫВА

FA3C	23	INX	H
FA3D	23	INX	H ; (H,L) - АДРЕС СОХРАНЕНИЯ
FA3E	7E	MOV	A,M
FA3F	91	SUB	C

FA40 23		INX	H	
FA41 7E		MOV	A,M	
FA42 98		SBB	B	
FA43 C200F8		JNZ	ИНМОН	; ЕСЛИ НЕ СОВПАДАЕТ
				; ВОССТАНОВЛЕНИЕ СОДЕРЖИМОГО ТОЧЕК РАЗРЫВА И ОБНУЛЕНИЕ
				; ОБЛАСТИ ИХ СОХРАНЕНИЯ
FA46 2EFA	ПЕР12:	MVI	L,0FАН	
FA48 1602		MVI	D,02	; (D) – КОЛИЧЕСТВО ТОЧЕК
FA4A AF	ЦИКЛ16:	XRA	A	
FA4B 4E		MOV	C,M	
FA4C 77		MOV	M,A	
FA4D 23		INX	H	
FA4E 46		MOV	B,M	
FA4F 77		MOV	M,A	
FA50 23		INX	H	
FA51 79		MOV	A,C	
FA52 B0		ORA	B	
FA53 CA58FA		JZ	ПЕР13	; ТОЧКА РАЗРЫВА НЕ УСТАНОВЛЕНА
FA56 7E		MOV	A,M	
FA57 02		STAX	B	
FA58 23	ПЕР13:	INX	H	
FA59 15		DCR	D	
FA5A C24AFA		JNZ	ЦИКЛ16	; ЕСЛИ НЕ ВСЕ ТОЧКИ ВОССТАНОВЛЕНЫ
FA5D C373F8		JMP	МОН	

Первоначально все регистры процессора сохраняются в стеке программы пользователя, из которого затем переписываются в область сохранения. При этом содержимое регистровых пар (PSW), (H, L), (D, E) и (B, C) переписывается с помощью итеративного цикла, а запись регистровых пар (H, L), (PC) и (SP) выполняется в отдельных фрагментах программы. Значение PC формируется из значения адреса возврата путем декрементирования его, так как при выполнении команды RST 0 адрес возврата указывает на следующую за ней команду. Содержимое указателя стека SP получается с помощью команды DAD SP при предварительно обнуленной регистровой паре (H, L).

После записи регистров в область сохранения в регистр (SP) загружается адрес начала стека монитора. Затем определяется, какая из двух точек разрыва «сработала». Для этого значения полученного адреса останова сравниваются с соответствующими элементами области сохранения, заполнение которых производилось при обработке директивы G. Если оба сравнения проходят безуспешно, это интерпретируется как сбой, и управление передается на вход инициализации монитора.

В заключительном фрагменте программы на основании информации, записанной в области сохранения, производится восстановление точек разрыва. Это осуществляется с помощью цикла, тело которого выполняется дважды (для каждой точки разрыва). Нулевое значение элемента области сохранения означает, что соответствующая точка разрыва не была установлена и восстановление содержимого этой точки не производится. Одновременно выполняется обнуление соответствующего элемента области сохранения. После завершения работы программы управление передается в диспетчер монитора.

#### 6.6.10. ДИРЕКТИВА ДОКУМЕНТИРОВАНИЯ

Директива Н позволяет установить или отменить режим копирования консольного вывода на печатающее устройство:

```

КМН:
; *****
; ПОДПРОГРАММА ОБРАБОТКИ ДИРЕКТИВЫ Н.
; *****
FA60 2A0300      LHLD      3
FA63 2EE4        MVI      L,0E4H ; (H,L) - АДРЕС ФЛАГА ПЕЧАТИ
FA65 7E          MOV      A,M
FA66 2F          CMA
FA67 77          MOV      M,A
FA68 C9          RET

```

Работа программы сводится к инвертированию флага печати, расположенного в рабочей области монитора. При его установке программа Вывс обеспечивает дублирование каждого выводимого на консоль символа выводом его на печатающее устройство.

#### 6.6.11. ДИСПЕТЧЕР ДИРЕКТИВ

Программа диспетчера обеспечивает инициализацию рабочей области, обработку ошибок пользователя, допущенных при вводе директив, ввод символа mnemonic обозначения директивы и вызов соответствующей подпрограммы обработки директивы. Программа диспетчера имеет вид:

ИНМОН:

\*\*\*\*\*  
; ПОДПРОГРАММА ИНИЦИАЛИЗАЦИИ МОНИТОРА.  
\*\*\*\*\*

```

F800 C303F8      JMP      ИНМОН+3 ; УСТАНОВКА СЧЕТЧИКА КОМАНД
; ОПРЕДЕЛЕНИЕ ВЕРХНЕЙ ГРАНИЦЫ ОЗУ
F803 210000      ГРОЗУ: LXI      H,0000H
F806 25          ЦИКЛ1: DCR      H      ; (H,L)-АДРЕС ПРОВЕРЯЕМОЙ ЯЧЕЙКИ
F807 46          MOV      B,H
; ПРОВЕРКА ЗАПИСИ НУЛЕЙ
F808 3E00        MVI      A,0
F80A 77          MOV      M,A
F80B B6          ORA      M
; ПРОВЕРКА ЗАПИСИ ЕДИНИЦ
F80C 2F          CMA
F80D 77          MOV      M,A
F80E 2F          CMA
F80F A6          ANA      M
F810 C206F8      JNZ      ЦИКЛ1      ; ЕСЛИ ЗАПИСЬ НЕВОЗМОЖНА
F813 70          MOV      M,B      ; ВОССТАНОВЛЕНИЕ СОДЕРЖИМОГО
; ЗАПИСЬ АДРЕСА ВЕРХНЕЙ ГРАНИЦЫ ОЗУ
F814 220300      SHLD     3
; УСТАНОВКА СТЕКА
F817 2EE4        MVI      L,0E4H
F819 F9          SPHL
; ИНИЦИАЛИЗАЦИЯ МОДУЛЕЙ СОПРЯЖЕНИЯ
F81A CD4FF4      CALL     ИИРПР      ; ДЛЯ ИРПР
F81D CD7AF4      CALL     ИИРПС      ; ДЛЯ ИРПС
; ОБНУЛЕНИЕ РАБОЧЕЙ ОБЛАСТИ
F820 AF          XRA      A
F821 77          ЦИКЛ2: MOV      M,A
F822 2C          INR      L
F823 C221F8      JNZ      ЦИКЛ2
; ЗАПОЛНЕНИЕ СТАРТОВОЙ ПОСЛЕДОВАТЕЛЬНОСТИ
; В РАБОЧЕЙ ОБЛАСТИ
F826 2EED        MVI      L,0EDH      ; (H,L)-НАЧАЛО ПОСЛЕДОВАТЕЛЬНОСТИ
F828 1159F8      LXI      D,ПУС      ; (D,E)-АДРЕС ШАБЛОНА
F82B 060C        MVI      B,ОСН      ; (B) - КОЛИЧЕСТВО БАЙТОВ
F82D 1A          ЦИКЛ3: LDAX     D
F82E 77          MOV      M,A
F82F 13          INX      D
F830 23          INX      H
F831 05          DCR      B
F832 C22DF8      JNZ      ЦИКЛ3      ; ЕСЛИ НЕ ВСЕ БАЙТЫ ПЕРЕПИСАНЫ
; УСТАНОВКА ПЕРЕХОДА НА "ЛОВУШКУ" ТОЧЕК РАЗРЫВА
F835 3EC3        MVI      A,ОСЗН      ; (A)-КОД КОМАНДЫ JMP
F837 320000      STA      0
F83A 2102FA      LXI      H,PRSTO
F83D 220100      SHLD     1
F840 2149F8      LXI      H,ТТ      ; ВЫВОД ПРЕДСТАВЛЕНИЯ
F843 CD9BFA      CALL     ТЕКСТ
F846 C373F8      JMP      МОН      ; ПЕРЕХОД НА ДИСПЕТЧЕР МОНИТОРА

```

ТТ:

; ТЕКСТ ПРЕДСТАВЛЕНИЯ:

```

F849 0D0A        DB      0DH,0AH
F84B 6D206F20     DB      'М О Н И Т О Р',0
F84F 6E206920

```



F853 74206F20  
F857 7200

ПУС:

; ШАБЛОН ПУСКОВОЙ ПОСЛЕДОВАТЕЛЬНОСТИ

F859 F3	DI	
F85A C1	POP	B
F85B D1	POP	D
F85C F1	POP	PSW
F85D E1	POP	H

F85E F9	SPHL	
F85F 210000	LXI	H,0
F862 FB	EI	
F863 C30000	JMP	0

ВОП:

\*\*\*\*\*  
; ДИСПЕТЧЕР МОНИТОРА.  
\*\*\*\*\*  
; УСТАНОВКА УКАЗАТЕЛЯ СТЕКА

F866 2A0300	LHLD	3	; (H,L) - ВЕРХНЯЯ ГРАНИЦА ОЗУ
F869 2EE4	MVI	L,0E4H	;
F86B F9	SPHL		

; ВЫВОД ПРИЗНАКА ОШИБКИ

F86C CD8DFA	CALL	КОНС
F86F CD79FA	CALL	ВЫВС1
F872 3F	DB	'??'

МОН:

; ВЫВОД ИНИЦИИРУЮЩЕГО СИМВОЛА

F873 CD8DFA	CALL	КОНС	; ВК, ПС
F876 CD79FA	CALL	ВЫВС1	; ВЫВОД "x"
F879 24	DB	'x'	
F87A CD80FA	CALL	КВВМ	; ВВОД ДИРЕКТИВЫ
F87D FE0D	CPI	ODH	
F87F CA73FB	JZ	МОН	; ЕСЛИ ВВЕДЕНО ВК

; ОПРЕДЕЛЕНИЕ АДРЕСА ПОДПРОГРАММЫ ОБРАБОТКИ

; ВВЕДЕННОЙ ДИРЕКТИВЫ С ИСПОЛЬЗОВАНИЕМ ТАБЛИЦЫ

F882 219FF8	LXI	H,ТАБК	; (H,L)-АДРЕС ТАБЛИЦЫ
F885 0607	MVI	B,(КТАБ-ТАБК+1)/3	

ЦИКЛ4:

F887 BE	CMR	M	
F888 CA95F8	JZ	ПЕР1	; ЕСЛИ ДИРЕКТИВА ОБНАРУЖЕНА
F88B 23	INX	H	
F88C 23	INX	H	
F88D 23	INX	H	; (H,L)-АДРЕС СЛЕДУЮЩЕГО ЭЛЕМЕНТА
F88E 05	DCR	B	
F88F C287F8	JNZ	ЦИКЛ4	; ЕСЛИ ПРОСМОТР НЕ ОКОНЧЕН
F892 C366F8	JMP	ВОП	; ЕСЛИ КОД ДИРЕКТИВЫ НЕ НАЙДЕН

; ЗАГРУЗКА В (H,L) АДРЕСА ПЕРЕХОДА НА ПОДПРОГРАММУ  
ОБРАБОТКИ ВВЕДЕННОЙ ДИРЕКТИВЫ

F895 23	ПЕР1:	INX	H
F896 7E		MOV	A,H
F897 23		INX	H
F898 66		MOV	H,H
F899 6F		MOV	L,A

; ПОДГОТОВКА АДРЕСА ВОЗВРАТА И ВЫЗОВ ПОДПРОГРАММЫ

F89A 1173F8	LXI	D,МОН
F89D D5	PUSH	D
F89E E9	PCHL	

## ТАБЛ:

## ; ТАБЛИЦА АДРЕСОВ ПОДПРОГРАММ ОБРАБОТКИ ДИРЕКТИВ

F89F 44	DB	'D'
F8A0 B4F8	DW	KMD
F8A2 46	DB	'F'
F8A3 E9F8	DW	KMF
F8A5 4D	DB	'M'
F8A6 F9F8	DW	KMM
F8A8 53	DB	'S'
F8A9 0BF9	DW	KMS
F8AB 58	DB	'X'
F8AC 29F9	DW	KMX
F8AE 47	DB	'G'
F8AF B9F9	DW	KMG
F8B1 48	DB	'H'
F8B2 60FA	DW	KMH

## КТАБ:

Инициализация монитора производится программой ИНМОН. Первая команда выполняет инициализацию счетчика команд МП, что позволяет размещать программу монитора в памяти, начиная с произвольного адреса. Затем выполняется процедура определения верхней границы оперативной памяти. Память просматривается страницами по 256 байт. Просмотр начинается с адреса 0FF00H и производится в сторону уменьшения адресов. Выполняются попытки записи в первый байт каждой страницы нулей и единиц с последующим контролем. Как только очередная попытка оказывается удачной, адрес этой страницы определяет верхнюю границу оперативной памяти. Значение данного адреса записывается в 3-ю и 4-ю ячейки памяти. Затем устанавливается значение указателя стека и вызываются подпрограммы инициализации драйверов, входящих в состав монитора, после чего обнуляется рабочая область, расположенная в верхних адресах памяти, и в нее копируется шаблон пусковой последовательности ПУС, указанный в теле монитора. Вслед за этими операциями в первые три ячейки памяти, начиная с нулевого адреса, записывается код команды безусловного перехода на программу обработки точек разрыва — создается «ловушка» точек разрыва. Инициализация завершается выдачей на консоль текста представления с меткой ТТ.

Основная точка входа в диспетчер имеет метку МОН. В первом фрагменте программы диспетчера выполняется вывод на консоль «побуждающего» символа, который информирует пользователя, что монитор готов к приему директив. Формирование адреса программы обработки введенной директивы осуществляется с помощью таблицы

ТАБК, которая состоит из трехбайтных элементов. Каждый элемент соответствует одной директиве и содержит код ее символа (первый байт) и адрес программы обработки. Код введенного символа сравнивается с содержанием первых байтов всех элементов таблицы. В случае удачного сравнения второй и третий байты элемента таблицы загружаются в регистровую пару (H, L), после чего в вершину стека помещается адрес точки входа в диспетчер (адрес возврата) и выполняется команда `PSHL`.

Если код введенного символа в таблице не найден, управление передается в ту часть диспетчера, которая выполняет обработку ошибок. По точке входа ВОП производится обработка ошибок пользователя, которая заключается в установке указателя стека в исходное состояние и выводе на консоль символа «?», который сообщает пользователю о допущенной ошибке. Установка указателя стека необходима, потому что передача управления на метку ВОП производится из подпрограмм ОД различного уровня вложения (при этом значение указателя стека не контролируется).

Рассмотренный монитор является простым и эффективным системным средством, позволяющим пользователю проводить отладку своих программ. Пуск и останов программ с возможностью отображения и модификации регистров МП и рабочих ячеек памяти — основной инструмент контроля и поиска ошибок в программах.

## ЗАКЛЮЧЕНИЕ

Умение создавать отдельные простые программы приходит к начинающим программистам обычно с первых практических занятий. Для этого достаточно ознакомиться с языком программирования, разобрать десяток примеров и попробовать самостоятельно решить несколько задач. Процесс самообучения должен проходить непосредственно за пультом микроЭВМ: каждую команду или оператор языка следует буквально «прощупать». «Интимный» диалог с машиной выявит все ошибки, вызванные поверхностным пониманием предмета. Такой подход гарантирует быстрое усвоение языка и развитие навыков программирования.

Разработка сложных многофункциональных программных комплексов, насчитывающих тысячи строк текста, требует качественно иного уровня подготовки программиста, который достигается освоением «философии программирования» и длительной практикой. В настоящем пособии сделана попытка описания некоторых аспектов этой философии на конкретном программном материале. Очевидно, что ее освоение связано с изучением строгих математических методов решения задач, хотя знание их само по себе еще не гарантирует успеха.

Процесс создания программ — творческий акт, основанный на свободе выбора языка программирования, его элементов и возможных путей синтеза (комбинирования) из них некоторой целостной, функционально законченной «конструкции», удовлетворяющей определенным критериям. Путь к конечному результату здесь всегда многовариантен и вместе с тем носит отпечаток ограничений внешней среды и личности программиста. В этом отношении программирование сродни любому другому виду творческой деятельности, будь то сочинение музыки или романов, или создание картин, и подчиняется тем же эстетическим критериям: выразительности, лаконичности, гармонии.

## ПРИЛОЖЕНИЯ

### 1. МИКРОПРОЦЕССОР СЕРИИ КР580 И ЕГО АРХИТЕКТУРА (ПРОГРАММНАЯ МОДЕЛЬ И НАБОР КОМАНД)

*Микропроцессор* (МП) серии КР580 (К580) представляет собой однокристалльный 8-разрядный процессор, который размещен в 40-контактном корпусе, имеет классическую трехшинную структуру с раздельными шинами адреса, данных, управления и выполняет фиксированный набор команд [2, 4, 12, 19, 20, 31]. Существует несколько модификаций этого МП: К580ИК80, К580ИК80А, КР580ИК80, КР580ИК80А (КР580ВМ80А), различающихся исполнением корпуса, быстродействием и некоторыми другими техническими характеристиками. Однако эти различия, за исключением особо отмеченных случаев, несущественны для последующего изложения, и поэтому под МП КР580 будем подразумевать любой из вышеперечисленных типов, а также их зарубежные прототипы Intel 8080 и 8080А [6, 16, 42, 59, 60, 63, 75]. Описание цоколевки, назначение выводов, внутренняя структура, принципы работы этих изделий и схемотехнические приемы построения на их основе МП систем подробно рассмотрены в литературе, и здесь нет необходимости их повторять.

Для разработки программ необходимо знание архитектуры микропроцессора и МП системы. *Архитектура* отражает возможности прикладного использования микропроцессора (а не его техническую реализацию) и содержит описание программной модели МП системы и набора команд, с помощью которых обеспечивается программный доступ к элементам модели [5, 6, 16, 20]. Под *программной моделью МП системы* понимается совокупность программно-доступных элементов (регистров), объединенных в систему посредством укрупненных направленных связей и дополнительных элементов, обеспечивающих достижение функциональной законченности и целостного представления модели (рис. П.1). Программная модель МП системы на базе МП КР580 содержит непосредственно модели МП, памяти М (MEMORY) и портов ввода-вывода I/O (INPUT/OUTPUT), объединенных тремя шинами: данных — ШД, адреса — ША и управления — ШУ. Порты ввода-вывода используются для сопряжения с различными системными (внутренними и внешними, или периферийными) устройствами.

