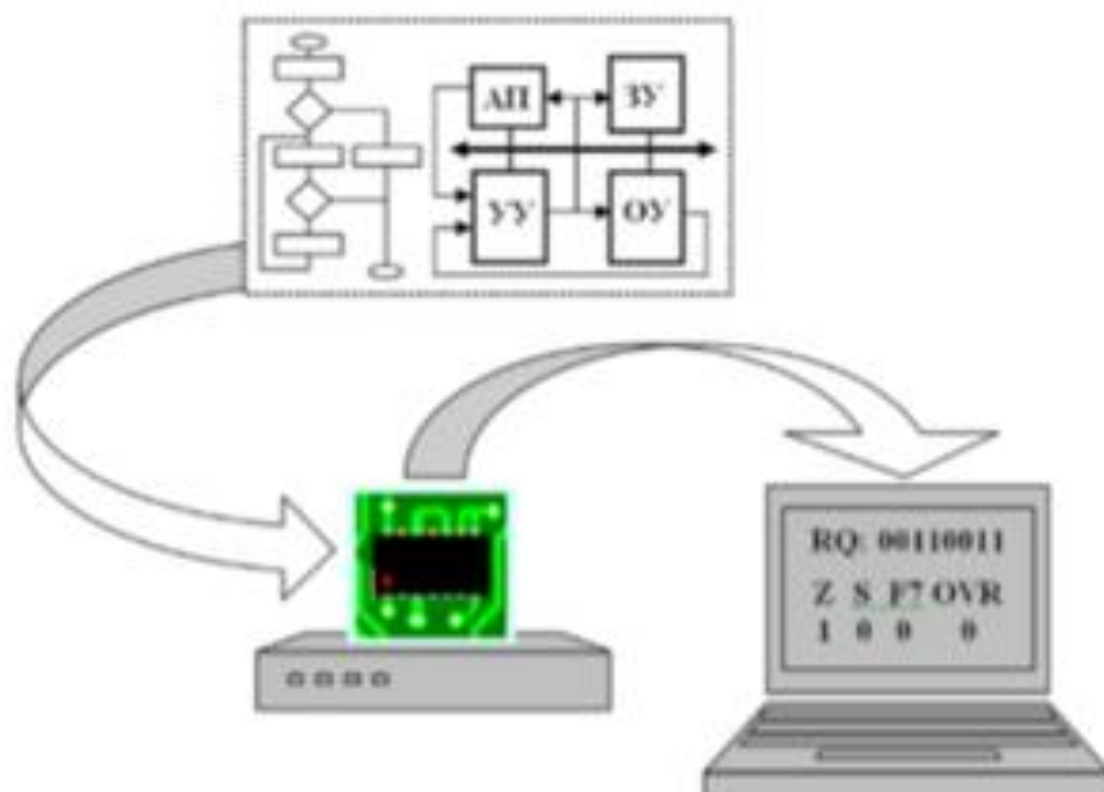




Д. А. СТРАБЫКИН

# ОРГАНИЗАЦИЯ ЭВМ: ЛАБОРАТОРНЫЙ ПРАКТИКУМ НА КОМПЬЮТЕРАХ



Учебное пособие

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных систем  
Факультет автоматики и вычислительной техники  
Кафедра электронных вычислительных машин

Д. А. СТРАБЫКИН

**ОРГАНИЗАЦИЯ ЭВМ:  
ЛАБОРАТОРНЫЙ ПРАКТИКУМ  
НА КОМПЬЮТЕРАХ**

Учебное пособие

4-е издание, переработанное и дополненное

Киров

2020

УДК 004.382.7(07)

С830

Рекомендовано к изданию методическим советом Института математики и информационных систем ВятГУ

Допущено редакционно-издательской комиссией методического совета ВятГУ в качестве учебного пособия для студентов направлений подготовки: 09.03.01 Информатика и вычислительная техника, 09.03.02 Информационные системы и технологии и 09.03.03 Прикладная информатика.

Рецензент

доктор технических наук, профессор кафедры «Вычислительная техника»  
ФГБОУ ВО «Ульяновский государственный технический университет»  
В. Н. Негода

**Страбыкин, Д. А.**

С830 Организация ЭВМ: лабораторный практикум на компьютерах : учеб. пособие – 4-е изд., перераб. и доп. / Д. А. Страбыкин. – Киров : ВятГУ, 2020. – 186 с.

В учебном пособии предлагается цикл взаимосвязанных лабораторных работ: по узлам и блокам, операционным устройствам, устройствам управления, процессорам с микропрограммным и программным уровнем управления, системам адресации и прерывания программ ЭВМ. Практикум предполагает решение упрощенных задач проектирования изучаемых устройств с использованием их микропрограммной реализации на заданных вычислительных структурах. Вычислительные структуры предоставляются разработчику в виде программных моделей, входящих в состав программного обеспечения практикума. Программное обеспечение практикума позволяет вводить данные, микропрограммы, программы и производить отладку разработанных устройств и ЭВМ, поддерживая пошаговый и автоматический режимы. Подготовка к лабораторным работам рассматривается в учебном пособии на примерах конкретных заданий.

УДК 004.382.7(07)

© ВятГУ, 2020

## ОГЛАВЛЕНИЕ

|  |    |
|--|----|
| ПРЕДИСЛОВИЕ.....   | 7  |
| 1. ОРГАНИЗАЦИЯ ПРАКТИКУМА.....   | 9  |
| 1.1. Состав лабораторных работ .....   | 9  |
| 1.2. Лабораторные установки на базе диалоговых систем управления.....                | 11 |
| 1.3. Управление лабораторной установкой .....  | 13 |
| 2. ОПЕРАЦИОННОЕ УСТРОЙСТВО .....   | 17 |
| 2.1. Основные положения.....   | 17 |
| 2.1.1. Три аспекта рассмотрения вычислительного устройства.....                      | 17 |
| 2.1.2. Модель дискретного преобразователя .....                                      | 18 |
| 2.1.3. Микрооперации и логические условия .....                                      | 19 |
| 2.1.4. Организация межрегистровых связей .....                                       | 20 |
| 2.1.5. Базовые структуры операционных устройств.....                                 | 20 |
| 2.2. Структура и микроинструкции .....   | 23 |
| 2.3. Лабораторная работа № 1 .....   | 27 |
| 2.3.1. Задание № 1.....  | 28 |
| 2.3.2. Разработка алгоритма .....  | 28 |
| 2.3.3. Распределение регистров и разработка микропрограммы.....                      | 29 |
| 2.3.4. Кодирование микропрограммы .....  | 30 |
| 2.3.5. Ввод и отладка микропрограммы.....  | 31 |
| 2.4. Контрольные вопросы.....  | 31 |
| 3. УСТРОЙСТВО УПРАВЛЕНИЯ .....   | 33 |
| 3.1. Основные положения.....   | 33 |
| 3.1.1. Два подхода к построению устройств управления.....                            | 33 |
| 3.1.2. Пример устройства управления.....   | 34 |
| 3.1.3. Способы кодирования микроопераций.....  | 35 |
| 3.1.4. Формирование адреса микрокоманды с учетом логических условий.....             | 37 |
| 3.1.5. Структура и рабочий цикл устройств управления.....                            | 38 |
| 3.2. Устройство управления с блоком управления последовательностью микрокоманд ..... | 40 |