*Модель МП* содержит следующие элементы.

1) шесть 8-разрядных *регистров общего назначения* (РОН) с однокбуквенными именами В, С, D, Е, Н, L. Эти регистры программно-доступны как автономно, так и попарно: как три 16-разрядные регистровые пары с однокбуквенными именами по первому, старшему регистру пары В — (В, С), D — (D, Е) и Н — (Н, L). Регистры и *регистровые пары* используются для временного хранения промежуточных данных, адресов (в качестве *сверхоперативного запоминающего устройства* — СОЗУ) и косвенной адресации основной памяти М (в качестве указателей памяти). Регистровая пара Н преимущественно



используется как указатель памяти в однобайтных командах МП, причем ячейка памяти, адрес которой указан в Н-паре, называется М-регистром. Этот регистр в функциональном отношении эквивалентен регистрам МП, но имеет большее время доступа;

2) 16-разрядный *регистр указателя стека* SP (STACK POINTER) — для хранения адреса вершины программного стека, размещаемого в оперативной памяти М и обеспечивающего необходимую глубину вложения подпрограмм при обработке многоуровневых прерываний и модульном построении программ;

3) 16-разрядный *регистр счетчика команд*, или *программный счетчик* PC (PROGRAM COUNTER), — для хранения адреса текущей команды выполняемой программы. При естественном ходе выборки команд, последовательно размещенных в памяти, содержимое счетчика увеличивается от команды к команде на число, равное количеству ячеек, занимаемых в памяти выполненной командой;

4) 8-разрядный А-регистр, или *аккумулятор* (накопитель), — основной, узловой рабочий регистр, используемый во всех арифметическо-логических командах, командах ввода-вывода данных и др. Совместно с F-регистром образует регистровую пару (A, F) *слова состояния процессора* PSW (PROCESSOR STATUS WORD). Это слово отражает результаты текущих преобразований данных в 8-разрядном *арифметическо-логическом устройстве* (АЛУ);

5) 8-разрядный F-регистр, или *регистр признаков* (флагов), — для хранения двоичных признаков — *бит*, отражающих некоторые особенности результата выполнения операции в АЛУ. Признаки могут использоваться последующими командами как для изменения естественной последовательности выборки команд из памяти (передачи управления, или перехода по программе), так и для модификации обрабатываемых данных. F-регистр фиксирует 5 различных признаков (3 разряда регистра не используются и содержат биты-константы):

S (SIGN) — *бит знака*, равен единице, если седьмой бит байта результата равен единице, т. е. результат — отрицательное число;

Z (ZERO) — *бит нуля*, равен единице, если результат операции равен нулю;

AC (AUXILIARY CARRY) — *бит вспомогательного переноса*, равен единице, если при выполнении операции был перенос из третьего разряда АЛУ в четвертый, используется в команде десятичной коррекции содержимого А-регистра;

P (PARITY) — *бит паритета*, равен единице, если число единиц результата операции четное;

CY (CARRY) — *бит переноса*, равен единице, если при выполнении операции был перенос из седьмого разряда АЛУ или заем в этот разряд (разряды нумеруются, начиная с младшего, нулевого разряда, размещаемого в регистре в крайней правой позиции). Признак очень важен при обработке данных с увеличенной разрядностью;

6) 1-разрядный *регистр (триггер) разрешения прерывания* INTE (INTERRUPT ENABLE). Если триггер установлен в нуль, МП не реагирует на запросы прерывания; если в единицу, прерывание разрешено. После приема запроса прерывания или сброса системы триггер INTE автоматически сбрасывается в нуль и для разрешения обработки последующих запросов прерывания его необходимо программно вновь установить в единицу;

7) связи между элементами МП и другими элементами МП системы: *внутренняя* двунаправленная 8-разрядная *шина данных* ВШД (8) — для передачи данных между элементами МП; *внешняя* двунаправлен-

ная 8-разрядная *шина данных* ШД(8) — для обмена данными между МП, памятью и портами ввода-вывода; однонаправленная 16-разрядная *шина адреса* ША(16) — для адресации памяти, портов; 10-разрядная *шина управления* ШУ(10), связанная с внутренним устройством управления (УУ) микропроцессора и предназначенная для временной коммутации элементов МП системы.

*Модель памяти* (М) представляет собой упорядоченную и пронумерованную последовательность 8-разрядных структурных элементов — *ячеек памяти*, или внешних регистров. 8-разрядное двоичное слово, хранимое в ячейке памяти (регистре), называется *байтом*, а отдельный двоичный разряд слова — *битом*. Номер ячейки памяти является ее *адресом*. 16-разрядная шина адреса МП позволяет обращаться к *адресному пространству* (максимальной совокупности адресуемых ячеек памяти) размером  $2^{16} = 65536 = 64 \text{ К байт}$  ( $K = 1024$  — общепринятая константа). Для нумерации адресов памяти используется шестнадцатеричная система счисления с цифрами 0, 1, ..., 9, A, B, C, D, E, F. Каждый байт может быть представлен двумя *полубайтами* (старшей и младшей тетрадами бит), значения которых однозначно кодируются указанными шестнадцатеричными цифрами. При этом адрес любой ячейки памяти представляется 4-разрядным, а ее содержимое — 2-разрядным шестнадцатеричными числами {0000, 0001, ..., FFFF} и {00, 01, ..., FF}.

Примем, что в модели памяти время доступа к содержимому любой ячейки памяти не зависит от значения ее адреса и над каждой ячейкой памяти может быть выполнена пара операций: запись байта в ячейку и чтение ее содержимого. Память такого типа называют *запоминающим устройством с произвольной выборкой* (ЗУПВ) или *оперативным запоминающим устройством* (ОЗУ) (*оперативной памятью*). Эта память используется для хранения программ, исходных, промежуточных и результирующих данных.

В реальных МП системах фактическое количество ячеек памяти — *емкость*, или *рабочее пространство*, памяти — может быть меньше адресного пространства и, кроме того, разделено на части по каким-либо конструктивным или функциональным признакам. Например, часть памяти может быть предназначена только для операции чтения хранимой информации. Такая память — *постоянное* (или *перепрограммируемое постоянное* — ППЗУ) *запоминающее устройство* (ПЗУ) — используется для хранения программ и констант. Для учета при программировании подобных реальных ограничений на использование адресного пространства применяют *карту памяти* — графическое распределение рабочего пространства памяти между отдельными блоками последовательных ячеек, сгруппированных по выделенным признакам [28].

*Модель портов ввода-вывода* представляет собой, как и модель памяти, упорядоченную и пронумерованную последовательность 8-разрядных регистров. В системе посредством однобайтных шестнадцатеричных адресов {00, 01, ..., FF} адресуются до 256 портов ввода и столько же портов вывода информации. Каждое системное устройство обменивается информацией (данными, адресами, управляющими сигналами) с МП путем посылки байта информации через соответствующий порт в А-регистр или приема байта из А-регистра. В качестве системных устройств ввода-вывода могут рассматриваться как периферийные устройства типа, например, АЦПУ, НГМД или дисплея, так и внутренние устройства типа, например, программируемого контроллера прерываний KP580BV59 или программируемого параллельного адаптера интерфейса KP580BV55.



Взаимодействие между элементами МП, памяти М и портов ввода-вывода I/O в программной модели МП системы сводится к выполнению четырех операций: записи информации в М и I/O из регистра МП и ее чтения из М и I/O в регистр МП. Вообще говоря, в системе возможны еще две операции: запись в М из порта I/O и чтение из М в порт I/O, которые появляются при организации прямого доступа в память (ПДП) со стороны системного устройства, например программируемого устройства прямого доступа КР580ВТ57 [2, 31]. Режим ПДП реализуется исключительно техническими средствами (за исключением программной инициализации устройства ПДП) и поэтому здесь не рассматривается. Последовательность функциональных взаимодействий между элементами программной модели МП системы определяется набором команд МП.

Выполнение команд в МП системе, как и в большинстве вычислительных систем, состоит из двух крупных фаз: *выборки* адресованной команды из памяти и ее *выполнения*, причем вторая фаза в ряде случаев состоит из двух полуфаз: *выборки операнда* из памяти и *выполнения операции над операндом*. При естественной последовательности выборки команд в заключение первой фазы или первой полуфазы второй фазы (если она имеет место) происходит автоматическое увеличение на единицу содержимого счетчика команд — адресация следующей команды, и после завершения выполнения текущей команды указанный двухфазный цикл повторяется уже для очередной команды. В реальных системах этот *командный цикл* выполняется с гораздо большей детализацией фаз, но для разработчика программ эти технические тонкости несущественны (они важны для схемотехнической реализации системы).

*Длительность командного цикла* индивидуальна для каждой команды и может быть выражена количеством тактов, или периодов, генератора синхронизации МП. Зная частоту генератора, можно определить реальное время выполнения команды и фрагментов программ. Длительность команд, оперирующих в МП системе с содержимым внутренних регистров, существенно короче, чем команд, использующих внешние регистры. Поэтому при необходимости минимизировать время выполнения программы желательно основную обработку информации вести с использованием внутренних регистров.

Функциональные возможности набора команд определяются в первую очередь форматом данных, команд и способами адресации операндов в командах. Для МП характерна малая разрядность основных структурных элементов (регистров), что приводит к использованию в МП преимущественно *одноадресных*, реже *двухадресных* (типа, например, команд межрегистровых пересылок) команд и различных способов непрямо́й адресации памяти, сокращающих длину командных слов и размеры программ.

*Формат данных* (базового информационного слова), принятый в МП КР580, — 8-разрядное двоичное слово ( $V_7, \dots, V_0$ ) — байт, где  $V_7$  — старший, а  $V_0$  — младший разряды байта. Хранение и пересылка всех данных в МП системе реализуются в виде последовательности байтов (побайтно), а для представления данных повышенной разрядности используются программно-организуемые *многобайтные слова*. Байт может использоваться для представления как целых двоичных чисел без знака в диапазоне от 0 до 255, так и чисел со знаком: целых положительных в диапазоне от 0 до 127 и отрицательных в диапазоне от  $-1$  до  $-128$  при представлении чисел в дополнительном коде (см. прил. 2). Интерпретация числового значения байта (число со знаком или без него) осуществляется программным путем. Для чисел со знаком старший,

седьмой разряд байта интерпретируется как знаковый бит: если он равен 0, число положительное; если — 1, — отрицательное. Кроме рассмотренных представлений байта в виде двоичных или шестнадцатеричных чисел, возможна их интерпретация в качестве двоично-десятичных чисел (байт содержит две десятичные двоично-кодированные цифры) или алфавитно-цифровых символов, используемых для обмена с периферийными устройствами (см. гл. 6).

*Формат команды (командного слова)* зависит от типа операции и может быть одно-, двух- или трехбайтным. Байты многобайтной команды обязательно размещаются в соседних ячейках памяти, и адрес первого байта является адресом команды в целом. Первый байт команды представляет код операции, второй и третий байты — данные или адрес. Заметим, что в трехбайтных командах старший и младший байты адреса или 16-разрядных данных меняются местами, т. е. вначале следует младший, а затем старший байт. Это является особенностью МП КР580, которую необходимо учитывать при программировании. К *однобайтным* относятся все команды межрегистровых пересылок (или команды типа регистр — регистр), ряд команд обращения к памяти, арифметических и логических операций, команды сдвига, возврата из подпрограммы, обращения к стеку, разрешения и запрещения прерываний; к *двухбайтным* — команды с непосредственными данными и ввода-вывода; к *трехбайтным* — команды переходов, вызова подпрограмм, загрузки регистровых пар и прямой записи в память.

В МП КР580 используются пять различных способов адресации.

*Прямая адресация*, при которой второй и третий байты команды содержат прямой адрес операнда в памяти.

*Регистровая, или неявная, адресация*, при которой адрес регистра источника и (или) приемника операнда определяется кодом команды.

*Регистровая косвенная адресация*, при которой адрес операнда находится в регистровой паре, адресуемой кодом команды.

*Непосредственная адресация*, при которой операнд размещается во втором байте — для двухбайтной команды или во втором и третьем байтах — для трехбайтной команды.

*Стековая адресация*, при которой адрес определяется указателем стека. Она отличается от регистровой косвенной адресации тем, что при обращении к памяти происходит запись или чтение двух байтов, а содержимое указателя стека автоматически соответственно уменьшается или увеличивается на 2. Информация в стеке хранится в том порядке, в котором туда поступает, а извлекается в обратном порядке, по принципу: «последним пришел — первым вышел».

Набор команд МП КР580 содержит 78 базовых команд, различающихся мнемоническим обозначением кода операции, а в целом включает 244 различные их модификации. Условные обозначения, используемые для описания набора команд, приведены в табл. П.1. В табл. П.2 дано описание набора команд [10, 36, 43, 55]. В табл. П.3 — П.5 представлены шестнадцатеричные значения кода операций команд с регистровой адресацией, а в табл. П.6 — определитель типа команд по кодам операций [12]. Этот определитель полезен при восстановлении исходного символического текста программы по ее объектному коду, т. е. при дезассемблировании программы.

Все команды МП подразделяются на 5 функционально специализированных групп:

1) *команды передачи данных* — используются для пересылки данных из регистра в регистр, из памяти в регистр (регистровую пару) и из регистра (регистровой пары) в память;

Табл. П.1. Условные обозначения в наборе команд

Обозначение	Комментарий
R1, R2, R M	Один из регистров A, B, C, D, E, H, L Ячейка памяти, адресуемая содержимым N-пары регистров
RM1, RM2, RM	Один из регистров A, B, C, D, E, H, L, ячейка памяти M или байт B2
RR	Одна из регистровых пар B — (B, C), D — (D, E), H — (H, L) или указатель стека SP
RRH, RRL	Старший и младший регистры регистровой пары RR
PCH, PCL	Старший и младший регистры программного счетчика PC
RH, RL	Старший и младший полубайты регистра (R)
R <sub>i</sub>	Разряд i регистра (R), где i=0, ..., 7
B2, B3	Второй и третий байты команды
N	Номер уровня (вектор) рестарта, где N=0, ..., 7
(X)	Содержимое элемента (адреса) X, где X — регистр, ячейка памяти, регистровая пара, байты команды или бит признака
((X))	Содержимое элемента, адресуемого по содержимому элемента X. Например, ((H, L)) — содержимое ячейки памяти, адрес которой указан в N-паре регистров
0,1	Состояния бита
←	Оператор присваивания: элемент слева от символа «←» заменяется элементом, находящимся справа от него
↔	Обмен элементами
ΛV	Логические операции И, ИЛИ, ИСКЛЮЧАЮЩЕЕ ИЛИ
—	Операция дополнения (инверсии) значения X
X	Знаки алгебраических сложения, вычитания и умножения. В колонке «Биты признаков» знак «+» означает, что бит устанавливается в зависимости от результата операции, а знак «—» — что бит не изменяется в данной операции
+, —, *	
n1/n2	Количество тактов, где n1 — число тактов при невыполнении условия, n2 — при его выполнении; n1, n2={4, 5, 7, 10, 11, 13, 16, 17, 18}

2) *команды арифметических операций* — применяются для двоичных операций сложения, вычитания, инкрементирования и декрементирования содержимого регистра (регистровой пары) или ячейки памяти, а также для сложения двоично-десятичных чисел;

3) *команды логических операций* — выполняют операции И, ИЛИ, ИСКЛЮЧАЮЩЕЕ ИЛИ (неравнозначность), сравнения и сдвига;

4) *команды передачи управления* — используются при условных и безусловных переходах в программе, вызовах подпрограмм и возвратах из них, рестартах подпрограмм при организации многоуровневых векторных прерываний;

Табл. П.2. Набор команд микропроцессора КР580

Формат, байтов	Время, тактов	Код	Мнемокод	Наименование команды	Описание операций	Биты признаков				
						S	Z	AC	P	CV
Команды передачи данных										
1	5	*	MOV R1, R2	Пересылка данных из регистра в регистр	(R1) ← (R2)	—	—	—	—	—
1	7	*	MOV R, M	Пересылка данных из памяти в регистр	(R) ← ((H, L))	—	—	—	—	—
1	7	*	MOV M, R	Пересылка данных из регистра в память	((H, L)) ← (R)	—	—	—	—	—
2	7	*	MVI R, B2	Пересылка непосредственных данных в регистр	(R) ← (B2)	—	—	—	—	—
2	10	36	MVI M, B2	Пересылка непосредственных данных в память	((H, L)) ← (B2)	—	—	—	—	—
3	10	*	LXI RR, B2B3	Непосредственная загрузка регистровой пары	(RRH) ← (B3), (RRL) ← (B2)	—	—	—	—	—
3	13	3A	LDA B2B3	Прямая загрузка A-регистра	(A) ← ((B3, B2))	—	—	—	—	—
3	13	32	STA B2B3	Прямое запоминание содержимого A-регистра	(B3, B2) ← (A)	—	—	—	—	—
1	7	*	LDAX RR	Косвенная загрузка A-регистра	(A) ← ((RR))	—	—	—	—	—
1	7	*	STAX RR	Косвенное запоминание содержимого A-регистра	((RR)) ← (A)	—	—	—	—	—
3	16	2A	LHLD B2B3	Прямая загрузка H-пары регистров	(L) ← ((B3, B2)), (H) ← ((B3, B2) + 1)	—	—	—	—	—