|  |    |
|--|----|
| 3.3. Лабораторная работа № 2 .....   | 52 |
| 3.3.1. Задание № 2.....  | 52 |
| 3.3.2. Разработка микропрограммы и распределение памяти<br>микропрограмм.....                    | 53 |
| 3.3.3. Кодирование микропрограммы .....  | 54 |
| 3.3.4. Ввод и отладка микропрограммы .....   | 55 |
| 3.4. Контрольные вопросы.....  | 57 |
| 4. ВЫЧИСЛИТЕЛЬНОЕ УСТРОЙСТВО С ЗАПОМИНАЮЩИМ<br>УСТРОЙСТВОМ.....                                  | 59 |
| 4.1. Основные положения.....   | 59 |
| 4.1.1. Функционально-структурные параметры вычислительного<br>устройства .....                   | 59 |
| 4.1.2. Два вида параллельной обработки информации.....   | 60 |
| 4.1.3. Виды конвейеров.....  | 61 |
| 4.1.4. Конвейерное выполнение микрокоманд.....   | 62 |
| 4.1.5. Эффективность конвейерного выполнения микрокоманд.....                                    | 62 |
| 4.2. Структура устройства и конвейерное выполнение микрокоманд .....                             | 64 |
| 4.3. Лабораторная работа № 3 .....   | 67 |
| 4.3.1. Задание № 3.....  | 68 |
| 4.3.2. Распределение ячеек ЗУ и регистров микропроцессора .....                                  | 68 |
| 4.3.3. Разработка микропрограммы для устройства без конвейерного<br>выполнения микрокоманд ..... | 69 |
| 4.3.4. Разработка микропрограммы для устройства<br>с конвейерным выполнением микрокоманд.....    | 75 |
| 4.3.5. Ввод и отладка микропрограмм.....   | 78 |
| 4.3.6. Сравнение микропрограмм .....   | 82 |
| 4.4. Контрольные вопросы.....  | 85 |
| 5. УЧЕБНАЯ ЭВМ.....  | 87 |
| 5.1. Основные положения.....   | 87 |
| 5.1.1. Понятие архитектуры ЭВМ.....  | 87 |
| 5.1.2. Система команд: набор операций.....   | 88 |

|   |     |
|---|-----|
| 5.1.3. Модификации команд.....  | 90  |
| 5.1.4. Число адресных полей в команде .....   | 91  |
| 5.1.5. Виды программистских структур .....  | 92  |
| 5.1.6. Микропрограммная и программная реализация алгоритмов .....                         | 93  |
| 5.1.7. Иерархия уровней внутренних языков .....   | 95  |
| 5.2. Аппаратные средства для микропрограммной реализации ЭВМ .....                        | 96  |
| 5.3. Лабораторная работа № 4 .....  | 97  |
| 5.3.1. Задание № 4.....   | 98  |
| 5.3.2. Определение архитектуры и программирование.....                                    | 98  |
| 5.3.3. Кодирование программы и распределение памяти программ<br>и данных .....            | 102 |
| 5.3.4. Разработка структуры и алгоритма работы .....                                      | 104 |
| 5.3.5. Микропрограммная реализация ЭВМ.....   | 108 |
| 5.3.6. Ввод и отладка микропрограммы командного цикла и программы<br>решения задачи ..... | 113 |
| 5.3.7. Расчет производительности и быстродействия.....                                    | 114 |
| 5.4. Контрольные вопросы.....   | 117 |
| 6. СИСТЕМА АДРЕСАЦИИ ЭВМ .....  | 120 |
| 6.1. Основные положения.....  | 120 |
| 6.1.1. Способы адресации .....  | 120 |
| 6.1.2. Особенности задания способов адресации.....  | 125 |
| 6.2. Аппаратные средства для микропрограммной реализации ЭВМ .....                        | 127 |
| 6.3. Лабораторная работа № 5 .....  | 128 |
| 6.3.1. Задание № 5.....   | 129 |
| 6.3.2. Определение архитектуры и программирование.....                                    | 130 |
| 6.3.3. Кодирование программы и распределение памяти программ<br>и данных .....            | 136 |
| 6.3.4. Разработка алгоритма работы и микропрограммная<br>реализация ЭВМ.....              | 139 |
| 6.3.5. Ввод и отладка микропрограммы командного цикла<br>и программы решения задачи ..... | 147 |
| 6.4. Контрольные вопросы.....   | 151 |

|   |     |
|---|-----|
| 7. СИСТЕМА ПЕРЕРЫВАНИЯ ЭВМ .....  | 153 |
| 7.1. Основные положения .....   | 153 |
| 7.1.1. Характеристики систем прерывания .....   | 153 |
| 7.1.2. Основные фазы процесса прерывания .....  | 155 |
| 7.2. Аппаратные средства системы прерывания .....                                       | 156 |
| 7.3. Лабораторная работа № 6 .....  | 158 |
| 7.3.1. Задание № 6 .....  | 159 |
| 7.3.2. Определение архитектуры и программирование .....                                 | 161 |
| 7.3.3. Кодирование программы и распределение памяти программ<br>и данных .....          | 167 |
| 7.3.4. Разработка алгоритма работы и микропрограммная<br>реализация ЭВМ .....           | 171 |
| 7.3.5. Ввод и отладка микропрограммы командного цикла и программ<br>решения задач ..... | 176 |
| 7.3.6. Построение диаграммы прерываний .....  | 177 |
| 7.4. Контрольные вопросы .....  | 179 |
| 8. ПРИМЕРЫ ЗАДАНИЙ .....  | 182 |
| БИБЛИОГРАФИЧЕСКИЙ СПИСОК .....  | 184 |

## ПРЕДИСЛОВИЕ

Эффективность подготовки бакалавров в значительной мере зависит от степени овладения ими навыками решения практических задач. Это обуславливает большое значение практикумов в обучении. Настоящий практикум поддерживает изучение дисциплины «Организация ЭВМ и систем» по направлению подготовки 09.03.01 Информатика и вычислительная техника и дисциплины «Архитектура ЭВМ и систем» по направлениям подготовки 09.03.02 Информационные системы и технологии и 09.03.03 Прикладная информатика.

Выполнение практикума предполагает, что студент владеет компетенциями, сформированными при изучении предшествующих дисциплин в области логических и арифметических основ построения ЭВМ.

*Логические основы построения ЭВМ. Знания.* Теория логических функций: определение, способы задания, вычисления и минимизации, функционально полные наборы, суперпозиция функций, системы логических функций, логические элементы и логические схемы. Теория конечных автоматов: определение, виды, способы задания, абстрактный и структурный синтез автоматов.

*Умения.* Описывать работу синтезируемого устройства системой логических функций. Переходить от табличного задания логических функций к формулам и наоборот. Минимизировать системы логических функций, строить по формулам логических функций схемы вычисления их значений. Описывать работу синтезируемого устройства моделью конечного автомата. Задавать автомат с помощью таблицы, графа, формул и переходить от одной формы задания к другой. Строить граф автомата по граф-схеме микропрограммы. Минимизировать число состояний автомата.

*Навыки.* Владеет методами синтеза комбинационных схем с минимальными аппаратными затратами в заданном базисе логических элементов. Владеет методами построения функциональной схемы цифрового устройства обработки информации на основе описания его работы с помощью модели конечного автомата. Методами построения функциональной схемы устройства управ-



ления по исполняемой им микропрограмме на основе описания его работы с помощью модели конечного автомата.

*Арифметические основы построения ЭВМ Знания.* Системы счисления (2, 8, 16), формы представления информации, методы выполнения арифметических операций и способы ускорения их выполнения.

*Умения.* Перевод числа из одной системы счисления в другую, выполнение арифметических операций над числами в форме с фиксированной и плавающей запятой в системах счисления с основаниями: 2, 8, 16 с выявлением всех видов некорректных завершений.

*Навыки.* Владение методами выполнения арифметических операций над числами в форме с фиксированной и плавающей запятой в двоичной системе счисления.

Практикум рассчитан на фронтально-поточную организацию занятий, обеспечивающую необходимый темп усвоения материала, его накопление и преемственность. В учебном пособии описание лабораторных работ дополнено необходимыми теоретическими сведениями и контрольными вопросами, что делает их независимыми от лекционного курса и дает возможность начать практикум одновременно с началом лекционного курса.

Последующее постепенное усложнение исследуемых устройств и использование микропрограмм, разработанных в предыдущих лабораторных работах, позволяют студентам решать достаточно сложные задачи, связанные с разработкой процессоров с программным уровнем управления, за приемлемое время. Сокращению времени на проведение исследований также способствует применение в практикуме специальных программных имитаторов микропрограммируемых вычислительных устройств с развитым интерфейсом пользователя.

Применяемые программы предоставляют необходимые средства для ввода и редактирования данных, микропрограмм и программ, позволяют наблюдать на экране содержимое внутренних регистров исследуемых устройств до и после выполнения микрокоманды или команды, обладают широким набором сервисных функций.

# **1. ОРГАНИЗАЦИЯ ПРАКТИКУМА**

## **1.1. Состав лабораторных работ**

Экспериментальные исследования выполняются по мере изучения соответствующих теоретических вопросов и представляют собой цикл взаимосвязанных лабораторных работ. Практикум состоит из двух частей. В первой части исследуются устройства с микропрограммным, а во второй – с программным уровнем управления. Первая часть включает лабораторные работы: операционные устройства (ОУ), устройства управления (УУ), вычислительные устройства (ВУ). В состав третьей части входят лабораторные работы: учебная ЭВМ, система адресации ЭВМ, система прерывания ЭВМ.

Лабораторный практикум основан на специальных программах, моделирующих лабораторные установки для исследования микропрограммируемых вычислительных структур.

В операционных устройствах исследуются: логическая структура, функциональные возможности, особенности выполнения микроинструкций и формирования логических условий, кодирование микрокоманд. Одной из основных задач является разработка и отладка микропрограммы заданной операции. При этом производится распределение внутренних регистров ОУ, определение форматов данных. Построение УУ не рассматривается, и при разработке микропрограммы используется упрощенный формат микрокоманды. Микрокоманда состоит только из операционной части, которая содержит все поля, необходимые для управления исследуемым ОУ.

В устройствах управления исследуются: логическая структура, способы адресации микрокоманд, возможности приема и анализа значений логических условий и кодов операций, кодирование микрокоманд и размещение микропрограмм в памяти УУ. Разработанная ранее для ОУ микропрограмма преобразуется с учетом возможностей механизма адресации микрокоманд исследуемого УУ, производится ее кодирование и отладка. При этом построение ОУ не рассматривается, а формат используемых микрокоманд включает только управляющую часть микрокоманды, которая содержит все поля, необходимые для управления исследуемым УУ. В процессе отладки микропрограммы предусматривается ввод в УУ значений логических условий и кодов операций.

Вычислительные устройства рассматриваются как микропроцессоры с микропрограммным уровнем управления. В ВУ исследуется совместная работа ОУ и УУ, а также взаимодействие ВУ с запоминающим устройством (ЗУ). В полученные для УУ микропрограммы вводятся полные коды микрокоманд, операционные части которых имеют структуру, использованную при исследовании ОУ, а управляющие – при УУ. Кроме того, операционная часть микрокоманды дополняется полем управления ЗУ. На основе микропрограммы выполнения заданной операции составляется микропрограмма решения учебной задачи, включающая выборку данных из ЗУ и дешифрацию кода операции. Распределяются ячейки памяти микропрограмм и ЗУ. Производится отладка и исследование составленной микропрограммы.

В лабораторной работе, посвященной учебной ЭВМ, решается упрощенная задача микропрограммной реализации ЭВМ на основе микропроцессора с микропрограммным уровнем управления. Выполнение лабораторной работы предполагает: определение архитектуры, разработку структуры и алгоритма работы ЭВМ, составление и отладку микропрограммы командного цикла по тактам и программы учебной задачи по шагам.

Система адресации ЭВМ исследуется на примере учебной ЭВМ, разрабатываемой на основе ЭВМ, реализованной при выполнении предыдущей лабораторной работы. При этом рассматриваются следующие основные вопросы: расширение архитектуры, составление программы с обращением к подпрограмме, распределение памяти программ, разработка алгоритма и микропрограммы командного цикла ЭВМ.

В лабораторной работе, посвященной системе прерывания, исследуется взаимодействие аппаратно-микропрограммных и программных средств ЭВМ в процессе прерывания программ. Выполнение работы включает: дальнейшее развитие архитектуры ЭВМ из предыдущей лабораторной работы, разработку программы для решения основной задачи и программ обработки прерываний, а также разработку микропрограммы командного цикла с подмикропрограммами перехода на прерывающую программу и возврата к прерванной программе. Кроме того, предполагается построение диаграммы, отражающей последовательность переключений ЭВМ с одной программы на другую для заданного множества поступивших запросов на прерывание.

## 1.2. Лабораторные установки на базе диалоговых систем управления

Лабораторные работы выполняются в компьютерном классе, на каждой машине которого программно моделируются специальные лабораторные установки. Предполагается, что лабораторная установка состоит из диалоговой системы управления (ДСУ) и модулей исследуемых устройств (МИУ) (рис. 1.1).

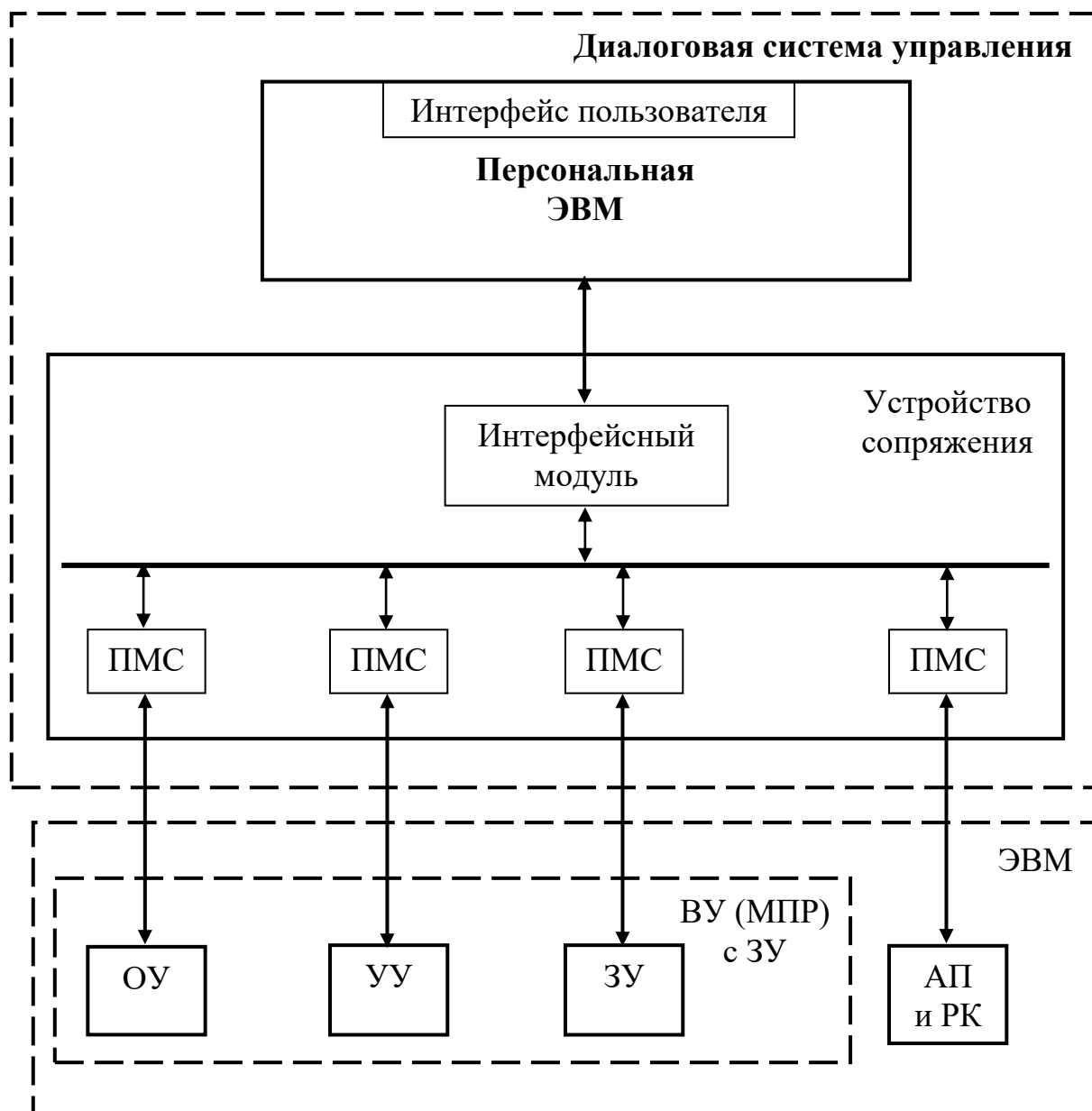


Рис. 1.1. Структура лабораторной установки

ДСУ включает персональную ЭВМ (ПЭВМ) и устройство сопряжения, содержащее набор программируемых модулей сопряжения (ПМС), соединенных внутренней магистралью, и интерфейсный модуль, обеспечивающий подключение устройства сопряжения к персональной ЭВМ. Принцип действия