Продолжение табл. П.2

Формат, байтов	Время, такты	Код	Мнемокод	Наименование команды	Описание операций	Биты признаков			
						S	Z	AC	P CY
3	16	22	SHLDY B2B3	Прямое запоминание содержимого Н-пары регистров	$(B3, B2) \leftarrow (L),$	—	—	—	—
1	4	EB	XCHG	Обмен между Н- и D-парами регистров	$(B3, B2) + 1 \leftarrow (H)$ $(H) \leftrightarrow (D),$ $(L) \leftrightarrow (E)$	—	—	—	—
<i>Команды арифметических операций</i>									
1	4	*	ADD R	Сложение содержимых регистра и А-регистра	$(A) \leftarrow (A) + (R)$	+	+	+	+
1	7	86	ADD M	Сложение содержимых ячеек памяти и А-регистра	$(A) \leftarrow (A) + ((H, L))$	+	+	+	+
2	7	C6	ADI B2	Сложение непосредственных данных и содержимого А-регистра	$(A) \leftarrow (A) + (B2)$	+	+	+	+
1	4	*	ADC R	Сложение содержимых регистра и А-регистра с переносом	$(A) \leftarrow (A) + (R) + (CY)$	+	+	+	+
1	7	8E	ADC M	Сложение содержимых ячеек памяти и А-регистра с переносом	$(A) \leftarrow (A) + ((H, L)) + (CY)$	+	+	+	+
2	7	CE	ACI B2	Сложение непосредственных данных и А-регистра с переносом	$(A) \leftarrow (A) + (B2) + (CY)$	+	+	+	+

1	4	*	SUB R	Вычитание содержимого регистра из содержимого A-регистра	$(A) \leftarrow (A) - (R)$	+	+	+	+
1	7	96	SUB M	Вычитание содержимого ячейки памяти из содержимого A-регистра	$(A) \leftarrow (A) - ((H, L))$	+	+	+	+
2	7	D6	SUI B2	Вычитание непосредственных данных из содержимого A-регистра	$(A) \leftarrow (A) - (B2)$	+	+	+	+
1	4	*	SBB R	Вычитание содержимого регистра из содержимого A-регистра с заемом	$(A) \leftarrow (A) - (R) - (CY)$	+	+	+	+
1	7	9E	SBB M	Вычитание содержимого ячейки памяти из содержимого A-регистра с заемом	$(A) \leftarrow (A) - ((H, L)) - (CY)$	+	+	+	+
2	7	Df	SBI B2	Вычитание непосредственных данных из содержимого A-регистра с заемом	$(A) \leftarrow (A) - (B2) - (CY)$	+	+	+	+
1	5	*	INR R	Инкрементирование содержимого регистра	$(R) \leftarrow (R) + 1$	+	+	+	+
1	10	34	INR M	Инкрементирование содержимого ячейки памяти	$((H, L)) \leftarrow ((H, L)) + 1$	+	+	+	+
1	5	*	INX RR	Инкрементирование содержимого регистровой пары	$(RR) \leftarrow (RR) + 1$	-	-	-	-
1	5	*	DCR R	Декрементирование содержимого регистра	$(R) \leftarrow (R) - 1$	+	+	+	+
1	10	35	DCR M	Декрементирование содержимого ячейки памяти	$((H, L)) \leftarrow ((H, L)) - 1$	+	+	+	+
1	5	*	DCX RR	Декрементирование содержимого регистровой пары	$(RR) \leftarrow (RR) - 1$	-	-	-	-
1	10	*	DAD RR	Сложение содержимых регистровой пары и H-пары регистров	$((H, L)) \leftarrow ((H, L)) + ((RR))$	-	-	-	+

Продолжение табл. П. 2

Формат, байтов	Время, тактов	Код	Мнемокод	Наименование команды	Описание операций	Биты признаков			
						S	Z	AC	P

1	4	27	DAA	Десятичная коррекция содержимого А-регистра	Если $(AL) > 9$ или $(AC-1)$ , то $(A) \leftarrow (A) + 6$ ; если затем $(AH) > 9$ или $(CY) = 1$ , то $(A) \leftarrow (A) + 6 * 2^4$	+	+	+	+
									—

## Команды логических операций

1	4	*	ANA R	Поразрядное И над содержимым регистра и А-регистра	$(A) \leftarrow (A) \wedge (R)$	+	+	0	+
1	7	A6	ANA M	Поразрядное И над содержимым ячейки памяти и А-регистра	$(A) \leftarrow (A) \wedge \Lambda((H, L))$	+	+	0	+
2	7	E6	ANI B2	Поразрядное И над непосредственными данными и содержимым А-регистра	$(A) \leftarrow (A) \wedge (B2)$	+	+	0	+
1	4	*	XRA R	Поразрядное ИСКЛЮЧАЮЩЕЕ ИЛИ над содержимым регистра и А-регистра	$(A) \leftarrow (A) \vee (R)$	+	+	0	+
1	7	AE	XRA M	Поразрядное ИСКЛЮЧАЮЩЕЕ ИЛИ над содержимым ячейки памяти и А-регистра	$(A) \leftarrow (A) \vee \Lambda((H, L))$	+	+	0	+

2	7	EE	XRI B2	Поразрядное ИСКЛЮЧАЮЩЕЕ ИЛИ над непосредственными данными и содержимым А-регистра	$(A) \leftarrow (A) \vee (B2)$	+	+	0	+	0
1	4	*	ORA R	Поразрядное ИЛИ над содержимым регистра и А-регистра	$(A) \leftarrow (A) \vee (R)$	+	+	0	+	0
1	7	B6	ORA M	Поразрядное ИЛИ над содержимым ячейки памяти и А-регистра	$(A) \leftarrow (A) \vee V(H, L)$	+	+	0	+	0
2	7	F6	ORI B2	Поразрядное ИЛИ над непосредственными данными и содержимым А-регистра	$(A) \leftarrow (A) \vee (B2)$	+	+	0	+	0
1	4	*	CMP R	Сравнение содержимых регистра и А-регистра	Если $(A) - (RM) = 0$ , то $(CY) = 0$ , $(Z) = 1$ ; если $> 0$ , то $(CY) = 0$ , $(Z) = 0$ ; если $< 0$ , то $(CY) = 1$ , $(Z) = 0$	+	+	+	+	0
1	7	BE	CMP M	Сравнение содержимых ячейки памяти и А-регистра		+	+	+	+	0
2	7	E	CPI B2	Сравнение непосредственных данных с содержимым А-регистра		+	+	+	+	0
1	4	07	RLC	Циклический сдвиг влево содержимого А-регистра	$(A_{i+1}) \leftarrow (A_i),$ $(A_0) \leftarrow (A_7), (CY) \leftarrow \leftarrow (A_7)$	—	—	—	—	+
1	4	17	RAL	Циклический сдвиг влево содержимого А-регистра через перенос	$(A_{i+1}) \leftarrow (A_i),$ $(A_0) \leftarrow (CY),$ $(CY) \leftarrow (A_7)$	—	—	—	—	+
1	4	OF	RRC	Циклический сдвиг вправо содержимого А-регистра	$(A_i) \leftarrow (A_{i+1}),$ $(A_7) \leftarrow (A_0),$ $(CY) \leftarrow (A_0)$	—	—	—	—	+
1	4	1F	RAR	Циклический сдвиг вправо содержимого А-регистра через перенос	$(A_i) \leftarrow (A_{i+1}),$ $(A_7) \leftarrow (CY),$ $(CY) \leftarrow (A_0)$	—	—	—	—	+



Формат, байтов	Время, такты	Код	Мнемонкод	Наименование команды	Описание операций	Биты признаков				
						S	Z	AC	P	CY
1	4	2F	CMA	Дополнение содержимого A-регистра	$(A) \leftarrow (\bar{A})$	—	—	—	—	—
1	4	3F	CMC	Дополнение содержимого признака переноса	$(CY) \leftarrow (\bar{CY})$	—	—	—	—	+
1	4	37	STC	Установка в единицу признака переноса	$(CY) \leftarrow 1$	—	—	—	—	1
<i>Команды передачи управления</i>										
3	10	C3	JMP B2B3	Безусловный переход	$(PC) \leftarrow (B3, B2)$ Если условие верно, то $(PC) \leftarrow (B3, B2)$ , в противном случае $(PC) \leftarrow (PC) + 3$	—	—	—	—	—
3	10	DA	JC B2B3	Переход, если перенос		—	—	—	—	—
3	10	D2	JNC B2B3	Переход, если не перенос		—	—	—	—	—
3	10	CA	JZ B2B3	Переход, если нуль		—	—	—	—	—
3	10	C2	JNZ B2B3	Переход, если не нуль		—	—	—	—	—
3	10	F2	JP B2B3	Переход, если плюс		—	—	—	—	—
3	10	FA	JM B2B3	Переход, если минус		—	—	—	—	—
3	10	EA	JPE B2B3	Переход, если четно		—	—	—	—	—
3	10	E2	JPO B2B3	Переход, если нечетно		—	—	—	—	—
3	17	CD	CALL B2B3	Безусловный вызов подпрограммы		—	—	—	—	—

3	11/17	DC	CC B2B3	Вызов подпрограммы, если перенос	Если условие верно, то $(SP) - 1 \leftarrow (PCH)$ , $(SP) - 2 \leftarrow (PC)$ , $(SP) \leftarrow (SP) - 2$ , $(PC) \leftarrow$	—	—	—	—
3	11/17	D4	CNC B2B3	Вызов подпрограммы, если не перенос		—	—	—	—
3	11/17	CC	CZ B2B3	Вызов подпрограммы, если нуль		—	—	—	—
3	11/17	C4	CNZ B2B3	Вызов подпрограммы, если не нуль	$(B3, B2)$ , в противном случае $(PC) \leftarrow (PC) + 3$	—	—	—	—
3	11/17	F4	CP B2B3	Вызов подпрограммы, если плюс		—	—	—	—
3	11/17	FC	CM B2B3	Вызов подпрограммы, если минус		—	—	—	—
3	11/17	EC	CPE B2B3	Вызов подпрограммы, если четно		—	—	—	—
3	11/17	E4	CPO B2B3	Вызов подпрограммы, если нечетно	$(PCL) \leftarrow ((SP))$ , $(PCH) \leftarrow ((SP) + 1)$ , $(SP) \leftarrow (SP) + 2$	—	—	—	—
1	10	C9	RET	Возврат		—	—	—	—

Формат, байтов	Время, такты	Код	Мнемонакод	Наименование команды	Описание операций	Биты признаков				
						S	Z	AC	P	CV
1	5/11	D8	RC	Возврат, если перенос	Если условие верно, то (RCL) ← ((SP)), (PCH) ← ((SP)) + 1, (SP) ← ((SP)) + 2, в противном случае (PC) ← (PC) + 1	—	—	—	—	—
1	5/11	D0	RNC	Возврат, если не перенос		—	—	—	—	—
1	5/11	C8	RZ	Возврат, если нуль		—	—	—	—	—
1	5/11	C0	RNZ	Возврат, если не нуль		—	—	—	—	—
1	5/11	F0	RP	Возврат, если плюс		—	—	—	—	—
1	5/11	F8	RM	Возврат, если минус		—	—	—	—	—
1	5/11	E8	RPE	Возврат, если четно		—	—	—	—	—
1	5/11	E0	RPO	Возврат, если нечетно		—	—	—	—	—
1	11	C7	RST 0	Рестарт по 0-му уровню		—	—	—	—	—
1	11	CF	RST 1	Рестарт по 1-му уровню		—	—	—	—	—
1	11	D7	RST 2	Рестарт по 2-му уровню	((SP) - 1) ← ← (PCH), ((SP) - 2) ← ← (PCL), (SP) ← (SP) - 2, (PC) ← 8*N	—	—	—	—	—
1	11	DF	RST 3	Рестарт по 3-му уровню		—	—	—	—	—
1	11	E7	RST 4	Рестарт по 4-му уровню		—	—	—	—	—
1	11	EF	RST 5	Рестарт по 5-му уровню		—	—	—	—	—
1	11	F7	RST 6	Рестарт по 6-му уровню		—	—	—	—	—
1	11	FF	RST 7	Рестарт по 7-му уровню	(PCH) ← (H), (PCL) ← (L)	—	—	—	—	—
1	5	E9	PCHL	Запись содержимого H-пары регистров		—	—	—	—	—
1						—	—	—	—	—

*Команды стека, ввода-вывода и управления*

1	11	*	PUSH RR	Запись в стек содержимого регистровой пары	$((SP) - 1) \leftarrow ((RRH), ((SP) - 2)) \leftarrow ((RRL), (SP)) \leftarrow ((SP) - 2)$	—	—	—
1	11	F5	PUSH PSW	Запись в стек слова состояния процессора	$((SP) - 1) \leftarrow (A), ((SP) - 2) \leftarrow ((SP) - 2), ((SP) - 2) \leftarrow (F)$	—	—	—
1	11	*	POP RR	Чтение из стека содержимого регистровой пары	$(RRL) \leftarrow ((SP)), (RRH) \leftarrow ((SP) + 1), (SP) \leftarrow (SP) + 2$	—	—	—
1	11	F1	POP PSW	Чтение из стека слова состояния процессора	$(F) \leftarrow ((SP)), (A) \leftarrow ((SP) + 1), (SP) \leftarrow (SP) + 2$	+	+	+
1	18	E3	XTHL	Обмен между вершиной стека и H-парой регистров	$(L) \leftarrow ((SP)), (H) \leftarrow ((SP) + 1)$	—	—	—
1	5	F9	SPHL	Запись содержимого H-пары регистров в указатель стека	$(SP) \leftarrow (H, L)$	—	—	—
2	10	DB	IN B2	Ввод данных	$(A) \leftarrow ((B2))$	—	—	—
2	10	D3	OUT B2	Вывод данных	$((B2)) \leftarrow (A)$	—	—	—
1	4	FB	EI	Разрешение прерывания	$(INTE) \leftarrow 1$	—	—	—
1	4	F3	DI	Запрет прерывания	$(INTE) \leftarrow 0$	—	—	—
1	7	76	HLT	Останов	$(PC) \leftarrow (PC) + 1$ , СТОП	—	—	—
1	4	00	NOP	Пустая операция	$(PC) \leftarrow (PC) + 1$	—	—	—

**Примечание.** \* — см. табл. П.3—П.5.

Табл. П.3. Коды команд MOV RM1, RM2

RM1	RM2							
	B	C	D	E	H	L	M	A
B	40 <sup>1</sup>	41	42	43	44	45	46	47
C	48	49 <sup>1</sup>	4A	4B	4C	4D	4E	4F
D	50	51	52 <sup>1</sup>	53	54	55	56	57
E	58	59	5A	5B <sup>1</sup>	5C	5D	5E	5F
H	60	61	62	63	64 <sup>1</sup>	65	66	67
L	68	69	6A	6B	6C	6D <sup>1</sup>	6E	6F
M	70	71	72	73	74	75	76 <sup>2</sup>	77
A	78	79	7A	7B	7C	7D	7E	7F <sup>1</sup>

<sup>1</sup> Команда MOV R, R эквивалентна команде NOP.

<sup>2</sup> Команда MOV M, M запрещена, ее код совпадает с кодом команды HLT.

Табл. П.4. Коды команд с регистрами

Команда	RM							
	B	C	D	E	H	L	M	A
ADD RM	80	81	82	83	84	85	86	87
ADC RM	88	89	8A	8B	8C	8D	8E	8F
SUB RM	90	91	92	93	94	95	96	97
SBB RM	98	99	9A	9B	9C	9D	9E	9F
ANA RM	A0	A1	A2	A3	A4	A5	A6	A7
XRA RM	A8	A9	AA	AB	AC	AD	AE	AF
ORA RM	B0	B1	B2	B3	B4	B5	B6	B7
CMP RM	B8	B9	BA	BB	BC	BD	BE	BF
INR RM	04	0C	14	1C	24	2C	34	3C
DCR RM	05	0D	15	1D	25	2D	35	3D
MVI RM, B2	06	0E	16	1E	26	2E	36	3E

Табл. П.5. Коды команд с регистровыми парами

Команда	KK			
	B	D	H	P
1	2	3	4	5
LXI RR	01	11	21	31
LDAX RR	0A	1A	—	—
STAX RR	02	12	—	—

1	2	3	4	5
INX RR	03	13	23	33
DCX RR	0B	1B	2B	3B
DAD RR	09	19	29	39
PUSH RR	C5	D5	E5	—
POP RR	C1	D1	E1	—

5) *команды стека, ввода-вывода и управления* — применяются для управления прерыванием, останова МП, ввода-вывода данных и обращения к стеку, его указателю и вершине.

Особый тип образуют *команды рестарта* RST0, ..., RST7 из четвертой группы. Эти однобайтные команды формируются, как правило, специальными аппаратными средствами (контроллерами прерываний) в режиме прерывания работы МП. В этот режим МП входит, получив сигнал прерывания от системного устройства (при условии, что триггер INTE был установлен в единицу). При этом МП заканчивает цикл выполнения текущей команды (в программном счетчике содержится уже адрес следующей команды программы), сбрасывает в нуль триггер INTE и считывает с ШД(8) код команды рестарта, который выставляется контроллером прерываний.

Действие этой аппаратной команды аналогично действию команды вызова подпрограммы, но отличается, во-первых, тем, что ограничено возможностью вызова в соответствии с вектором прерывания лишь одной из восьми подпрограмм, размещаемых в первых 64 ячейках оперативной памяти, и, во-вторых, тем, что при выполнении команды рестарта не происходит, как при выполнении любой другой команды, инкрементирования содержимого программного счетчика. Благодаря этому в стек записывается адрес очередной команды прерываемой программы, а по команде возврата (в подпрограмме обслуживания прерывания) этот адрес извлекается из стека и управление передается в ту точку прерванной программы, куда поступило прерывание, — продолжается выполнение основной программы.

Таким образом, команды рестарта — это аппаратные аналоги команд вызова подпрограмм, инициализируемые сигналом прерывания в любой точке и в любой момент времени выполняемой программы. При таком неожиданном вызове подпрограммы необходимо обеспечить сохранение в стеке слова состояния процессора и содержимого РОН для возобновления после обслуживания прерывания нормального хода прерванной программы. Заметим, что модификации МП КР580 с индексом А обладают возможностью отработки в режиме прерывания не только команд рестарта, но любой последовательности команд, аппаратно формируемой в этом режиме контроллером прерываний. В частности, они воспринимают аппаратно формируемую трехбайтную команду CALL B2B3, что позволяет осуществлять вызов подпрограммы из любой области памяти.

В целом набор команд МП КР580 ориентирован на решение сложных задач управления данными, в частности создания систем, работающих в реальном масштабе времени. Эффективному решению этих задач способствует широкий спектр команд передачи данных, передачи управления, вызова и рестарта подпрограмм, управления прерыванием и обращения к стеку программируемой глубины.