ПМС основан на программном задании уровней сигналов на каждой линии связи с МИУ и передаче ответных сигналов путем программного опроса выводов МИУ. Наличие в ПМС специальных буферных регистров обеспечивает синхронность изменения сигналов на входах МИУ, а также одновременный прием по линиям связи сигналов с выводов МИУ. В процессе проведения исследования ПЭВМ подает через ПМС в модули исследуемых устройств управляющие и информационные сигналы и принимает ответные сигналы.

В качестве МИУ используются относительно простые устройства, которые подключаются непосредственно к ПМС. Примерами таких устройств являются: ОУ, ЗУ, УУ, а также аппаратура прерывания (АП) и регистр команд (РК). Более сложные устройства образуются из простых устройств путем организации связи между ними программным способом с помощью ПЭВМ через ПМС. Таким образом исследуются микропроцессор и ЭВМ. Программная организация связи позволяет ПЭВМ контролировать информационные потоки, циркулирующие по линиям связи между частями сложного устройства.

Программное обеспечение ПЭВМ выполняет следующие функции:

- подача управляющих сигналов в МИУ;
- прием осведомительных сигналов из МИУ;
- копирование состояния внутренних регистров (ячеек памяти) МИУ после каждого такта (цикла) работы;
- организация работы МИУ по тактам и в автоматическом режиме;
- обеспечение ввода пользователем управляющих воздействий (сигналов, микрокоманд) и их последовательностей (микропрограмм, программ);
- вывод на экран текущего и нового состояния МИУ после каждого такта (цикла) работы;
- запись и считывание с диска введенных микропрограмм (программ) и данных;
- сервисные функции (поддержка справочной системы, проверка корректности задаваемых пользователем управляющих воздействий и др.).

В лабораторных установках компьютерного практикума устройство сопряжения и модули исследуемых устройств моделируются программно. Программное обеспечение лабораторных работ включает четыре программы, соответствующие четырем вариантам лабораторных установок:

«Имитатор операционного устройства» – для лабораторной работы по исследованию ОУ;

«Имитатор устройства управления» – для лабораторной работы по исследованию УУ;

«Имитатор микропрограммируемого микропроцессора» – для лабораторных работ по исследованию ВУ с ЗУ и разработке учебной ЭВМ;

«Имитатор микропрограммируемой микроЭВМ» для лабораторных работ по исследованию системы адресации и системы прерывания программ ЭВМ.

Все программы являются приложениями операционной системы Windows и имеют однотипные интерфейсы пользователя.

### 1.3. Управление лабораторной установкой

Управление лабораторной установкой рассмотрим на примере программы «Имитатор операционного устройства». При запуске программ на экран выводится главное окно приложения (рис. 1.2).

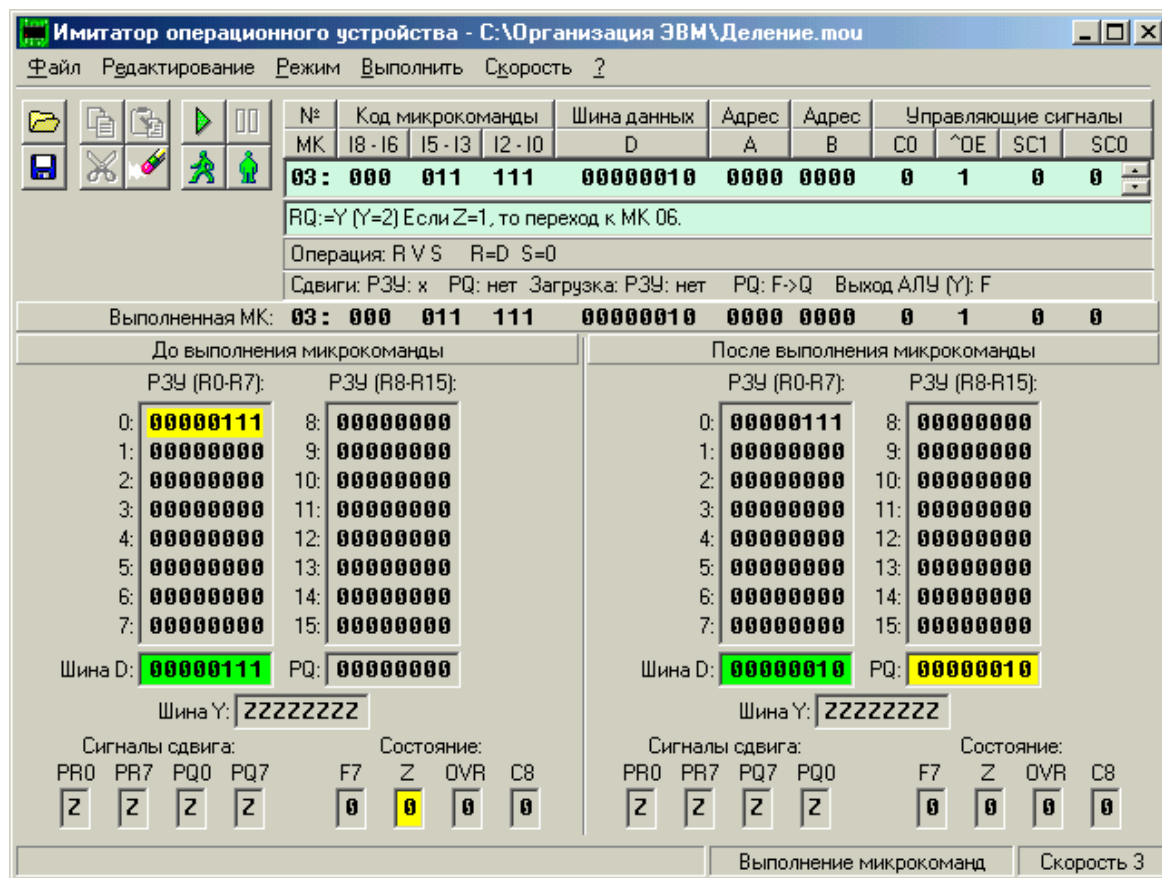


Рис. 1.2. Окно выполнения микрокоманд

Это окно соответствует режиму выполнения микрокоманд и содержит номер и двоичный код, подлежащей выполнению микрокоманды, с комментариями и расшифровкой заданных микроопераций, а также номер и код выполненной ранее микрокоманды. Кроме того, на экране отображается состояние внутренних регистров и внешних сигналов операционного устройства до и после выполнения микрокоманды. Микрокоманды можно вводить в главном окне программы, однако, для удобства ввода микропрограмм предусмотрено специальное окно (рис. 1.3).

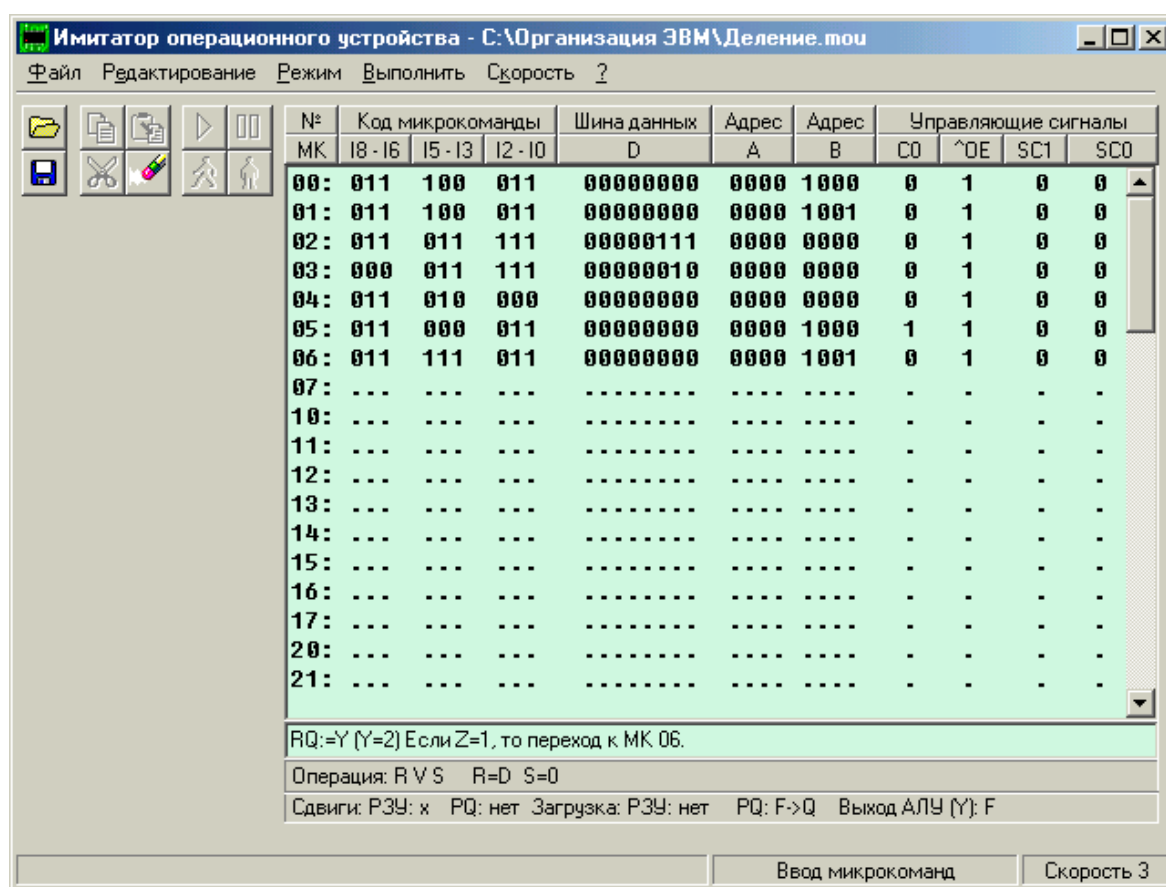


Рис. 1.3. Окно ввода микрокоманд

Управляющее меню окна программы содержит следующие группы команд (рис. 1.4): «Файл», «Редактирование», «Режим», «Выполнить», «Скорость» (скорость автоматического выполнения микропрограммы) и «Справка».

| <b>Файл</b>                   |               |
|-------------------------------|---------------|
| <b>Открыть...</b>             | <b>Ctrl+O</b> |
| <b>Сохранить</b>              | <b>Ctrl+S</b> |
| <b>Сохранить как...</b>       |               |
| <b>Создать отчет по ЛР...</b> |               |
| <b>Выход</b>                  |               |

а)

| <b>Редактирование</b>         |                   |
|-------------------------------|-------------------|
| <b>Вырезать</b>               | <b>Ctrl+X</b>     |
| <b>Копировать</b>             | <b>Ctrl+C</b>     |
| <b>Вставить</b>               | <b>Ctrl+V</b>     |
| <b>Очистить</b>               | <b>Del</b>        |
| <b>Удалить строку</b>         | <b>Ctrl+Y</b>     |
| <b>Вставить пустую строку</b> | <b>Ctrl+Enter</b> |

б)

| <b>Режим</b>                  |                 |
|-------------------------------|-----------------|
| <b>Ввода микрокоманд</b>      | <b>Ctrl+O</b>   |
| <b>Выполнения микрокоманд</b> | <b>Ctrl+S</b>   |
| <b>Переключить</b>            | <b>Ctrl+Tab</b> |

в)

| <b>?</b>                     |  |
|------------------------------|--|
| <b>Содержание</b>            |  |
| <b>Поиск справки о...</b>    |  |
| <b>Использование справки</b> |  |
| <b>О программе...</b>        |  |

г)

| <b>Выполнить</b>                         |                |
|--|----------------|
| <b>Микрокоманду</b>                      | <b>F8</b>      |
| <b>Микрокоманду без перехода к след.</b> | <b>F7</b>      |
| <b>Микропрограмму</b>                    | <b>Ctrl+F9</b> |
| <b>Остановку выполнения</b>              | <b>Ctrl+F2</b> |
| <b>Точка останова</b>                    | <b>Ctrl+F8</b> |
| <b>Сброс</b>                             | <b>Ctrl+R</b>  |

д)

| <b>Скорость</b> |  |
|-----------------|--|
| <b>Высокая</b>  |  |
| <b>Средняя</b>  |  |
| <b>Низкая</b>   |  |











е)

Рис. 1.4. Группы команд управляющего меню программы «Имитатор операционного устройства»: «Файл» (а), «Редактирование» (б), «Режим» (в), «Справка» (г), «Выполнить» (д), «Скорость» (е)

Для ускорения задания наиболее часто встречающихся команд может быть использована панель инструментов управляющего меню. Описание кнопок панели инструментов приведено в табл. 1.1.



Назначение кнопок панели инструментов

|   |   |
|---|---|
|    | – Открыть файл с микропрограммой                  |
|    | – Сохранить микропрограмму                        |
|    | – Скопировать в буфер                             |
|    | – Вырезать в буфер                                |
|    | – Вставить из буфера                              |
|   | – Очистить выделенное                             |
|  | – Выполнить микропрограмму                        |
|  | – Остановить выполнение микропрограммы            |
|  | – Выполнить микрокоманду                          |
|  | – Выполнить микрокоманду без перехода к следующей |

Программы для исследования более сложных устройств имеют дополнительные окна и специфические команды управляющего меню. Особенности интерфейсов пользователя этих программ лабораторного практикума рассматриваются при описании соответствующих лабораторных работ.

## 2. ОПЕРАЦИОННОЕ УСТРОЙСТВО

### 2.1. Основные положения

#### 2.1.1. Три аспекта рассмотрения вычислительного устройства

Можно выделить три «среза» (страты) при анализе вычислительного устройства (ВУ): конструктивное обеспечение (КО), физическое обеспечение (ФО) и информационное обеспечение (ИО) (рис. 2.1). С точки зрения конструктивного обеспечения устройство представляет собой материальный объект, состоящий из определенных материалов и характеризуемый массой, формой, объемом, составом химических элементов. ВУ может занимать небольшую площадь на кристалле интегральной микросхемы или быть автономным устройством.