Табл. П.6. Определитель

СТЦ	МЛЦ							
	0	1	2	3	4	5	6	7
0	NOP	LXI B (B2, B3)	STAX B	INX B	INR B	DCR B	MVI B	RLC
1	—	LXI D (B2, B3)	STAX D	INX D	INR D	DCR D	MVI D (B2)	RAL
2	—	LXI H (B2, B3)	SHLD (B2, B3)	INX H	INR H	DCR H	MVI H (B2)	DAA
3	—	LXI SP (B2, B3)	STA (B2, B3)	INX SP	INR M	DCR M	MVI M (B2)	STC
4	MOV B,B	MOV B,C	MOV B,D	MOV B,E	MOV B,H	MOV B,L	MOV B,M	MOV B,A
5	MOV D,B	MOV D,C	MOV D,D	MOV D,E	MOV D,H	MOV D,L	MOV D,M	MOV D,A
6	MOV H,B	MOV H,C	MOV H,D	MOV H,E	MOV H,H	MOV H,L	MOV H,M	MOV H,A
7	MOV M,B	MOV M,C	MOV M,D	MOV M,E	MOV M,H	MOV M,L	HLT	MOV M,A
8	ADD B	ADD C	ADD D	ADD E	ADD H	ADD L	ADD M	ADD A
9	SUB B	SUB C	SUB D	SUB E	SUB H	SUB L	SUB M	SUB A
A	ANA B	ANA C	ANA D	ANA E	ANA H	ANA L	ANA M	ANA A
B	ORA B	ORA C	ORA D	ORA E	ORA H	ORA L	ORA M	ORA A
C	RNZ	POP B	JNZ (B2, B3)	JMP (B2, B3)	CNZ (B2, B3)	PUSH B	ADI (B2)	RST 0
D	RNC	POP D	JNC (B2, B3)	OUT (B2)	CNC (B2, B3)	PUSH D	SUI (B2)	RST 2
E	RPO	POP H	JPO (B2, B3)	XTHL	CPO (B2, B3)	PUSH H	ANI (B2)	RST 4
F	RP	POP PSW	JP (B2, B3)	DI	CP (B2, B3)	PUSH PSW	ORI (B2)	RST 6

## 2. ТАБЛИЦЫ ЭКВИВАЛЕНТНЫХ ШЕСТНАДЦАТЕРИЧНО-ДЕСЯТИЧНЫХ ЗНАЧЕНИЙ ЧИСЛОВЫХ ДАННЫХ

В табл. П.7 приведены шестнадцатеричные и десятичные значения однобайтных данных при представлении беззнаковых двоичных чисел и чисел в дополнительном коде (H — старший, L — младший полубайты байта). Эта таблица удобна для быстрой интерпретации значения байта при написании и отладке программ. Табл. П.8, П. 9 необходимы для преобразования шестнадцатеричных и десятичных значений целых многобайтных чисел. Эти таблицы облегчают задачу тестирования числовых программ над полем конкретных данных.

## 3. ЯЗЫК МАКРОАССЕМБЛЕРА

Языки ассемблера и макроассемблера относятся к машинно-ориентированным языкам программирования низкого уровня. Эти языки специфичны для каждого типа микропроцессора, так как основаны на машинной системе команд, и, кроме того, связаны с используемой программой-ассемблером, т. е. транслятором с языка ассемблера в машинные коды. Так, программы, представленные в гл. 1—4 книги, написаны на макроассемблере (без использования микросредств) микроЭВМ «Электроника К1-10», а в гл. 5, 6 — на макроассемблере (с использованием макросредств) ДОС СМ 1800. Поскольку обе эти машины построены на базе микропроцессора серии КР580, их языки макроассемблера близки, но тем не менее имеют и некоторые различия.

8	9	A	B	C	D	E	F
—	DAD B	LDAX B	DCX B	INR C	DCR C	MVI C (B2)	RRC
—	DAD D	LDAX D	DCX D	INR E	DCR E	MVI E (B2)	RAR
—	DAD H	LHLD (B2, B3)	DCX H	INR L	DCR L	MVI L (B2)	CMA
—	DAD SP	LDA (B2, B3)	DCX SP	INR A	DCR A	MVI A (B2)	CMC
MOV C,B MOV E,B MOV L,B MOV A,B ADC B SBB B XRA B CMP B	MOV C,C MOV E,C MOV L,C MOV A,C ADC C SBB C XRA C CMP C	MOV C,D MOV E,D MOV L,D MOV A,D ADC D SBB D XRA D CMP D	MOV C,E MOV E,E MOV L,E MOV A,E ADC E SBB E XRA E CMP E	MOV C,H MOV E,H MOV L,H MOV A,H ADC H SBB H XRA H CMP H	MOV C,L MOV E,L MOV L,L MOV A,L ADC L SBB L XRA L CMP L	MOV C,M MOV E,M MOV L,M MOV A,M ADC M SBB M XRA M CMP M	MOV C,A MOV E,A MOV L,A MOV A,A ADC A SBB A XRA A CMP A
RZ	RET	JZ (B2, B3)	—	CZ (B2, B3)	CALL (B2, B3)	ACI (B2)	RST 1
RC	—	JC (B2, B3)	IN (B2)	CC (B2, B3)	—	SBI (B2)	RST 3
RPE	PCHL	JPE (B2, B3)	XCHG	CPE (B2, B3)	—	XRI (B2)	RST 5
RM	SPHL	JM (B2, B3)	EI	CM (B2, B3)	—	CPI (B2)	RST 7

Кратко опишем те языковые средства, которые являются одинаковыми (или почти одинаковыми) для обоих указанных языков ассемблера и использованы в приведенных программах.

Программа представляет собой последовательность *предложений языка ассемблера* — строк, ограниченных символом ВК. Возможны четыре типа предложений: 1) пустое предложение (содержит один символ ВК, улучшает восприятие исходного текста программы); 2) предложение машинной команды; 3) предложение вспомогательной команды языка (псевдокоманды); 4) предложение макрокоманды. Часто предложения второго и третьего типов называют *операторами языка ассемблера*. В общем случае предложение на языке ассемблера содержит 4 поля и имеет следующий вид:

**НАЗВАНИЕ ОПЕРАЦИЯ ОПЕРАНДЫ ;КОММЕНТАРИЙ (ВК)**

В конкретном предложении каждое из указанных полей может быть пустым. Если все поля пусты, то и предложение будет пустым. Поля предложения ограничиваются разделителями, которые могут быть дополнены произвольным количеством пробелов (говорят, что предложения записывается в свободном формате). В качестве разделителей применяются следующие знаки: пробел (разделяет поля), запятая (разделяет операнды в поле операндов), двоеточие (завершает имена, используемые в качестве меток), точка с запятой (открывает поле комментариев), апострофы (ограничивают строку символов) и скобки (ограничивают выражение).

Содержимое полей предложений исходной программы записывается



Табл. П.7. Определитель шестнадцатеричных и десятичных значений байта

СТЦ	МЛЦ															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
	-128	-127	-126	-125	-124	-123	-122	-121	-120	-119	-118	-117	-116	-115	-114	-113
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
	-112	-111	-110	-109	-108	-107	-106	-105	-104	-103	-102	-101	-100	-99	-98	-97

A	160 —96	161 —95	162 —94	163 —93	164 —92	165 —91	166 —90	167 —89	168 —88	169 —87	170 —86	171 —85	172 —84	173 —83	174 —83	175 —81
B	176 —80	177 —79	178 —78	179 —77	180 —76	181 —75	182 —74	183 —73	184 —72	185 —71	186 —70	187 —69	188 —68	189 —67	190 —66	191 —65
C	192 —64	193 —63	194 —62	195 —61	196 —60	197 —59	198 —58	199 —57	200 —56	201 —55	202 —54	203 —53	204 —52	205 —51	206 —50	207 —49
D	208 —48	209 —47	210 —46	211 —45	212 —44	213 —43	214 —42	215 —41	216 —40	217 —39	218 —38	219 —37	220 —36	221 —35	222 —34	223 —33
E	224 —32	225 —31	226 —30	227 —29	228 —28	229 —27	230 —26	231 —25	232 —24	233 —23	234 —22	235 —21	236 —20	237 —19	238 —18	239 —17
F	240 —16	241 —15	242 —14	243 —13	244 —12	245 —11	246 —10	247 —9	248 —8	249 —7	250 —6	251 —5	252 —4	253 —3	254 —2	255 —1

**Примечание.** Пример преобразования:  $AC_{16}=172_{10}$  или  $-84_{10}$ ;  $126_{10}=7E_{16}$ .

Табл. П.8. Шестнадцатеричный преобразователь десятичных чисел

Десяти- ная цифра	Десятичный множитель									
	1	10	10 <sup>2</sup>	10 <sup>3</sup>	10 <sup>4</sup>	10 <sup>5</sup>	10 <sup>6</sup>	10 <sup>7</sup>	10 <sup>8</sup>	
1	1	A	64	3E8	2710	186A0	F4240	989680	5F5E100	
2	2	14	C8	7D0	4E20	30D40	1E8480	1312D00	BEBC200	
3	3	1E	12C	BB8	7530	493E0	2DC6C0	1C9C380	11E1A300	
4	4	28	190	FA0	9C40	61A80	3D0900	2625A00	17D78400	
5	5	32	1F4	1388	C350	7A120	4C4B40	2FAF080	1DCD6500	
6	6	3C	258	1770	EA60	927C0	5B8D80	3938700	23C34600	
7	7	46	2BC	1B58	11170	AAE60	6ACFC0	42C1D80	29B92700	
8	8	50	320	1F40	13880	C3500	7A1200	4C4B400	2FAF0800	
9	9	5A	384	2328	15F90	DBBA0	895440	55D4A80	35A4E900	

Примечание. Пример преобразования:  $1234=10^3+2 \cdot 10^2+3 \cdot 10^1+4=3E8_{16}+C8_{16}+1E_{16}+4_{16}=4D2_{16}$ .

Табл. П.9. Десятичный преобразователь шестнадцатеричных чисел

Шестнадцатеричная цифра	Шестнадцатеричный множитель									
	1	16	16 <sup>2</sup>	16 <sup>3</sup>	16 <sup>4</sup>	16 <sup>5</sup>	16 <sup>6</sup>	16 <sup>7</sup>		
1	1	16	256	4096	65536	1048576	16777216	268435456		
2	2	32	512	8192	131072	2097152	33554432	536870912		
3	3	48	768	12288	196608	3145728	50331648	805306368		
4	4	64	1024	16384	262144	4194304	67108864	1073741824		
5	5	80	1280	20480	327680	5242880	83886080	1342177280		
6	6	96	1536	24576	393216	6291456	100663296	1610612736		
7	7	112	1792	28672	458752	7340032	117440512	1879048192		
8	8	128	2048	32768	524288	8388608	134217728	2147483648		
9	9	144	2304	36864	589824	9437184	1500994944	2415919104		
A	10	160	2560	40960	655360	10485760	167772160	2684354560		
B	11	176	2816	45056	720896	11534336	184549376	2922790016		
C	12	192	3072	49152	786432	12582912	201326592	3221225472		
D	13	208	3328	53248	851968	13631488	218103808	3489660928		
E	14	224	3584	57344	917504	14680064	234881024	3758096384		
F	15	240	3840	61440	983040	15728640	251658240	4026531840		

Примечание. Пример преобразования:  $FEDC_{16} = F \cdot 16^3 + E \cdot 16^2 + D \cdot 16^1 + C = 61440 + 3584 + 208 + 12 = 65244$ .

с помощью символов, образующих алфавит языка ассемблера, в качестве которых используются:

буквы латинского и русского алфавитов (как правило, прописные);

десятичные цифры от 0 до 9;

специальные символы (знаки): «+», «—», «\*», «/», «:», «.», «;», «( )», «?» и т. п.;

любые символы системы КОИ-7 (в строках, заключенных в апострофы, и в комментариях).

Основное отличие языка ассемблера от машинного языка заключается в использовании вместо двоичных кодов операций и адресов мнемонических и символических наименований — символических имен. *Символическое имя* в языке ассемблера — это последовательность букв и цифр, начинающаяся с буквы. Длина последовательности символов в имени не ограничивается, различными считаются символические имена, которые отличаются только первыми N символами (обычно  $N = 5 \dots 8$  и зависит от специфики ассемблера-транслятора). В языке используют два типа имен: *постоянные* и *определенные пользователем*. Постоянные имена не нуждаются в определении перед их использованием в исходной программе. К ним относятся мнемокоды машинных команд, мнемокоды псевдокоманд, имена регистров и регистровых пар, мнемокоды операций языка ассемблера. Символические имена пользователя следует обязательно определять в исходной программе, причем они не должны совпадать с постоянными именами (иначе ассемблер-транслятор выдаст сообщение об ошибке).

В *поле названия* предложения содержится либо имя, либо метка. *Имя* служит для идентификации элементов программы в псевдокомандах EQU, SET, MACRO (табл. П.10). Если имя используется для организации адресных ссылок на указанные операторы, оно называется *меткой*. Синтаксически метка определяется так же, как и имя, но отделяется от поля операции двоеточием. Числовое значение метки определяется по содержимому счетчика адреса ассемблера-транслятора при трансляции исходной программы. Метка всегда соответствует адресу первого байта машинной команды или области данных, которые она именует. *Поле операции* содержит либо имя машинной команды, либо имя псевдокоманды, либо имя макровывода. В *поле операндов* имеются один или несколько операндов, каждый из которых в общем случае представляет собой некоторое выражение языка ассемблера. *Поле комментария* содержит произвольную последовательность символов, которая используется для описания программных элементов при создании листинга программы. Предложение может состоять только из поля комментария, образуя предложение-комментарий.

*Выражения языка ассемблера*, определяющие операнды, — это комбинации термов, операций ассемблера, а также открывающих и закрывающих скобок. Числовые значения выражений вычисляются ассемблером-транслятором по модулю  $64K = 65536$ . Каждый *терм* представляет собой некоторое числовое значение, которое либо определяется самим термом (*самоопределенный терм*), либо присваивается выражению в результате вычислений ассемблера-транслятора. Самоопределенный терм представляет собой константу *двоичного* (последовательность двоичных цифр, оканчивающаяся буквой B), *восьмеричного* (последовательность восьмеричных цифр, оканчивающаяся буквой Q), *десятичного* (последовательность десятичных цифр с конечной буквой D или без нее), *шестнадцатеричного* (последовательность шестнадцатеричных цифр с конечной буквой H и предначальной цифрой 0, если первая шестнадцатеричная цифра является буквой) или *символьного*

Табл. П.10. Псевдокоманды макроассемблера

Имя	Формат оператора	Имя	Формат оператора
ORG	[МЕТКА:] ORG ВЫРАЖЕНИЕ	DB	[МЕТКА:] DB СПИСОК
END	[МЕТКА:] END	DW	[МЕТКА:] DW СПИСОК
EQU	ИМЯ EQU ВЫРАЖЕНИЕ	DS	[МЕТКА:] DS ВЫРАЖЕНИЕ
SET	ИМЯ SET ВЫРАЖЕНИЕ	MACRO	ИМЯ MACRO [СПИСОК]
		ENDM	[МЕТКА:] ENDM

**Примечание.** Квадратные скобки используются для обозначения необязательных элементов в формате оператора.

(последовательность символов, заключенная в апострофы и транслируемая в соответствующие системы КОИ-7) типов.

В качестве термина могут быть использованы имя, определенное предварительно псевдокомандами языка ассемблера EQU или SET (имена, определенные с помощью псевдокоманды SET, могут в последующем тексте программы переопределяться в отличие от имен, определенных псевдокомандой EQU), а также метка, имя регистра или регистровой пары и машинная команда языка ассемблера, взятая в круглые скобки. Выражения представляют собой последовательность вышеуказанных имен и констант, объединенных знаками арифметических, логических и некоторых других операций.

*Псевдокоманды* не имеют простых аналогов команд на машинном языке. Они предназначены для передачи ассемблеру-транслятору дополнительной информации о транслируемой программе. Можно выделить четыре класса псевдокоманд (см. табл. П.10): 1) *управления трансляцией* ORG, END; 2) *определения данных и выделения области памяти для данных* DB, DW, DS; 3) *определения символических имен* EQU, SET; 4) *определения макрокоманд* MACRO, ENDM.

Псевдокоманда ORG используется для установки начального адреса программы (ее фрагмента) или данных в памяти, а псевдокоманда END — для прекращения процесса трансляции исходной программы. Псевдокоманда DB формирует в оперативной памяти массив 8-битных констант, значения которых задаются в *списке* — последовательности выражений языка ассемблера, разделенных запятыми. Псевдокоманда DW аналогична команде DB, но формирует в отличие от нее массив 16-битных констант. Псевдокоманда DS используется для резервирования области памяти под данные (содержимое этой области ассемблер-транслятор не определяет).

Псевдокоманды MACRO и ENDM применяются для создания макросредств в языке ассемблера, тем самым превращая его в язык макроассемблера. *Макросредства* служат для повышения эффективности труда программиста. Потребность в них возникает тогда, когда в программе часто должны встречаться одинаковые или почти одинаковые последовательности команд. Макросредства позволяют неоднократно описать эти повторяющиеся последовательности с помощью макроопределений, а в тех местах программы, где встречаются эти

последовательности, помещать только краткие ссылки на них — макрокоманды (макровызовы).

*Макроопределение* представляет собой фрагмент текста программы, заключенный между псевдокомандами MACRO и ENDM. В поле названия псевдокоманды MACRO указывается имя определяемой макрокоманды, а в поле операндов — *список формальных параметров* — имен, имеющих в теле микроопределения, которые требуется изменять при раскрытии макроопределения. Место подстановки макроопределения в текст программы указывается с помощью *макрокоманды*, которая в поле операции содержит имя соответствующего макроопределения, а в поле операндов — *список фактических параметров*, подставляемых вместо формальных параметров. При трансляции макрокоманды вместо нее подставляется текст соответствующего макроопределения. Замена формальных параметров фактическими производится по принципу позиционного соответствия простой текстовой подстановкой: первый фактический параметр в списке заменяет первый формальный параметр и т. д. Наличие списка параметров в макрокоманде не является обязательным условием, а зависит от вида макроопределения.