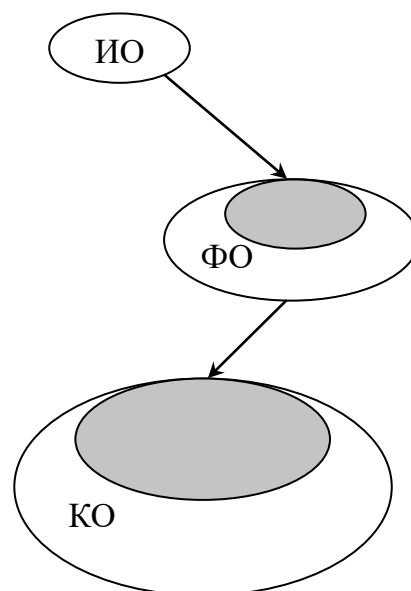


Рис. 2.1. Аспекты рассмотрения ВУ

С точки зрения физического обеспечения вычислительное устройство ЭВМ можно рассматривать как электронную слаботочную систему с большим числом компонентов. К особенностям такой динамической электронной системы следует отнести наличие внутреннего или внешнего возмущающего узла (генератора) и преобладание двух уровней напряжения. При этом высокая рабочая частота переключения элементов системы проявляется в виде тепловых процессов.

С точки зрения информационного обеспечения ВУ представляет собой устройство для ввода, хранения, обработки и выдачи информации (данных, инструкций). Двоичное кодирование информации позволяет сопоставить логический ноль одному, а логическую единицу – другому уровню напряжения в электронной системе и организовать ее работу в соответствии с правилами обработки информации в ВУ. Информационные процессы «накладываются» на физические процессы, а за переключениями напряжений в электронной системе «скрывается» передача и обработка двоичных кодов данных и инструкций.

В лабораторном практикуме вычислительные устройства рассматриваются с точки зрения информационного обеспечения. Минимальной структурной единицей устройства в этом случае является логический элемент. При поступлении на входы логического элемента двоичных сигналов на выходе формируется значение реализуемой им логической функции. Простейший логический элемент представляет собой инвертор.

Из логических элементов строятся узлы ВУ. При этом применяются определенные функционально полные наборы логических элементов: И, ИЛИ, НЕ; И-НЕ, ИЛИ-НЕ и другие. Все узлы делятся на два вида: комбинационные схемы и последовательностные схемы. В комбинационных схемах выходные сигналы зависят только от входных сигналов. При построении комбинационных схем используется теория логических функций. В последовательностных схемах (схемах с памятью) выходные сигналы зависят не только от входных сигналов, но и от внутренних состояний. При построении последовательностных схем используется теория конечных автоматов.

Более крупные структурные единицы – блоки ВУ строятся из узлов. При построении блоков большое распространение получили стандартные узлы: сумматоры, счетчики, дешифраторы, мультиплексоры, сдвигатели и другие.

### 2.1.2. Модель дискретного преобразователя

При анализе и синтезе сложных вычислительных устройств целесообразно использовать модель дискретного преобразователя. В соответствии с этой моделью устройство делится на две части: операционную (ОЧ) и управляющую (УЧ) – и описывается композицией двух конечных автоматов: операционного (ОА) и управляющего (УА) (рис. 2.1).

Операционную часть можно также назвать операционным устройством (ОУ), а управляющую – устройством управления (УУ).

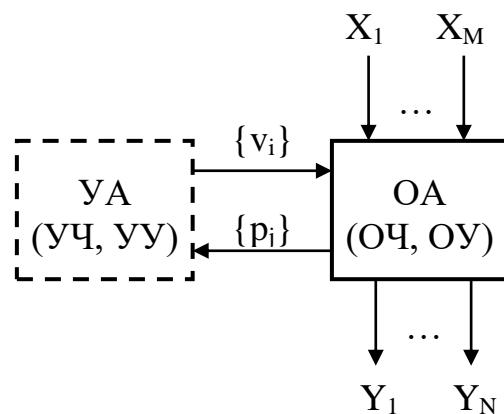


Рис. 2.1. Композиция автоматов

ОУ имеет информационные входы ( $X_1, \dots, X_M$ ) и выходы ( $Y_1, \dots, Y_N$ ). Кроме того, УУ вырабатывает для операционного устройства множество управляющих сигналов  $\{v_i\}$ , а в УУ из ОУ поступает множество сигналов логических условий  $\{p_j\}$ . ОУ описывается конечным автоматом, характеризуемым большим числом входов, выходов и внутренних состояний. Применение методов синтеза конечных автоматов при проектировании ОУ в общем случае затруднено. К функционально-структурным характеристикам ОУ можно отнести: набор операций и алгоритмов их выполнения; формы представления и форматы данных; число и разрядность внешних входов и выходов, а также внутренних регистров; набор микроопераций и множество логических условий.

### ***2.1.3. Микрооперации и логические условия***

Простейшее преобразование информации, выполняемое в каком-либо узле ОУ за один такт, называется микрооперацией (МО). В ОУ реализуются различные, как правило, функционально полные наборы МО. Наиболее распространенными МО являются: хранение, пересылка, счет, суммирование, логические (конъюнкция, сумма по модулю 2, инверсия и др.), а также арифметические и логические сдвиги (в сторону младших и старших разрядов, простые и циклические). Одна и та же МО может быть реализована в различных узлах и, наоборот, один узел может выполнять несколько различных МО. Например, микрооперация счета может быть реализована с помощью следующих аппаратных средств: комбинационной схемы инкремента, накапливающего счетчика, арифметико-логического устройства с использованием входа переноса в младший разряд, регистра сдвига, постоянного запоминающего устройства и др.

Значения логических условий обычно формируются специальными комбинационными схемами. Наиболее распространенными логическими условиями являются следующие признаки: знака (S), нуля (Z), переноса из старшего разряда (C), переполнения (OVR).

#### ***2.1.4. Организация межрегистровых связей***

Существуют два основных способа организации связей между регистрами ОУ: с использованием изолированных шин и с помощью магистралей. Организация связей между регистрами, изолированными шинами, предполагает включение мультиплексора на входе регистра-приемника, при этом выходы регистров-источников подключаются к входам мультиплексора. Соединение регистров достигается с помощью соответствующей настройки мультиплексора, которая обеспечивает подключение к выходу мультиплексора выходов заданного регистра.

При соединении регистров с помощью магистралей могут быть использованы регистры с обычными выходами, выдача данных с которых управляется с помощью специальных логических элементов. При этом объединение выходов регистров в магистраль также реализуется с помощью логических схем. Однако наибольшее распространение имеют магистрали, для создания которых используются регистры с отключаемыми выходами, соответствующее соединение которых и образует магистраль.

#### ***2.1.5. Базовые структуры операционных устройств***

Можно выделить две базовые структуры операционных устройств (ОУ): с закрепленными МО и с общими МО.

*Операционные устройства с закрепленными МО.* Основу таких устройств составляют множества многофункциональных регистров. Пример ОУ приведен на рис. 2.2, а. С каждым многофункциональным регистром (МФР) в соответствии с алгоритмами операций связывается набор выполняемых на регистре микроопераций  $\{v_i\}$  (счет, суммирование, сдвиг и т. п.) и множество формируемых им логических условий  $\{p_i\}$  (признак нуля, знака, значение младшего разряда и т. п.). Число регистров ОУ определяется максимальным числом данных, которые одновременно необходимо хранить в процессе выполнения алгоритмов. Связи между регистрами реализуются с помощью изолированных шин в соответствии с направлениями пересылок данных, выполняемыми в алгоритмах операций. Устройства с подобной структурой широко используются при изучении различных алгоритмов выполнения операций умножения и деления.

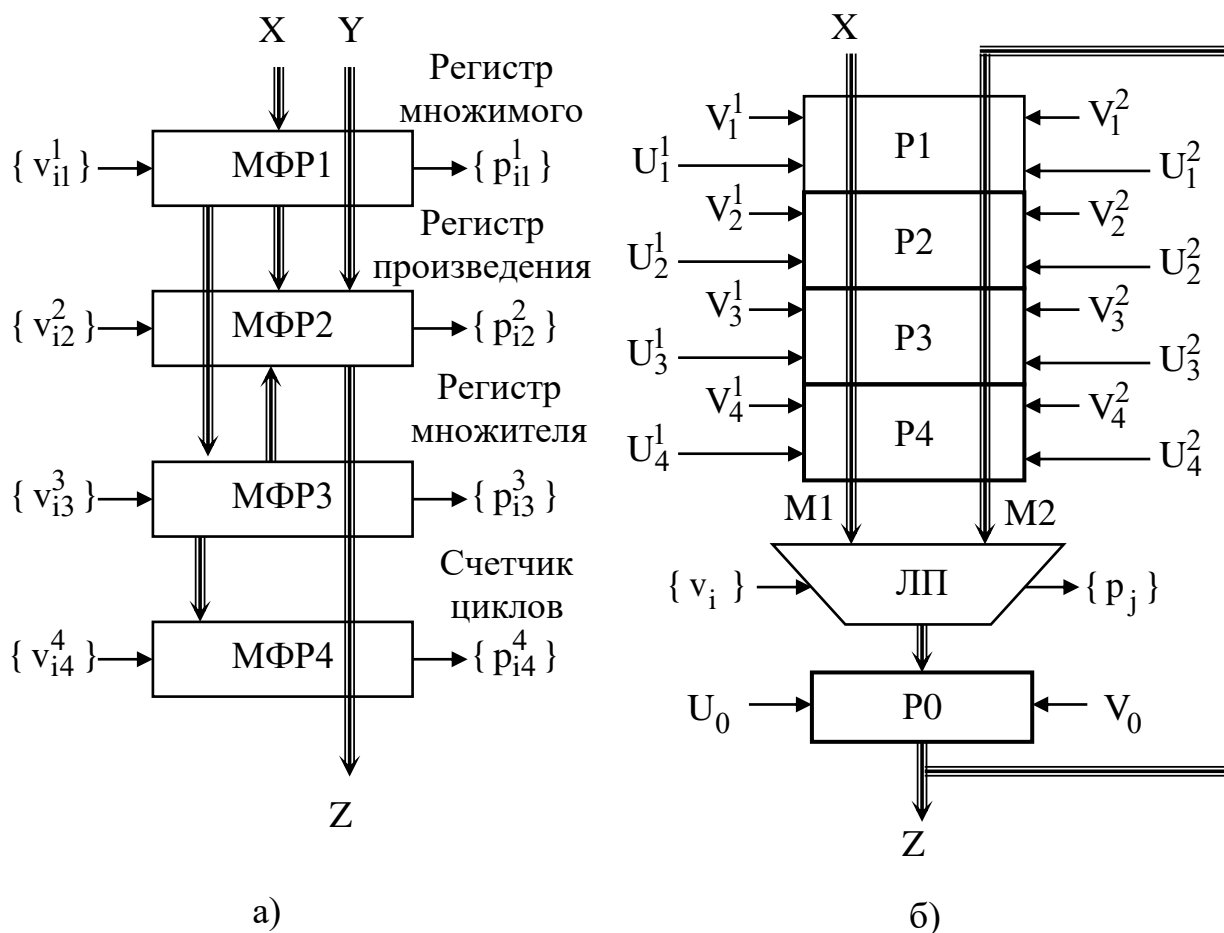


Рис. 2.2. Операционные устройства: с закрепленными (а) и общими (б) микрооперациями

Важным достоинством ОУ с закрепленными микрооперациями является возможность достижения высокого быстродействия за счет одновременного выполнения большого числа МО на различных многофункциональных регистрах. В то же время ОУ с закрепленными МО – это специализированное устройство, структура которого определяется особенностями выбранных алгоритмов операций. Причем даже незначительное изменение алгоритмов, может повлечь за собой необходимость модификации структуры. Кроме того, ОУ с закрепленными МО имеют низкую степень регулярности структуры, что является недостатком при ориентации на изготовление устройств с использованием технологии больших интегральных схем.

*Операционные устройства с общими МО.* В ОУ с общими МО выделяются две основных части: блок регистров и логический преобразователь (ЛП). Блок регистров представляет собой набор унифицированных регистров, объ-

единенных одной или двумя магистралями. Логический преобразователь, как правило, строится на основе комбинационного АЛУ, которое дополняется необходимыми схемами таким образом, чтобы обеспечивалось выполнение функционально полного набора микроопераций.

Пример ОУ с общими МО приведен на рис. 2.2, б. В примере блок регистров содержит четыре регистра (P1–P4), объединенных двумя магистралями (M1, M2). Содержимое  $i$ -го регистра ( $i = 1, 2, 3, 4$ ) с помощью сигнала  $V_i^1$  может быть выдано на магистраль M1, а с помощью сигнала  $V_i^2$  – на магистраль M2. Аналогично данные с магистрали M1 записываются в  $i$ -й регистр по сигналу  $U_i^1$ , а с магистрали M2 – по сигналу  $U_i^2$ .

Вместо блока регистров в ОУ с общими МО может быть использовано регистровое ЗУ. Логический преобразователь обеспечивает выполнение микроопераций  $\{v_i\}$  (счет, суммирование, сдвиги в сторону младших и старших разрядов на один разряд, формирование дополнительного кода, инверсия, конъюнкция, дизъюнкция и др.). Множество логических условий  $\{p_i\}$  (признак нуля, знака, переполнения значение старшего разряда и т. п.) формируется преобразователем по результатам выполнения заданных микроопераций. В состав ОУ с общими микрооперациями входит также буферный регистр P0, который используется для временного хранения результатов преобразования перед записью их в блок регистров. Запись данных в этот регистр выполняется по сигналу  $U_0$ , а выдача на магистраль M2 – по сигналу  $V_0$ . Достоинствами ОУ с общими МО являются:

- универсальность, рассматриваемая как возможность при достаточном числе регистров и функционально полном наборе микроопераций реализовать в устройстве любой алгоритм;
- гибкость, которая проявляется в том, что изменения алгоритмов, как правило, не требуют изменения структуры устройства;
- возможность минимизировать аппаратные затраты за счет совместной реализации в логическом преобразователе выполняемых микроопераций;
- более высокая степень регулярности структуры.

Недостатком ОУ с общими МО является более низкое по сравнению с ОУ с закрепленными МО быстродействие.

Выделенные структуры ОУ следует рассматривать как два «полюса». Практически используемые ОУ обычно в той или иной степени «тяготеют» к одному из этих «полюсов».

## 2.2. Структура и микроинструкции

Исследуемое операционное устройство выполняет: сложение, вычитание, инкремент и декремент, сдвиги в сторону младших и старших разрядов, пять логических операций и формирует четыре признака результата. ОУ состоит из следующих блоков: арифметико-логического (БАЛ), внутренней памяти (БВП), регистра Q (БР), мультиплексоров (БМП) и управления (БУ). Структурная схема ОУ приведена на рис. 2.3.

**Блок арифметико-логический.** БАЛ включает арифметико-логическое устройство (АЛУ), селектор источника данных (СИД), селектор выходных данных (СВД). БАЛ предназначен для выполнения арифметических и логических операций над двумя операндами с формированием сигналов: переноса из старшего разряда C8; признака нулевого результата АЛУ Z; переполнения АЛУ OVR; старшего разряда результата АЛУ F7. СИД позволяет задавать в качестве пары операндов на входах АЛУ данные от четырех источников: с внешней шины данных D7...D0, с регистров РА и РВ РЗУ, из дополнительного регистра Q.

Выбор пар операндов осуществляется с помощью мультиплексоров МП1 и МП2 в соответствии с кодом микроопераций на управляющих входах I2...I0 (табл. 2.1).