#### 4. ПРОГРАММИРОВАНИЕ ПЕРИФЕРИЙНОГО ПАРАЛЛЕЛЬНОГО АДАПТЕРА КР580ВВ55

Микросхема КР580ВВ55 программируемого периферийного параллельного адаптера (ППА) предназначена для применения в МП системе в качестве универсального элемента ввода-вывода, обеспечивающего обмен данными в параллельном формате между МП и системными, в частности периферийными, устройствами. Аналогичное назначение, функции и структуру имеют микросхема К580ИК55 и зарубежный прототип ППА микросхема Intel 8255 [2, 3, 4, 12, 19, 31, 41, 42].

Условное обозначение ППА и его программная модель показаны на рис. П.2. Микросхема КР580ВВ55 размещена в 40-контактном корпусе и подключается к МП системе посредством двунаправленной трехстабильной 8-разрядной шины данных ШД(8), двухразрядной шины адреса ША(2) и 4-разрядной шины управления ШУ(4) с сигналами: ЗП — запись, ЧТ — чтение, ВМ — выбор микросхемы и СБР — сброс. Адаптер включает три программно-доступных 8-разрядных порта ввода-вывода (ПА, ПВ, ПС) и 7-разрядный регистр управляющего слова РУС, содержимое которого определяет направление передачи и функциональное назначение 24 двунаправленных трехстабильных линий ввода-вывода, т. е. конфигурацию и режимы работы портов. Порты ПА и ПВ предназначены для обмена байтами данных с системными устройствами, а порт ПС, как правило, — для обмена интерфейсными сигналами управления. Порт ПС в отличие от портов ПА и ПВ программно-доступен при операциях записи данных не только как элемент в целом, но и поразрядно, т. е. с независимой адресацией каждого отдельного разряда  $ПС_i$  ( $i = 0, 1, \dots, 7$ ), а в операциях выбора режима — как два полупорта: старший — ПСС, младший — ПСМ или их части. Обмен данными между элементами ППА и МП системой происходит через внутреннюю шину, связанную с ШД и через устройство управления (УУ) с шинами ША(2) и ШУ(4).

В МП системе, содержащей ППА, возможны два типа операций над его элементами: чтение (ввод) в МП содержимого адресуемого элемента и запись (вывод) из МП байта данных в адресуемый элемент ППА. Эти операции выполняются программно с помощью двух команд МП: IN B2 и OUT B2, где B2 — системный адрес конкретного порта

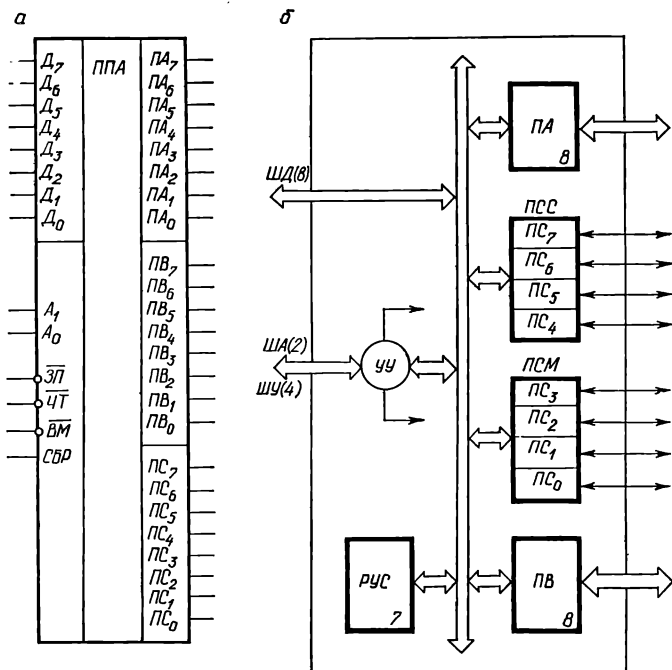


Рис. П.2. Микросхема ППА КР580ВВ55:

а — условное обозначение; б — программная модель

ППА (см. прил. 1). В процессе выполнения указанных команд в МП системе формируются сигналы управления, комбинация которых определяет ту или иную операцию над элементами ППА (табл. П.11). Два разряда адреса (они не обязательно должны совпадать с одноименными разрядами шины адреса МП) определяют выбор одного из трех портов или регистра. Заметим, что если для каждого порта существует пара операций чтение — запись, то для регистра РУС отсутствует операция чтения.

При начальной установке МП системы на вход СБР адаптера необходимо подать сигнал сброса. Этот сигнал устанавливает ППА в исходное состояние, при котором содержимое всех портов и регистра РУС обнуляется, а линии ввода-вывода устанавливаются в состояние ввода. После этого ППА доступен для программирования.

Программирование ППА, или его настройка, осуществляется с помощью операции записи *управляющего слова* (УС) в регистр РУС. Возможны два типа настройки ППА (табл. П.12).

1. Настройка разрядов — поразрядное программирование состояний порта РС: сброс в нуль или установка в единицу каждого отдельно адресуемого разряда РС<sub>i</sub>, независимо от состояний других разрядов РС. УС для этого типа программирования имеет вид (D<sub>7</sub>...D<sub>0</sub>) =



Табл. П.11. Сигналы управления и операции ППА

Тип операции	Операция	Сигналы управления				
		адрес		ЧТ	ЗП	ВМ
		A <sub>1</sub>	A <sub>0</sub>			
Чтение (Ввод)	ЩД ← ПА	0	0	0	1	0
	ЩД ← ПВ	0	1	0	1	0
	ЩД ← ПС	1	0	0	1	0
Запись (Вывод)	ЩД → ПА	0	0	1	0	0
	ЩД → ПВ	0	1	1	0	0
	ЩД → ПС	1	0	1	0	0
	ЩД → РУС	1	1	1	0	0

Отключение от ЩД

Нет действия

—	—	—	—	1
1	1	0	1	0

= (0 — — + + + +), где разряд (D<sub>7</sub>) = 0 определяет именно указанный тип настройки, значения разрядов (D<sub>6</sub>D<sub>5</sub>D<sub>4</sub>) произвольны и могут быть доопределены любой двоичной комбинацией; разряды (D<sub>3</sub>D<sub>2</sub>D<sub>1</sub>) задают двоичный адрес *i* разряда ПС<sub>*i*</sub>, а разряд D<sub>0</sub> — вид операции: сброс разряда ПС, если (D<sub>0</sub>) = 0, и его установку, если (D<sub>0</sub>) = 1. В графе 19 табл. П.12 приведены шестнадцатеричные значения УС, используемые в командах обращения к ППА. Так как значение старшей шестнадцатеричной цифры кода УС зависит от доопределения разрядов (D<sub>6</sub>D<sub>5</sub>D<sub>4</sub>) ∈ {0, 1, ..., 7}, то значения УС лежат в диапазоне чисел {00, 01, ..., 7F}. Заметим, что, хотя для настройки разрядов используется операция записи УС, содержимое регистра РУС при этом не изменяется (фиктивная запись). Для изменения состояния нескольких разрядов или формирования временной диаграммы на выходе определенного разряда ПС необходимо загружать в ППА соответствующую временную последовательность УС. Данный тип настройки позволяет использовать порт ПС в качестве регистра состояний периферийного устройства при организации программно-управляемого обмена данными.

2. Настройка режимов — программирование портов ПА, ПВ и ПС на один из трех возможных режимов или их комбинацию: 0 — режим простого однонаправленного обмена; 1 — режим стробируемого однонаправленного обмена; 2 — режим стробируемого двунаправленного обмена. Для настройки режимов используются УС, в которых разряд (D<sub>7</sub>) = 1 (в этом случае УС действительно записывается в регистр РУС). Пара разрядов УС (D<sub>6</sub>D<sub>5</sub>) определяет выбор режима для порта ПА: (00) — режим 0; (01) — режим 1 и (1 —) — режим 2; разряды D<sub>4</sub> и D<sub>3</sub> определяют соответственно для портов ПА и ПСС направления линий: если бит равен 1, линии имеют направление на ввод; если бит равен 0, — на вывод. Аналогично для порта ПВ разряд D<sub>2</sub> задает режим 0, если (D<sub>2</sub>) = 0, и режим 1, если (D<sub>2</sub>) = 1, а разряды D<sub>1</sub> и D<sub>0</sub> определяют соответственно для портов ПВ и ПСМ направления линий. Каждому режиму и конфигурации линий ввода-вывода однозначно соответствует определенное значение УС или группа его значений (из-за возможности произвольного доопределения некоторых разрядов УС), а

Табл. П.12. Режимы программирования ППА

Состояние портов										Управляющее слово											
ПА	ПСС							ПСМ			ПВ	Двоичный код								Н-код	
	ПС <sub>7</sub>	ПС <sub>6</sub>	ПС <sub>5</sub>	ПС <sub>4</sub>	ПС <sub>3</sub>	ПС <sub>2</sub>	ПС <sub>1</sub>	ПС <sub>0</sub>	Д <sub>7</sub>	Д <sub>6</sub>		Д <sub>5</sub>	Д <sub>4</sub>	Д <sub>3</sub>	Д <sub>2</sub>	Д <sub>1</sub>	Д <sub>0</sub>				
1	2	3	4	5	6	7	8	9	10	0 —											

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
	—	1	—	—	—	—	—	—						1	1	0		XD
	1	—	—	—	—	—	—	—						1	1	1		XF
Режим 0																		
										1	0	0	+	+	0	+	+	
										1	0	0			0			80
															0	0	1	81
															1	0	0	82
															1	1	1	83
										1	0	0	0	1	0	0	0	88
															0	0	1	89
															1	0	0	8A
															1	1	1	8B
															1	0	0	90
															0	0	1	91



Окончание табл. П.12

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Режим 2																		
	$\overline{\text{ГПД}}_A$	$\overline{\text{ППД}}_A$	$\overline{\text{ППР}}_A$	$\overline{\text{СТР}}_A$	$\text{ЗПР}_A$	$\rightarrow$	$\rightarrow$		$\rightarrow$	1	1	—	—	—	+	+	+	+
						$\leftarrow$	$\leftarrow$		$\rightarrow$	1	1	0	0	0	0	0	0	1
						$\rightarrow$	$\rightarrow$		$\rightarrow$									0
						$\leftarrow$	$\leftarrow$		$\rightarrow$									0
						$\rightarrow$	$\rightarrow$		$\rightarrow$									0
						$\leftarrow$	$\leftarrow$		$\rightarrow$									0
						$\rightarrow$	$\rightarrow$		$\rightarrow$									0
						$\leftarrow$	$\leftarrow$		$\rightarrow$									0
						$\rightarrow$	$\rightarrow$		$\rightarrow$									1
						$\leftarrow$	$\leftarrow$		$\rightarrow$									0
						$\rightarrow$	$\rightarrow$		$\rightarrow$									1
						$\leftarrow$	$\leftarrow$		$\rightarrow$									1
						$\rightarrow$	$\rightarrow$		$\rightarrow$									—
						$\leftarrow$	$\leftarrow$		$\rightarrow$									—

**Примечания:** 1. В режиме 1 разряд  $D_0$  и в режиме 2 разряды ( $D_5D_4D_3$ ) доопределены нулями. Возможны и другие доопределения, которые соответствующим образом изменят код УС.

2. Стрелка « $\rightarrow$ » означает направление всех линий данного элемента на вывод; стрелка « $\leftarrow$ » — на ввод.

3.  $X = (X_2X_1X_0)$ .

при записи нового УС в регистр РУС соответствующим образом меняется и настройка. Заметим, что настройка разрядов порта ПС не изменяет режима ППА.

В режиме 0 порты ПА, ПВ и полупорты ПСС, ПСМ могут быть настроены на любую из 16 возможных конфигураций однонаправленного ввода или вывода (см. табл. П.12).

Обмен данными реализуется по командам ввода-вывода МП синхронным или асинхронным способами. *Синхронный*, или *безусловный*, обмен предполагает безусловную готовность системного устройства выдать или принять данные по команде МП за строго фиксированный интервал времени. Такой обмен в МП системе используется только для быстрых и полностью определенных процессов, например процессов обмена между МП и основной памятью. *Асинхронный*, или *условный*, обмен предполагает, что готовность устройства к обмену появляется через произвольный, неопределенный интервал времени после подачи команды начала обмена. Поэтому такой обмен выполняется в два этапа: вначале устанавливается факт готовности устройства к обмену, а затем производится сам обмен данными.

В режиме 0 может быть реализован синхронный или асинхронный программно-управляемый обмен, причем во втором случае данные сопровождаются сигналами управления (квитирования), значения которых непрерывно контролируются программой управления обменом.

При операции ввода данные (или сигналы управления) периферийного устройства (ДПУ) через соответствующий порт ППА по сигналу  $\overline{CT}$  передаются на ШД МП системы и в аккумулятор, а в операции вывода — из аккумулятора на ШД и далее по сигналу  $\overline{ЗП}$  на выход соответствующего порта, а затем на вход ПУ. В интервале между командами обращения к ППА состояния линий выводов портов не изменяются, а возможные изменения состояний линий ввода не воспринимаются МП системой до очередной команды чтения ППА. Режим 0 используется, как правило, для ввода относительно медленно меняющихся (по сравнению с временем выполнения обслуживающей программы) данных ПУ, а также начальных условий и констант.

В режиме 1 порты ПА и ПВ могут быть настроены на любую из четырех возможных конфигураций однонаправленного стробируемого ввода или вывода. При этом в отличие от режима 0 два разряда полупорта ПСС и полупорт ПСМ однозначно настраиваются для каждой указанной конфигурации на ввод или вывод сигналов управления обменом, а два оставшихся разряда полупорта ПСС могут быть настроены на простой ввод или вывод для обмена одно- или двухразрядными данными. Если порт ПА настроен на вывод, свободно программируемыми разрядами полупорта ПСС являются  $PC_5$ ,  $PC_4$ ; если на ввод, —  $PC_7$ ,  $PC_6$ . При настройке порта ПА на вывод разряд  $PC_7$  используется для вывода на ПУ сигнала  $\overline{ГПД}$  — готовность к передаче (значение сигнала  $\overline{ГПД} = 0$  свидетельствует о готовности данных порта к передаче в ПУ), разряд  $PC_6$  — для ввода от ПУ сигнала  $\overline{ППД}$  — подтверждение передачи (значение сигнала  $\overline{ППД} = 0$  свидетельствует, что данные приняты ПУ), а разряд  $PC_3$  — для вывода в МП систему сигнала  $\overline{ЗПР}$  — запрос прерывания (значение сигнала  $\overline{ЗПР} = 1$  воспринимается МП как запрос прерывания). При настройке порта ПА на ввод разряд  $PC_5$  используется для вывода на ПУ сигнала  $\overline{ППР}$  — подтверждение приема (значение сигнала  $\overline{ППР} = 1$  свидетельствует, что данные от ПУ приняты МП), разряд  $PC_4$  — для ввода от ПУ сигнала  $\overline{СТР}$  —

строб (по значению сигнала  $\overline{СТР} = 0$  данные ПУ заносятся в порт ввода ППА), а разряд  $ПС_3$  — для вывода в МП систему сигнала ЗПР.

В режиме 1 реализуется асинхронный (или условный) обмен данными с использованием сигналов квитирования, причем в отличие от режима 0 с асинхронным обменом в данном режиме сигналы управления формируются аппаратно, поэтому отпадает необходимость в программном контроле за их состоянием и появляется возможность асинхронного обмена по прерываниям. Такой обмен использует ресурсы МП только на время непосредственного обмена данными (не тратятся ресурсы на ожидание готовности устройства к обмену). Вместе с тем в режиме 1 имеется возможность маскирования-демаскирования запросов прерывания ЗПР путем программирования разрядов порта  $ПС$ : прерывания разрешены, если в конфигурации ввода разряд ( $ПС_4$ ) = 1 для порта ПА (разряд  $ПС_2$  для порта ПВ), а в конфигурации вывода ( $ПС_6$ ) = 1 (разряд  $ПС_2$  для порта ПВ). Если эти разряды обращены в нуль, сигналы запроса не формируются, и МП система либо игнорирует обмен с конкретным ПУ, либо использует программный опрос состояний сигналов квитирования. Режим 1 используется, как правило, для обслуживания либо быстродействующих ПУ (чтобы избежать потерь информации при обмене), либо, наоборот, ПУ с большим временем доступа (чтобы избежать потерь производительности МП).

В режиме 2 порт ПА настраивается на конфигурацию двунаправленного стробируемого обмена с использованием полупорта  $ПСС$  и разряда  $ПС_3$  для ввода-вывода сигналов управления, а порт ПВ может быть настроен в режим 0 или 1. В первом случае группа из трех разрядов ( $ПС_2$ ,  $ПС_1$ ,  $ПС_0$ ) может свободно программироваться на ввод либо вывод, а во втором случае эти разряды используются для ввода-вывода сигналов управления обменом порта ПВ. В режиме 2 имеются те же возможности маскирования прерываний, что и в режиме 1. Ввод или вывод данных через порт ПА инициализируется сигналами запроса от ППА. Режим 2, как и режим 1, используется для управления быстродействующими ПУ по прерываниям, но с передачей данных по одной двунаправленной шине.

Режимы работы портов можно изменять как до, так и в процессе работы системных программ, что позволяет с помощью одного ППА обслуживать в определенном порядке различные ПУ (необходимо учитывать, что при изменении режима содержимое портов обнуляется).

## **5. ПРОГРАММИРОВАНИЕ ПЕРИФЕРИЙНОГО ПОСЛЕДОВАТЕЛЬНОГО АДАПТЕРА КР580ВВ51**

Микросхема КР580ВВ51 (КР580ВВ51А) программируемого периферийного адаптера, или, иначе, универсального синхронно-асинхронного приемопередатчика (УСАПП), предназначена для применения в МП системе в качестве универсального элемента ввода-вывода информации в последовательном формате (бит за битом) для обмена данными по каналу связи. Такой обмен используется для устройств с последовательным интерфейсом, удаленных периферийных устройств или в распределенных системах сбора и обработки данных. Прототипная микросхема — Intel 8251 (8251А) [2, 4, 12, 16, 19, 31, 41, 42].

Условное обозначение УСАПП и его программная модель показаны на рис. ПЗ, а назначения сигнальных выводов раскрыты в табл. П.13.

Микросхема КР580ВВ51 размещена в 40-контактном корпусе и подключается к МП системе посредством двунаправленной трехстабильной 8-разрядной шины данных ШД(8) и 6-разрядной шины управления ШУ(6) с сигналами  $\overline{ВМ}$ ,  $\overline{ЗП}$ ,  $\overline{ЧТ}$ ,  $У/Д$ , СИН и СБР. Сигналы ГПД, КПД, ГПР, ВСИН могут быть использованы для запросов прерывания при организации обмена по прерываниям между МП и УСАПП или в качестве условий, программно-доступных через слово состояния при организации программно-управляемого обмена. К стыку с каналом связи УСАПП подключается непосредственно своим выходом ДПД и (или) входом ДПР. Сигналы  $\overline{ЗПР}_T$ ,  $\overline{ГПР}_T$ ,  $\overline{ЗПД}_T$  и  $\overline{ГПД}_T$  используются для управления обменом между УСАПП и стыком (модемом). Линии сигналов  $\overline{СПД}$  и  $\overline{СПР}$  служат для подключения генераторов синхронизирующих импульсов (СИ) соответственно передачи и приема данных.

УСАПП содержит семь программно-доступных регистров: 8-разрядные регистры данных (РД), состояния (РС), режима (РР), команд (РК), первого синхросимвола (РСС1), второго синхросимвола (РСС2) и 13-разрядный, но доступный, как 8-разрядный, регистр передатчика (РПД) (дополнительные 5 разрядов регистра используются для записи старт-бита, до двух стоп-бит, бита паритета и служебного бита). РПД служит для преобразования параллельного формата передаваемых данных в последовательный. УСАПП содержит также два 9-разрядных регистра приемника РПР1 и РПР2 (девятый разряд регистров предназначен для приема бита паритета), образующих двойной буфер приема. РПР1 используется для преобразования последовательного формата принятых данных в параллельный. Двойная буферизация приема

Табл. П.13. Сигнальные выводы УСАПП

Обозначение	Наименование и назначение
1	2
Д7, ..., Д0	Двунаправленные трехстабильные линии для обмена с МП информацией: данными, словами управления и состояния
ВМ	Выбор микросхемы, вход сигнала
ЗП	Запись, вход сигнала записи информации с ШД в адресуемый регистр УСАПП
ЧТ	Чтение, вход сигнала чтения информации из адресуемого регистра УСАПП на ШД
У/Д	Управление / данные, вход сигнала идентификации типа информации: если $(У)=0$ — данные; если $(У)=1$ , то при записи — управляющее слово, а при чтении — слово состояния
СИН	Синхронизация, вход сигнала от тактового генератора МП для выработки служебной синхронизации УСАПП
СБР	Сброс, вход сигнала сброса УСАПП в исходное состояние ожидания последовательности УС
ВСИН	Вид синхронизации, двунаправленная программируемая линия. В синхронном режиме с внутренней синхронизацией является выходом и устанавливается в единицу после приема синхросимвола (или двух синхросимволов), сбрасывается в нуль после чтения слова состояния. В ре-



1	2
	<p>жиге с внешней синхронизацией является входом: появление единицы на этом входе служит для УСАПП сигналом приема данных по ДПР (сигнал можно снять через один период синхроимпульсов на входе СПР). Сигнал ВСИН доступен программно через слово состояния. В асинхронном режиме сигнал ВСИН является выходным и индицирует ошибку в принятых данных</p>
СПД	<p>Синхронизация передатчика, вход сигнала синхронизации передачи данных по выходу ДПД. В синхронном режиме данные выдаются на выход ДПД по каждому фронту сигнала СПД, т. е. скорость передачи равна частоте синхронизации. В асинхронном режиме скорость передачи программно устанавливается в 16 или 64 раза меньше частоты синхронизации</p>
СПР	<p>Синхронизация приемника, вход сигнала синхронизации приема данных по входу ДПР. Данные записываются в РПР1 по срезу СПР. Использование синхронизации в синхронном и асинхронном режимах аналогично использованию СПД</p>
ГПР <sub>т</sub>	<p>Готовность приемника терминала. Вход сигнала готовности внешнего приемника (модема) принять данные от УСАПП. При ГПР<sub>т</sub>=0 разрешена передача данных на выход ДПД (если передача разрешена словом режима), в противном случае запрещена</p>
ГПД <sub>т</sub>	<p>Готовность передатчика терминала. Вход сигнала готовности внешнего передатчика (модема) передать данные в УСАПП. Состояние входа может быть программно опрошено в слове состояния</p>
ДПР	Данные приемника, вход
ДПД	Данные передатчика, выход
ЗПД <sub>т</sub>	<p>Запрос передатчика терминала, выход сигнала запроса о готовности терминала к передаче данных в УСАПП (готовность приемника УСАПП к приему данных по ДПР)</p>
ЗПР <sub>т</sub>	<p>Запрос приемника терминала, выход сигнала запроса о готовности терминала к приему данных от УСАПП (готовность передатчика УСАПП к выдаче данных по ДПД)</p>
КПД	<p>Конец передачи, выход сигнала условия окончания передачи данных (РПД пуст). Может быть программно опрошен в слове состояния</p>
ГПР	<p>Готовность приемника, выход сигнала условия готовности приемника УСАПП передать данные в МП. Сбрасывается по срезу сигнала ЧТ при чтении данных (но не состояния). Может быть программно опрошен в слове состояния</p>
ГПД	<p>Готовность передатчика, выход сигнала условия готовности передатчика УСАПП принять данные от МП.</p>

1	2
---	---

Сбрасывается по срезу сигнала ЗП при записи данных (но не команды). Может быть программно опрошен в слове состояния

позволяет каналу работать с максимальной скоростью передачи, предоставляя вместе с тем МП время на обработку, равное длительности приема одного символа. Данные, принятые в регистры приемника, программно-доступны через РД. Взаимодействие элементов УСАПП происходит через внутреннюю шину под управлением устройства управления (УУ), связанного с ШУ(6).

В МП системе возможны два типа операций над элементами УСАПП: чтение (ввод) в МП содержимого адресуемого элемента и запись (вывод) из МП байта данных в адресуемый элемент УСАПП. Эти операции выполняются программно двумя командами МП: IN B2 и OUT B2, где B2 — системный адрес конкретного регистра УСАПП (см. прил. 1). В процессе выполнения указанных команд в МП системе формируются сигналы управления, комбинация которых определяет ту или иную операцию над элементами УСАПП (см. табл. П.14, в которой под регистром управления (РУ) подразумевается один из регистров РР, РК, РСС1 или РСС2. Доступность того или иного РУ определяется порядком загрузки в УСАПП управляющих слов (УС), определяющих режим работы, форматы и скорости передачи данных).

Для организации связи между МП системой и удаленным устройством используются обычно два УСАПП: один — в системе, другой — в устройстве (здесь вместо УСАПП может применяться любая схема с аналогичными функциями). При этом УСАПП позволяет строить

Табл. П.14. Сигналы управления в операции УСАПП

Тип операции	Операция	Сигналы управления			
		У/Д	ЧТ	ЗП	ВМ
Чтение (Ввод)	ШД ← РД	0	0	1	0
	ШД ← РС	1	0	1	0
Запись (Вывод)	ШД → РПД	0	1	0	0
	ШД → РУ	1	1	0	0
Отключение от ШД		—	—	—	1
Запрещено		—	0	0	0

*дуплексные* (прием и передача возможны одновременно в двух направлениях по двум каналам), *полудуплексные* (прием и передача возможны по одному каналу связи с разделением времени) и *симплексные*

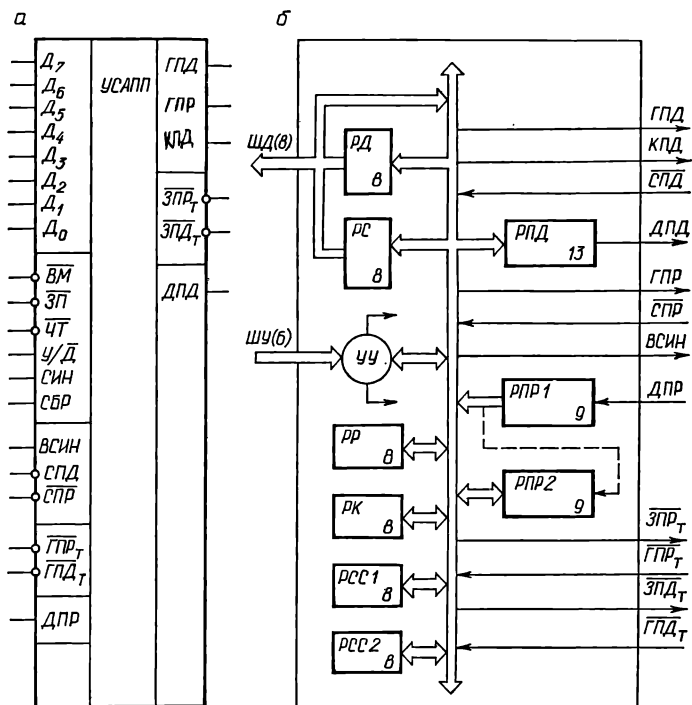


Рис. П.3. Микросхема УСАПП КР580ВВ51:

а — условное обозначение; б — программная модель

(по одному каналу связи идет либо только прием, либо только передача) системы связи. Для каждой из этих систем обмен данными возможен в одном из двух режимов работы УСАПП: асинхронном либо синхронном. Единицей обмена в последовательном формате в обоих режимах обычно является символ, содержащий от 5 до 8 бит данных и представленный в одной из стандартных систем кодирования, например 5-битным телеграфным или 7-битным кодом для обмена информацией (КОИ-7), что не исключает и побайтного обмена некодированными данными.

В асинхронном режиме используется *стартстопный метод передачи* символов, при котором каждый передаваемый символ обрамляется двумя управляющими сигналами: *старт-битом*, всегда начинающим передачу символа, и *стоп-битом*, оканчивающим эту передачу (рис. П.4, а). Управляющие биты в отличие от бит данных имеют всегда строго фиксированные логические значения: 0 — для старт-бита и 1 — для стоп-бита. При отсутствии передачи символа непрерывно передается стоп-бит, поэтому переход 1→0 является для принимающего УСАПП признаком начала передачи очередного символа и используется для отсчета фиксированных временных интервалов с целью определения моментов прихода (точек опроса) последующих бит символа.

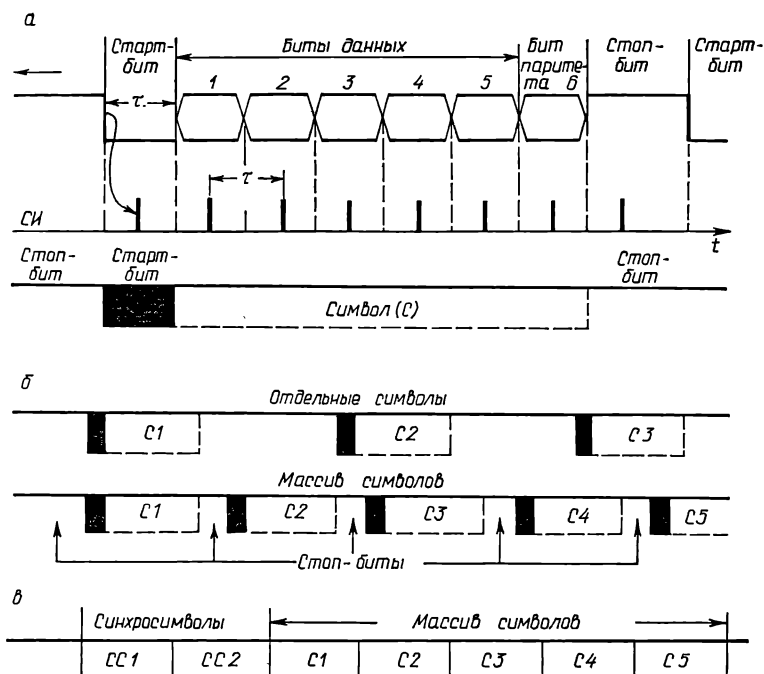


Рис. П.4. Диаграммы последовательной передачи данных:

а — стартовая передача отдельного символа; б — стартовая передача последовательности символов; в — синхронная передача последовательности символов

Адаптеры на разных концах линии связи не имеют, как правило, единого источника или общей линии синхронизации, но должны точно отмеривать период между битами последовательности. Это достигается за счет использования приемником и передатчиком удаленных адаптеров двух отдельных несинхронизированных, но калиброванных и настроенных на одинаковую частоту генераторов СИ. Принимающий УСАПП синхронизируется старт-битом таким образом, чтобы стробирующие СИ, используемые для выделения бит символа, приходились на середину интервала каждого принимаемого бита. Если длительность старт-бита и бита данных равна  $\tau$ , то первый СИ должен появиться через период времени  $\tau/2$  относительно фронта старт-бита, а последующие СИ — через период времени  $\tau$ .

В асинхронном режиме генератор синхронизируется по старт-биту только один раз за время приема символа и не подстраивается в процессе приема бит данных. Вследствие нестабильности частот генераторов передающего и принимающего УСАПП и погрешности синхронизации по старт-биту точка опроса может смещаться от бита к биту относительно идеального центрального положения, что приводит к ошибочному приему бит данных. Ошибка такого типа — ошибка формата, или кадра,—

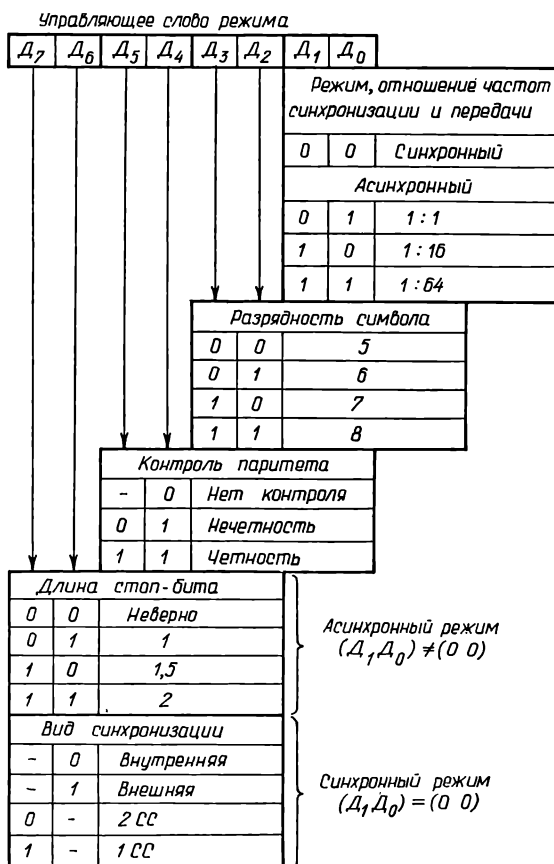


Рис. П.5. Структура управляющего слова режима УСАПП

обнаруживается по стоп-биту: в этом случае на месте стоп-бита вместо 1 фиксируется 0. В УСАПП длительность стоп-бита программируется по величине относительно длительности бита данных с кратностью 1; 1,5 и 2, что позволяет учесть особенности работы различных ПУ. Для повышения достоверности приема в условиях помех передаваемый символ часто дополняют контрольным битом паритета (четного или нечетного). Если принятый бит паритета не соответствует контрольному, сформированному в УСАПП в соответствии с принятыми данными, фиксируется *ошибка паритета*.

Асинхронный режим обмена используется, как правило, для удаленных ПУ с ручным вводом информации, когда передача идет отдельными символами, между которыми имеются длительные стоповые проме-

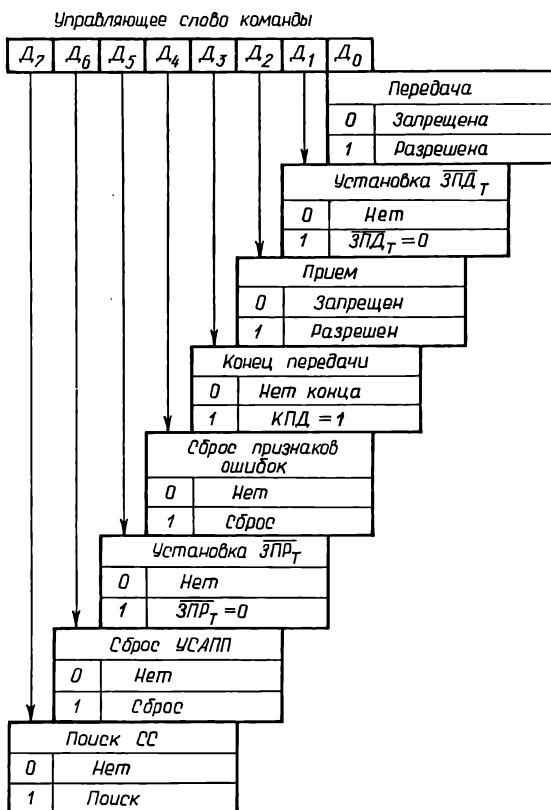


Рис. П.6. Структура управляющего слова команды УСАПП

жутки, или при низком уровне помех в каналах связи при передаче массивов символов (рис. П. 4, б). При передаче массивов эффективная скорость передачи данных существенно снижается из-за временных затрат на передачу управляющих бит для каждого символа. Эта скорость ограничена для УСАПП в асинхронном режиме величиной  $9600 \text{ с}^{-1}$  (Бод). Для помехоустойчивой передачи массивов символов при высоких требованиях к скорости передачи используется синхронный режим УСАПП, позволяющий достичь скорости  $56000 \text{ бит/с}$ .

В синхронном режиме (*синхронный метод передачи*) данные передаются в виде сплошных массивов символов (без управляющих бит между символами), и индивидуальные символы определяются их положением на временной шкале относительно начала массива (рис. П.4, в). Для указания приемнику начала массива используются один или два специальных символа — *синхросимволы* СС1 и СС2 (два синхро-

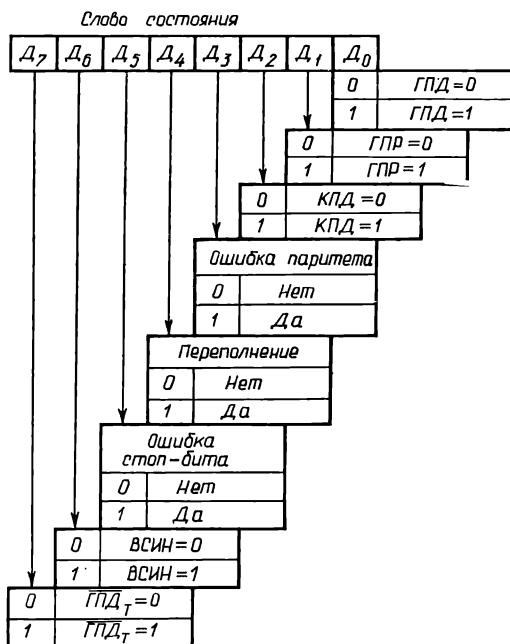


Рис. П.7. Структура слова состояния УСАПП

символа необходимы для обеспечения кодопрозрачности передаваемых данных [11, 76]). После приема синхросимволов приемник интерпретирует последующие символы как данные.

В синхронном режиме передаваемая последовательность символов имеет обычно большую длину, поэтому метод однократной синхронизации по началу массива, аналогичный методу однократной синхронизации по началу символа при стартстопной передаче, здесь не обеспечивает устойчивой синхронизации. В синхронном режиме необходимо иметь либо общую шину синхронизации между удаленными УСАПП (что делается крайне редко), либо отдельные генераторы СИ, которые синхронизируются непрерывно (подстраиваются) потоком символов и синхросимволов. В режиме синхронной передачи при отсутствии, хотя бы временно, данных от МП, УСАПП автоматически выдает в линию синхросимволы, что позволяет подстраивать синхронизацию и «держать ее наготове» для приема очередного символа данных. На приемном конце синхросимволы автоматически удаляются из потока данных. При работе передатчика на несколько приемников синхросимволы, помимо своей основной функции определения начала массива и синхронизации, могут выполнять и адресную функцию: определять своим кодом адрес приемника (приемник содержит аналогичный синхросимвол).

Программирование режимов работы УСАПП осуществляется операцией записи *управляющего слова режима* (УСР) в регистр режима

(РР). Два младших разряда этого слова определяют тип режима (синхронный или асинхронный), а для асинхронного режима позволяют указать, кроме того, коэффициент отношения частоты синхронизации приема-передачи данных к скорости передачи (рис. П.5). Этот коэффициент позволяет более гибко приспособляться к возможностям генераторов СИ. Заметим, что стартстопный асинхронный режим работы возможен только при коэффициентах деления частоты 1:16 и 1:64 (коэффициент 1:1 необходим для обеспечения стартстопной работы по синхронным каналам). Следующие два разряда УСР определяют разрядность символа в зависимости от используемой системы кодирования, а очередные два разряда — возможности контроля паритета. Два старших разряда УСР в зависимости от режима работы, определенного младшими разрядами, указывают на используемую длину стоп-бита для асинхронного режима либо на вид синхронизации для синхронного режима. В случае синхронного режима с внутренней синхронизацией начало массива символов определяется синхросимволами, а для режима с внешней синхронизацией — внешним сигналом синхронизации начала массива, подаваемым на вход ВСИН. Старший разряд УСР определяет для синхронного режима количество используемых синхросимволов.

Непосредственное управление процессом приема-передачи осуществляется *управляющим словом команды* (УСК), которое записывается в регистр команды (РК). Младший разряд этого слова определяет состояние передачи, следующий разряд служит для установок сигнала ЗПД, запроса терминала, очередной разряд определяет состояние приема, а следующий — условие конца передачи (рис. П. 6). Разряд D<sub>4</sub> используется для сброса признаков ошибок, следующий разряд — для установки сигнала ЗПР, запроса терминала, очередной разряд — для программного сброса УСАПП, а старший разряд УСК определяет режим поиска синхросимволов в синхронном режиме. Этот режим предпочтует режиму приема символов данных.

Программный контроль за состоянием УСАПП возможен посредством слова состояния (рис. П. 7). Три младших разряда слова позволяют определять состояние сигналов ГПД, ГПР, КПД, следующие три разряда — состояния сигналов ошибки: паритета, переполнения и формата. Ошибка переполнения возникает, если МП не успел прочесть принятый символ до запроса на обработку следующего принятого символа. Очередной разряд слова используется для контроля сигнала начала массива, а старший разряд — для контроля сигнала ГПД, запроса терминала.

Порядок записи в УСАПП управляющих слов и байтов данных должен соответствовать такому алгоритму. После сигнала аппаратного сброса УСАПП производится запись УСР. Следующее УС в зависимости от содержимого УСР интерпретируется либо как синхросимвол СС1 (для синхронного режима), либо как команда (для асинхронного режима). При синхронном режиме с использованием двух синхросимволов третье УС также интерпретируется как синхросимвол (СС2), а при асинхронном режиме, в котором второе УС не содержит программного сброса, третье слово интерпретируется как слово данных для передачи. После слова команды (не содержащей программного сброса) в УСАПП может быть записана произвольная последовательность команд или данных. Конкретная последовательность слов режима, команд и данных определяется протоколом связи на физическом уровне [11, 76].



В заключение отметим некоторые различия в условиях работы микросхем КР580ВВ51 и КР580ВВ51А в синхронном режиме: во-первых, КР580ВВ51 после инициализации (загрузки режима, синхросимволов, команды и данных) начинает передачу с данных, а КР580ВВ51А сначала передает синхросимволы, а затем — данные; во-вторых, КР580ВВ51 не выполняет команды отмены поиска СС и на каждый принятый СС выдает сигнал ВСИН, не выставляя сигнала ГПР (кодопрозрачность в этом случае обеспечивается за счет байт-стаффинга [11, 76]), а КР580ВВ51А выполняет указанную команду и после этого выставляет сигнал ГПР на каждый принятый СС как на обычный символ (не генерируя сигнал ВСИН), что обеспечивает кодопрозрачность протокола передачи аппаратными средствами. Существуют и другие тонкие отличия в работе микросхем, проявляющиеся в синхронном режиме.

## ЛИТЕРАТУРА

1. Абрамов С. А. Математические построения и программирование.— М.: Наука, 1978.— 192 с.
2. Алексенко А. Г., Галицын А. А., Иванников А. Д. Проектирование радиоэлектронной аппаратуры на микропроцессорах.— М.: Радио и связь, 1984.— 272 с.
3. Артвик Б. А. Сопряжение микроЭВМ с внешними устройствами.— М.: Машиностроение, 1983.— 352 с.
4. Балашов Е. П., Григорьев В. Л., Петров Г. А. Микро- и мини-ЭВМ.— Л.: Энергоатомиздат, 1984.— 376 с.
5. Балашов Е. П., Пузанков Д. В. Микропроцессоры и микропроцессорные системы.— М.: Радио и связь, 1981.— 328 с.
6. Баумс А. К., Гуртовцев А. Л., Зазнова Н. Е. Микропроцессорные средства.— Рига: Зинатне, 1977.— 236 с.
7. Бахвалов Н. С. Численные методы.— М.: Наука, 1975.— 632 с.
8. Брадис В. М. Четырехзначные математические таблицы.— М.: Просвещение, 1979.— 96 с.
9. Бронштейн И. Н., Семендяев К. А. Справочник по математике для инженеров и учащихся втузов.— М.: Наука, 1981.— 720 с.
10. Вайда Ф., Чакань А. МикроЭВМ.— М.: Энергия, 1980.— 360 с.
11. Вейцман К. Распределенные системы мини- и микроЭВМ.— М.: Финансы и статистика, 1982.— 382 с.
12. Вершинин О. Е. Применение микропроцессоров для автоматизации технологических процессов.— Л.: Энергоатомиздат, 1986.— 208 с.
13. Ворожук А. Н. Основы ЦВМ и программирование.— М.: Наука, 1978.— 464 с.
14. Выгодский М. Я. Справочник по высшей математике.— М.: Наука, 1973.— 872 с.
15. Выгодский М. Я. Справочник по элементарной математике.— М.: Наука, 1979.— 336 с.
16. Гибсон Г., Лю Ю-Ч. Аппаратные и программные средства микроЭВМ.— М.: Финансы и статистика, 1983.— 255 с.
17. Гивоне Д., Россер Р. Микропроцессоры и микрокомпьютеры.— М.: Мир, 1983.— 464 с.
18. Глазов А. Б., Костарев С. А., Суханова Е. В. Эффективные программы умножения для микропроцессора КР580ИК80А//Микропроцессорные средства и системы.— 1986.— № 5.— С. 43 — 44.

19. Горбунов В. Л., Пánфилов Д. И., Преснухин Д. Л. Микропроцессоры. Основы построения микроЭВМ.— М.: Высш. шк., 1984.— 144 с.
20. Григорьев В. Л. Программное обеспечение микропроцессорных систем.— М.: Энергоатомиздат, 1983.— 208 с.
21. Гудрин Д. Л. Очень эффективная программа умножения и деления для микропроцессора 8080// Электроника.— 1982.— № 4.— С. 74—75.
22. Гуртовцев А. Л., Пурчик М. Е., Дреннов А. Н. Сопряжение микроЭВМ «Электроника К1-10» с дисплеем РИН-609// Приборы и техника эксперимента.— 1983.— № 5.— С. 77—79.
23. Гуртовцев А. Л., Гурчик М. Е., Сабалаяускас А. И. Микропроцессорная информационно-измерительная система учета и контроля энергии ИИСЭЗ// Приборы и системы управления.— 1988.— № 1.— С. 29—31.
24. Гуртовцев А. Л., Мельников Б. С., Горелик Д. Г. Погрешности накопления измерительной информации в системах учета и контроля энергии// Измерительная техника.— 1984.— № 12.— С. 5—6.
25. Гусак Г. М., Гусак Е. А. Функции и пределы.— Мн. Выш. шк., 1987.— 207 с.
26. Дамке М. Операционные системы микроЭВМ.— М.: Финансы и статистика, 1985.— 150 с.
27. Данилов Ю. А. Многочлены Чебышева.— Мн. Выш. шк., 1984.— 157 с.
28. Дельтон В. Ф. Систематизированный подход к разработке микрокомпьютерных программных средств// Электроника.— 1978.— № 2.— С. 23—30.
29. Ершов А. П. Вызов программистам// Микропроцессорные средства и системы.— 1986.— № 5.— С. 2.
30. Каган Б. М. Электронные вычислительные машины и системы.— М.: Энергия, 1979.— 528 с.
31. Каган Б. М., Сташин В. В. Основы проектирования микропроцессорных устройств автоматики.— М.: Энергоатомиздат, 1987.— 304 с.
32. Карцев М. А. Арифметика цифровых машин.— М.: Наука, 1969.— 576 с.
33. Карцев М. А., Брик В. А. Вычислительные системы и синхронная арифметика.— М.: Радио и связь, 1981.— 360 с.
34. Квитнер П. Задачи, программы, вычисления, результаты.— М.: Мир, 1980.— 424 с.
35. Кейслер С. Проектирование операционных систем для малых ЭВМ.— М.: Мир, 1986.— 680 с.
36. Клингман Э. Проектирование микропроцессорных систем.— М.: Мир, 1980.— 576 с.
37. Кнут Д. Искусство программирования для ЭВМ: В 3 т.— М.: Мир, 1977.— Т. 2.— 726 с.
38. Компьютеры: Справочное руководство: В 3 т.— М.: Мир, 1986.— Т. 3.— 403 с.
39. Контроллер программируемый универсальный «Электроника МС2702»: Программное обеспечение.— Киев: Минэлектронпром, 1980.— 54 с.
40. Костин А. Е., Шаньгин В. Ф. Организация и обработка структур данных в вычислительных системах.— М.: Высш. шк., 1987.— 248 с.
41. Коффрон Дж. Технические средства микропроцессорных систем: Практический курс.— М.: Мир, 1983.— 344 с.

42. *Коффрон Дж., Лонг В.* Расширение микропроцессорных систем.— М.: Машиностроение, 1987.— 320 с.
43. *Левенталь Л.* Введение в микропроцессоры.— М.: Энергоатомиздат, 1983.— 464 с.
44. *Левенталь Л., Сэйвилл У.* Программирование на языке ассемблера для микропроцессоров 8080 и 8085.— М.: Радио и связь, 1987.— 448 с.
45. *Лихолетов И. И.* Высшая математика, теория вероятностей и математическая статистика.— Мн.: Выш. шк., 1976.— 720 с.
46. *Лысков Б. Г.* Арифметические и логические основы цифровых автоматов.— Мн.: Выш. шк., 1980.— 336 с.
47. *Любимский Э. З., Мартынюк В. В., Трифонов Н. П.* Программирование.— М.: Наука, 1980.— 608 с.
48. *Люстерник Л. А., Червоненкис О. А., Ямпольский А. Р.* Справочная математическая библиотека: Математический анализ: Вычисление элементарных функций.— М.: Физматгиз, 1963.— 248 с.
49. *Макгуайр Д.* Программа для микроконтроллера 8048, осуществляющая деление 16-разрядного числа на 8-разрядное // Электроника.— 1983.— № 10.— С. 62—64.
50. *Мак-Кракен Д., Дорн У.* Численные методы и программирование на ФОРТРАНе.— М.: Мир, 1977.— 326 с.
51. Математические алгоритмы и программы для малых ЭВМ/ Г. Л. Литвинов, Р. И. Толстых, Л. И. Шадурская и др.— М.: Финансы и статистика, 1981.— 112 с.
52. *Морисуэ М., Есикава Т.* МикроЭВМ за три дня.— М.: Мир, 1981.— 184 с.
53. *Поснов Н. Н.* Арифметика вычислительных машин в упражнениях и задачах: Системы счисления, коды.— Мн.: Изд-во «Университетское», 1984.— 221 с.
54. *Поспелов Д. А.* Арифметические основы вычислительных машин дискретного действия.— М.: Высш. шк., 1970.— 308 с.
55. Программирование микропроцессоров/ В. Фрибель, Х. Ролоф, Х. Шиллер и др.— М.: Энергоиздат, 1982.— 88 с.
56. Публикация МЭК 559-82: Двоичные арифметические операции с плавающей запятой для систем микроустройств обработки данных.— М.: Изд-во стандартов, 1982.— 28 с.
57. Расширение возможностей микроЭВМ «Электроника К1-10»/ С. В. Гудыменко, А. Л. Гуртовцев, М. Е. Гурчик и др.// Электронная техника. Сер. Экономика и системы управления.— 1985.— Вып. 2.— С. 46—51.
58. *Рыбасенко В. Д., Рыбасенко И. Д.* Элементарные функции: Формулы, таблицы, графики.— М.: Наука, 1987.— 416 с.
59. *Сима М., Феггин Ф.* Быстродействующий однокристалльный *n*-канальный процессор// Электроника.— 1974.— № 8.— С. 44—51.
60. *Соботка З., Стары Я.* Микропроцессорные системы.— М.: Энергоиздат, 1981.— 496 с.
61. *Соловьев Г. Н.* Арифметические устройства ЭВМ.— М.: Энергия, 1978.— 176 с.
62. Сопряжение микроЭВМ «Электроника К1-10» с алфавитно-цифровым печатающим устройством «DARO-1156»/ С. В. Гудыменко, А. Л. Гуртовцев, М. Е. Гурчик и др.// Приборы и техника эксперимента.— 1984.— № 3.— С. 85—87.
63. *Соучек Б.* Микропроцессоры и микроЭВМ.— М., Сов. радио, 1979.— 520 с.

64. Справочник по цифровой вычислительной технике/ Б. Н. Малиновский, В. Я. Александров, В. П. Боюн и др.; Под ред. Б. Н. Малиновского.— Киев: Техника, 1974.— 512 с.
65. Стоун Г. С., Сиворек Д. П. Введение в организацию ЭВМ и структуры данных.— М.: Машиностроение, 1980.— 320 с.
66. Таненбаум Э. Многоуровневая организация ЭВМ.— М.: Мир, 1979.— 550 с.
67. Турчак Л. И. Основы численных методов.— М.: Наука, 1987.— 320 с.
68. Уокерли Д. Архитектура и программирование микроЭВМ: В 2 кн.— М.: Мир, 1984.— Кн. I.— 486 с.
69. Фаулджер Р. Программирование встроенных микропроцессоров.— М.: Мир, 1985.— 275 с.
70. Фильчаков П. Ф. Справочник по высшей математике.— Киев: Наукова думка, 1973.— 744 с.
71. Фихтер А. Программа микрокомпьютера 8080 для деления 32-разрядных чисел на 16-разрядные// Электроника.— 1980.— № 7.— С. 78—79.
72. Форсайт Дж., Малькольм М., Моулер К. Машинные методы математических вычислений.— М.: Мир, 1980.— 280 с.
73. Фридмен М., Ивнс Л. Проектирование систем с микрокомпьютерами.— М.: Мир, 1986.— 405 с.
74. Хемминг Р. В. Численные методы.— М.: Наука, 1972.— 400 с.
75. Хилбурн Д., Джулич П. МикроЭВМ и микропроцессоры.— М.: Мир, 1979.— 464 с.
76. Якубайтис Э. А. Архитектура вычислительных сетей.— М.: Статистика, 1980.— 279 с.

# ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

- Адаптер периферийный параллельный (ППА) 303, 326  
Адрес 303  
Адресация базовая косвенная 216  
— — непосредственная 305  
— прямая 305  
— регистровая (неявная) 305  
— — косвенная 305  
— стековая 305  
Аккумулятор (А) 302  
Алгоритм Бута 42  
Антипереполнение порядка произведения 133  
— — суммы 114  
— — частного 138  
— формата 19  
Арифметика ограниченной точности (АОТ) 19  
— с плавающей запятой 25, 105  
— с фиксированной запятой 23, 24  
— целых чисел 21  
Архитектура микропроцессора 300  
— микропроцессорной (МП) системы 300
- Байт 26, 303  
— младший (МЛБ) 26  
— средний (СРБ) 26  
— старший (СТБ) 26  
Бит (флаг) 302, 303  
— знака (S) 302  
— нуля (Z) 302  
— паритета (P) 302  
— переноса (CY) 27, 302  
— — вспомогательного (AC) 302
- Вес разряда двоичного 14  
— — позиционный 12  
Видеотерминал (дисплей) 246  
Время выполнения программы 29  
Выборка команды 304  
— операнда 304  
Выравнивание порядков 113  
Выражения языка ассемблера 324  
Вычитание чисел двоично-десятичных 33  
— — двоичных беззнаковых 27, 29
- Граница числа 17
- Дезассемблирование 305  
Дек 214  
Деление без восстановления остатка 80  
— двоичное 78  
— с восстановлением остатка 79  
— с остатком 77  
— чисел двоичных дробных 77—102  
— — — многобайтных 32  
— — — с плавающей запятой 137  
— — — целых 95  
— — десятичных 102—104  
Денормализация мантиссы 113  
Диапазон чисел 19  
— — нормализованных 107  
Директивы монитора 273  
Диспетчер директив 274, 294—298  
Длина программы 29
- Длительность командного цикла 304  
Драйвер 245
- Заем 17  
Закон ассоциативный 19  
— дистрибутивный 19  
Запись 212  
Знак числа 14  
Значность числа 23
- Имя 324  
— символическое 324  
— — определяемое пользователем 324  
— — постоянное 324  
Инструкция команды 255  
— режима 255  
Интервал сходимости 179  
Интерфейс радиальный параллельный (ИРПР) 246  
— — последовательный (ИРПС) 246  
Карта памяти 303  
Ключ 212  
Код двоично-десятичный 16  
— дополнительный 17  
— — двоичный 30  
— — десятичный 33  
— — модифицированный 121  
— международный телеграфный (МТК-2) 255—264  
— обмена информацией семибитный (КОИ-7) 255  
— обратный 17  
— прямой 17  
Команда вызова подпрограмм 29  
— двухадресная 304  
— двухбайтная 305  
— двухбайтного сложения 43  
— десятичной коррекции 32  
— интерфейсная 214  
— одноадресная 304  
— однобайтная 305  
— трехбайтная 305  
Командный цикл 304  
Команды операций арифметических 306  
— — логических 306  
— передачи данных 305  
— — управления 306  
— рестарта 317  
— стека, ввода-вывода и управления 317  
Компенсация ошибок 23  
Консоль (пульт оператора) 255, 273
- Макрокоманда (макровызов) 326  
Макроопределение 326  
Макросредства 325  
Мантисса 106  
— ненормализованная 106  
— нормализованная 106  
Массив многомерный 212  
— одномерный 212  
Машина абстрактная 245  
Метка 324  
Метод передачи символов синхронный 341  
— — — стартовый 339  
Микропроцессор (МП) 300  
Модель МП 300  
— МП системы программная 300  
— памяти 303

- портов ввода-вывода 303
- Монитор 246, 273, 274
- Накопитель на гибких магнитных дисках (НГМД)** 246
- Норма равномерная 178
  - функции 178
- Нормализация влево 114
  - вправо 113
- Нуль машинный 19
  - *n*-кратный 185
- Область монитора рабочая** 274
- Обмен асинхронный (условный) 333
  - дуплексный 246
  - синхронный (безусловный) 333
- Округление чисел 21
  - несимметричное (отбрасывание) 21
  - симметричное 21
- Оператор языка ассемблера 319
- Операционная система 245
- Отсутствие переполнения 113
- Очередь 214
- Ошибка абсолютная суммы 25
  - антипереполнения 19
  - граничная абсолютная разности 25
    - округления 22
    - — относительная деления 77
    - — — произведения 35
    - — — разности 26
    - — — суммы 25
  - метода 78
  - округления абсолютная 22
    - — — мантиссы суммы 115
    - — — относительная 22
    - — — деления 77, 138
    - — — произведения 35, 133
  - паритета 340
  - переполнения 19
  - формата (кадра) 340
- Память оперативная** 303
- Парабола *n*-го порядка 185
- Пара регистровая 300
- Перенос 17
  - двоично-десятичный 32
- Переполнение мантиссы суммы 113
  - — частного 138
  - — порядка произведения 133
  - — суммы 114
  - — частного 138
  - — формата числа 19
  - — частного 81
- Поле 212
  - комментария 324
  - названия 324
  - операндов 324
  - операции 324
- Полином 12
  - Тейлора 180
- Полубайт 303
- Порт ввода-вывода 265
- Порядок операций 20
  - целочисленный 106
  - — несмещенный 109
  - — смещенный 109
- Правила округления чисел 21
- Правило Гаусса 22
- Предложения языка ассемблера 319
- Представление нуля 17
- Премопередатчик универсальный синхронно-асинхронный (УСАПП) 253
- Признак Лейбница 180
  - переполнения 29
- Произведение в сокращенном формате 51
  - промежуточное 55
  - точное 35
  - частичное (ЧП) 36
- Пространство памяти адресное 303
  - — рабочее (емкость) 303
- Псевдокоманда 325
  - определения данных 325
  - — имен 325
  - — макрокоманд 325
  - управления трансляцией 325
- Псевдопроизведение 42
- Радиус сходимости** 179
- Разряд знаковый (знака) 14
  - цифровой 14
- Разрядность 12
  - мантиссы 107
  - порядка 107
- Распаковка 74
- Регистр общего назначения (РОН) 300
  - признаков (флагов) (F) 302
  - разрешения прерываний (триггер) (INTE) 302
  - счетчика команд (программный счетчик) (PC) 302
  - указателя стека (SP) 302
  - управляющего слова (PUC) 251
  - управляющий 256
  - латинский (ЛАТ) 256
  - русский (РУС) 256
  - цифровой (ЦИФ) 256
- Реентерабельность подпрограммы 214
- Ряд степенной 179
  - Тейлора 176, 179, 202, 206
- Символ управляющий 256
  - — «Возврат каретки» (BK) 256
  - — «Перевод строки» (PC) 256
  - — «Пробел» 256
- Синхросимволы 341
- Система микропроцессорная (МП система) 8, 242, 243
  - операционная 245
  - реального времени 214
  - связи дуплексная 339
  - — полудуплексная 339
  - — симплексная 339
  - счисления позиционная 12
    - — двоичная 12
    - — — двоично-десятичная 16
    - — — десятичная 12
    - — — шестнадцатеричная 12
    - — смешанная 15
- Слово многобайтное 304
  - состояния процессора (ССП) 302
  - управляющее 327
  - — команды (инструкция команды) 344
  - — режима (УСР) (инструкция режима) 344
- Сложение чисел в дополнительном коде 30, 31
  - — двоично-десятичных 32
  - — двоичных беззнаковых 27
  - — — знаковых 30
  - — — с плавающей запятой 112
- Список 325
  - двухсвязный 213
  - многосвязный (сетовой) 213
  - односвязный 212

- параметров фактических 326
- — формальных 326
- связный 212
- Старт-бит 339
- Статус ввода отрицательный 251
- — положительный 251
- Стек 214
- Стоп-бит 339
- Структурирование данных 211
- Структуры данных 211
- динамические 212
- полустатические 213
- статические 212
- Сумма двоично-десятичная 32
- частичных произведений (СЧП) 36
- Схема вычислительная 26, 38
- — деления 78
- — сложения 26
- — умножения 38
- Горнера 37, 38
- Сходимость ряда 180

Таблица 212

Телетайп 246

Терм 324

- самоопределенный 324
- — двоичный 324
- — восьмеричный 324
- — десятичный 324
- — символьный 324
- — шестнадцатеричный 324

Терминатор 280

Тип данных 211

Точка перегиба графика 185

#### Умножение чисел двоичных беззнаковых 36

- — — знаковых 41
- — — многобайтных 32
- — десятичных 74—77
- — дробных знаковых 62—74
- — по вычислительной схеме 144
- — по вычислительной схеме 248
- — по вычислительной схеме 346
- — путем накопления 43
- — с плавающей запятой 132, 133
- — с разрядов множителя младших 37
- — — старших 38

#### Устройство арифметическо-логическое (АЛУ) 302

- внешнее (ВУ), периферийное (ПУ) 245
- запоминающее оперативное (ОЗУ) 303
- — постоянное (ПЗУ) 303
- — — перепрограммируемое (ППЗУ) 303
- — сверхоперативное (СОЗУ) 300
- — с произвольной выборкой (ЗУПВ) 303
- печатающее 245
- сопряжения (УС) 249

#### Факториал 177, 197—201

Формат данных 304

- команды (командное слово) 305

#### Формула Герона 192

— Ньютона 192

#### Функция алгебраическая 177

- аппроксимирующая 178
- гиперболическая 177, 209, 210
- дробно-линейная 177
- дробно-рациональная 177
- — нелинейная 177

- извлечения корня 177
- иррациональная 177
- квадратичная 176
- косинуса 201
- котангенса 201
- — линейная 176
- логарифмическая 177
- неэлементарная 177
- обратной пропорциональности 177
- показательная 177
- постоянная 176
- рациональная 176
- синуса 201
- сложная 177
- степенная 176
- — с рациональным показателем 177
- тангенса 201
- трансцендентная 177
- тригонометрическая (круговая) 177, 201—209
- экспоненциальная (экспонента) 199
- элементарная 177

#### Характеристика числа 12

Характеристики точностные 22

#### Цифра верная 24

- значащая 24
- младшая (МЛЦ) 150
- старшая (СТЦ) 150
- *R*-ичная (*R*-ичной системы счисления) 12

#### Частичное произведение (ЧП) 36

Часть числа дробная 12

- — знаковая 17
- — целая 12
- — цифровая 17

#### Число беззнаковое (без знака) 14

- — дробное 14
- — целое 14
- — двоичное целое со знаком 14
- — дробное (правильная дробь) 13
- — — двоичное беззнаковое 13
- — — со знаком 13, 95—98
- — знаковое 17
- — максимальное 14
- — минимальное 14
- — многобайтное 26
- — ненормализованное 106
- — нормализованное 106
- — ограниченной точности 19
- — округленное 21
- — округляемое 21
- — с запятой плавающей 12, 105
- — фиксированной 13
- — с «избытком» 23
- — смешанное (неправильная дробь) 13
- — с «недостатком» 23
- — целое 13
- — двоичное беззнаковое 14
- — — со знаком 14

#### Шина адреса (ША) 303

- данных внешняя (ШД) 302
- — внутренняя (ВШД) 302
- управления (ШУ) 303

#### «Эхо» 276

#### Язык ассемблера 318

- директив 246
- макроассемблера 318
- Ячейка памяти 303

# ОГЛАВЛЕНИЕ

Предисловие . . . . .	3
Условные сокращения . . . . .	6
Методологические замечания . . . . .	8
<b>1. Программы арифметики с фиксированной запятой</b>	
1.1. Общие сведения . . . . .	12
1.2. Сложение и вычитание $N$ -байтных чисел . . . . .	25
1.2.1. Методика сложения и вычитания . . . . .	25
1.2.2. Двоичные числа . . . . .	28
1.2.3. Десятичные числа . . . . .	32
1.3. Умножение двоичных чисел . . . . .	35
1.3.1. Методика умножения . . . . .	35
1.3.2. Целые беззнаковые числа . . . . .	43
1.3.2.1. Формат $8 \cdot 8 = 16$ . . . . .	43
1.3.2.2. Форматы $8 \cdot 16 = 24$ , $16 \cdot 16 = 16$ . . . . .	49
1.3.2.3. Формат $16 \cdot 16 = 32$ . . . . .	52
1.3.3. Целые числа со знаком . . . . .	56
1.3.3.1. Формат $8 \cdot 8 = 16$ . . . . .	56
1.3.3.2. Формат $8 \cdot 16 = 24$ . . . . .	57
1.3.3.3. Формат $16 \cdot 16 = 32$ . . . . .	59
1.3.3.4. Формат $8 \cdot 24 = 32$ . . . . .	60
1.3.4. Дробные числа со знаком . . . . .	62
1.3.4.1. Умножение дробных чисел . . . . .	62
1.3.4.2. Формат $16 \cdot 16 = 16$ . . . . .	63
1.3.4.3. Формат $17 \cdot 17 = 17$ . . . . .	66
1.3.4.4. Формат $24 \cdot 24 = 24$ . . . . .	68
1.3.4.5. Формат $16 \cdot 16 = 32$ . . . . .	72
1.4. Умножение десятичных чисел . . . . .	74
1.5. Деление двоичных чисел . . . . .	77
1.5.1. Методика деления . . . . .	77
1.5.2. Целые беззнаковые числа . . . . .	82
1.5.2.1. Формат $16 : 8 = (8,8)$ . . . . .	82
1.5.2.2. Формат $16 : 8 = (16,8)$ . . . . .	85
1.5.2.3. Формат $24 : 16 = (8,16)$ . . . . .	87
1.5.2.4. Формат $32 : 16 = (16,16)$ . . . . .	89
1.5.2.5. Формат $16 : 16 = (16,16)$ . . . . .	92
1.5.3. Целые числа со знаком . . . . .	95
1.5.4. Дробные числа со знаком . . . . .	98
1.5.4.1. Деление дробных чисел . . . . .	98
1.5.4.2. Формат $16 : 16 = 16$ . . . . .	99
1.5.4.3. Формат $17 : 17 = 17$ . . . . .	100
1.6. Деление десятичных чисел . . . . .	102
<b>2. Программы арифметики с плавающей запятой</b>	
2.1. Общие сведения . . . . .	105
2.2. Сложение и вычитание двоичных чисел . . . . .	112
2.2.1. Методика сложения и вычитания . . . . .	112
2.2.2. Формат $(8,16) + (8,16) = (8,16)$ . . . . .	116
2.2.3. Формат $(8,24) + (8,24) = (8,24)$ . . . . .	126
2.3. Умножение двоичных чисел . . . . .	132
2.3.1. Методика умножения . . . . .	132
2.3.2. Формат $(8,16) \cdot (8,16) = (8,16)$ . . . . .	133
2.3.3. Формат $(8,24) \cdot (8,24) = (8,24)$ . . . . .	136
2.4. Деление двоичных чисел . . . . .	137
2.5. Умножение целого числа на число с плавающей запятой . . . . .	140
<b>3. Программы преобразования представлений чисел</b>	
3.1. Общие сведения . . . . .	145
3.2. Преобразования целых десятичных чисел в двоичные . . . . .	149
3.3. Преобразования целых двоичных чисел в десятичные . . . . .	151
3.4. Преобразования дробных десятичных чисел в двоичные . . . . .	155
3.5. Преобразования дробных двоичных чисел в десятичные . . . . .	157
3.6. Преобразования десятичных чисел с плавающей запятой в двоичные числа с плавающей запятой . . . . .	159
3.7. Преобразования двоичных чисел с плавающей запятой в десятичные числа с плавающей запятой . . . . .	167



#### 4. Программы вычисления элементарных функций

4.1. Общие сведения . . . . .	176
4.2. Обратная пропорциональность . . . . .	183
4.3. Степенная функция . . . . .	185
4.4. Полином . . . . .	189
4.5. Квадратный корень . . . . .	192
4.6. Факториал . . . . .	197
4.7. Показательная функция . . . . .	199
4.8. Тригонометрические функции . . . . .	201
4.9. Гиперболические функции . . . . .	209

#### 5. Программы обработки структур данных

5.1. Общие сведения . . . . .	211
5.2. Формирование массивов . . . . .	217
5.2.1. Простое формирование массива . . . . .	217
5.2.2. Формирование массива с контролем кодов окончания и забоя . . . . .	218
5.2.3. Формирование массива с контролем произвольных управляющих кодов . . . . .	220
5.3. Копирование массивов . . . . .	222
5.3.1. Пересылка информации с конца массива . . . . .	222
5.3.2. Пересылка информации с начала массива . . . . .	224
5.4. Поиск в структурах . . . . .	225
5.4.1. Задачи поиска . . . . .	225
5.4.2. Простое сравнение массивов . . . . .	225
5.4.3. Поиск однобайтного кода в массиве . . . . .	227
5.4.4. Поиск последовательности кодов в массиве . . . . .	228
5.4.5. Поиск элемента таблицы по ключу . . . . .	230
5.4.6. Поиск элемента списка . . . . .	232
5.5. Преобразования структур . . . . .	233
5.5.1. Задачи преобразования . . . . .	233
5.5.2. Прямая перекодировка строки . . . . .	234
5.5.3. Обратная перекодировка строки . . . . .	235
5.5.4. Удаление фрагмента массива . . . . .	237
5.5.5. Вставка фрагмента в массив . . . . .	238
5.5.6. Вставка элемента в список . . . . .	240
5.5.7. Удаление элемента из списка . . . . .	242

#### 6. Программы системного обеспечения

6.1. Общие сведения . . . . .	245
6.2. Драйвер обмена по ИРПС . . . . .	246
6.3. Драйвер обмена по ИРПС . . . . .	253
6.4. Драйвер обмена с телетайпом . . . . .	255
6.5. Драйвер обмена с дисководом . . . . .	264
6.6. Системный монитор . . . . .	272
6.6.1. Функции и структура . . . . .	272
6.6.2. Вспомогательные программы . . . . .	275
6.6.3. Директива вывода содержимого памяти . . . . .	281
6.6.4. Директива заполнения области памяти . . . . .	283
6.6.5. Директива перемещения содержимого области памяти . . . . .	283
6.6.6. Директива модификации содержимого области памяти . . . . .	284
6.6.7. Директива отображения и модификаций регистров . . . . .	285
6.6.8. Директива запуска программы пользователя . . . . .	289
6.6.9. Обработка точек разрыва . . . . .	291
6.6.10. Директива документирования . . . . .	294
6.6.11. Диспетчер директив . . . . .	294

Заключение . . . . .	299
----------------------	-----

Приложения . . . . .	300
----------------------	-----

1. Микропроцессор серии KP580 и его архитектура (программная модель и набор команд) . . . . .	300
2. Таблицы эквивалентных шестнадцатерично-десятичных значений числовых данных . . . . .	318
3. Язык макроассемблера . . . . .	318
4. Программирование периферийного параллельного адаптера KP580BB55 . . . . .	326
5. Программирование периферийного последовательного адаптера KP580BB51 . . . . .	334

Литература . . . . .	344
----------------------	-----

Предметный указатель . . . . .	348
--------------------------------	-----

15.33.74

**А.Л.ГУРТОВЦЕВ  
С.В.ГУДЫМЕНКО**

# **ПРОГРАММЫ ДЛЯ МИКРОПРОЦЕССОРОВ**