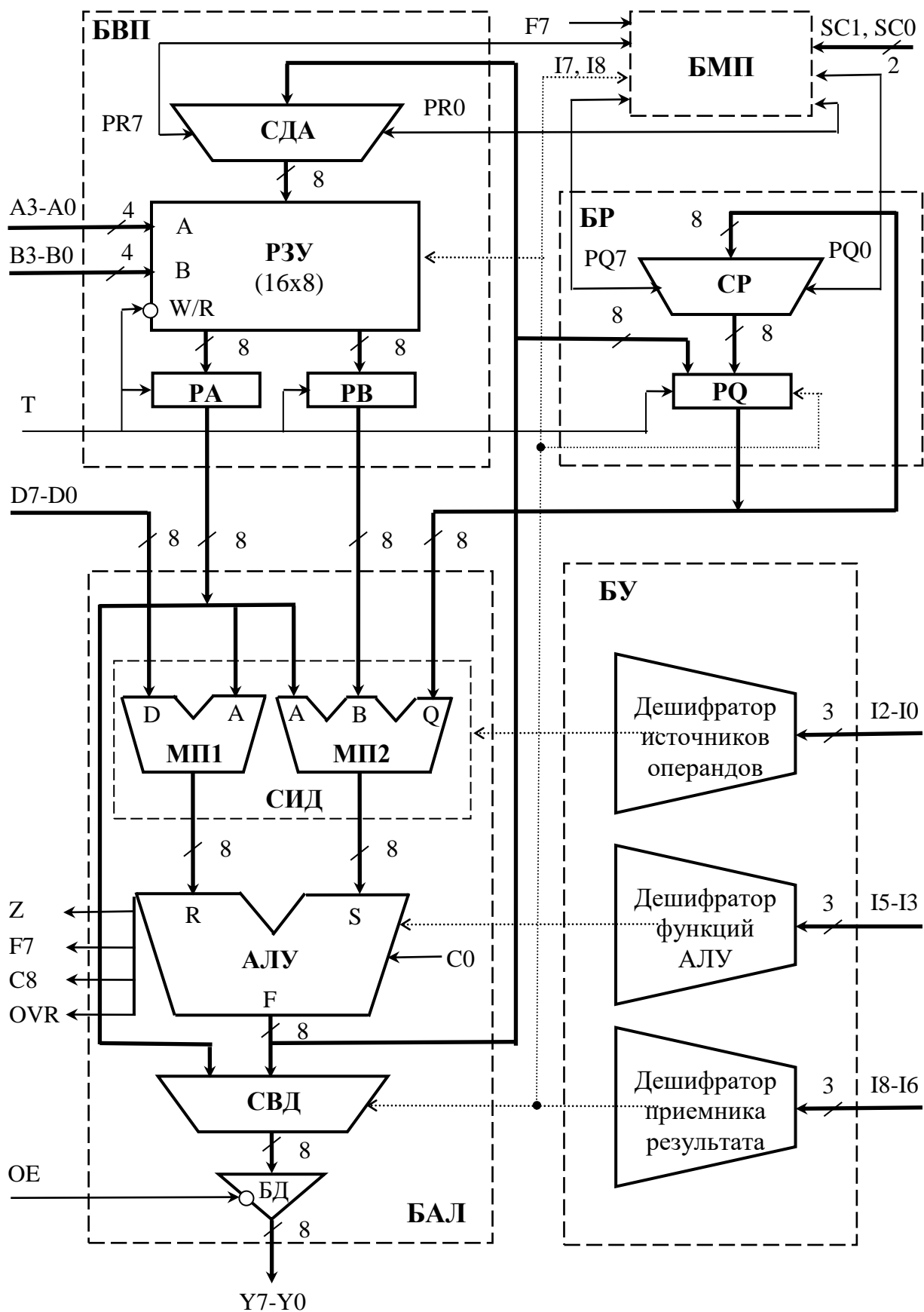


Рис. 2.3. Структурная схема операционного устройства

Таблица 2.1

## Источники АЛУ

| Микрокод |    |    |     | Источник |   |
|----------|----|----|-----|----------|---|
| I2       | I1 | I0 | код | R        | S |
| 0        | 0  | 0  | 0   | A        | Q |
| 0        | 0  | 1  | 1   | A        | B |
| 0        | 1  | 0  | 2   | 0        | Q |
| 0        | 1  | 1  | 3   | 0        | B |
| 1        | 0  | 0  | 4   | 0        | A |
| 1        | 0  | 1  | 5   | D        | A |
| 1        | 1  | 0  | 6   | D        | Q |
| 1        | 1  | 1  | 7   | D        | 0 |

Таблица 2.2

## Источники АЛУ

| Микрокод |    |    |     | Операция<br>АЛУ            |
|----------|----|----|-----|----------------------------|
| I5       | I4 | I3 | код |                            |
| 0        | 0  | 0  | 0   | $R + S + C0$               |
| 0        | 0  | 1  | 1   | $S - R - 1 + C0$           |
| 0        | 1  | 0  | 2   | $R - S - 1 + C0$           |
| 0        | 1  | 1  | 3   | R or S                     |
| 1        | 0  | 0  | 4   | R and S                    |
| 1        | 0  | 1  | 5   | $\wedge R$ and S           |
| 1        | 1  | 0  | 6   | $R \text{ xor } S$         |
| 1        | 1  | 1  | 7   | $\wedge(R \text{ xor } S)$ |

Таблица 2.3

## Управление приемником АЛУ

| Микрокод |    |    |     | РЗУ    |             | PQ     |             | Вых<br>Y | СДА  |      | СР   |      |
|----------|----|----|-----|--------|-------------|--------|-------------|----------|------|------|------|------|
| I8       | I7 | I6 | код | сдвиг  | загрузка    | сдвиг  | загрузка    |          | PR0  | PR7  | PQ0  | PQ7  |
| 0        | 0  | 0  | 0   | х      | нет         | нет    | $F - > Q$   | F        | х    | х    | х    | х    |
| 0        | 0  | 1  | 1   | х      | нет         | х      | нет         | F        | х    | х    | х    | х    |
| 0        | 1  | 0  | 2   | нет    | $F - > B$   | х      | нет         | A        | х    | х    | х    | х    |
| 0        | 1  | 1  | 3   | нет    | $F - > B$   | х      | нет         | F        | х    | х    | х    | х    |
| 1        | 0  | 0  | 4   | вправо | $F/2 - > B$ | вправо | $Q/2 - > Q$ | F        | F0   | вход | Q0   | вход |
| 1        | 0  | 1  | 5   | вправо | $F/2 - > B$ | х      | нет         | F        | F0   | вход | Q0   | х    |
| 1        | 1  | 0  | 6   | влево  | $2F - > B$  | влево  | $2Q - > B$  | F        | вход | F7   | вход | Q7   |
| 1        | 1  | 1  | 7   | влево  | $2F - > B$  | х      | нет         | F        | вход | F7   | х    | Q7   |

Таблица 2.4

## Управление сдвигами в ОУ

| Микрокод |    |     |     | Коды на выходах СДА (F)<br>и СР (Q)                 | Вид сдвига                           |
|----------|----|-----|-----|---|--------------------------------------|
| I8       | I7 | SC1 | SC0 |   |                                      |
| 0        | х  | х   | х*  | $F = F7 F6 \dots F1 F0 \quad Q = Q7 Q6 \dots Q1 Q0$ | сдвига нет                           |
| 1        | 0  | 0   | 0   | $F = 0 F7 \dots F2 F1 \quad Q = 0 Q7 \dots Q2 Q1$   | логический вправо                    |
| 1        | 0  | 0   | 1   | $F = F0 F7 \dots F2 F1 \quad Q = Q0 Q7 \dots Q2 Q1$ | циклический вправо                   |
| 1        | 0  | 1   | 0   | $F = Q0 F7 \dots F2 F1 \quad Q = F0 Q7 \dots Q2 Q1$ | циклический двойного слова вправо    |
| 1        | 0  | 1   | 1   | $F = F7 F7 \dots F2 F1 \quad Q = F0 Q7 \dots Q2 Q1$ | арифметический двойного слова вправо |
| 1        | 1  | 0   | 0   | $F = F6 F5 \dots F0 0 \quad Q = Q6 Q5 \dots Q0 0$   | логический влево                     |
| 1        | 1  | 0   | 1   | $F = F6 F5 \dots F0 F7 \quad Q = Q6 Q5 \dots Q0 Q7$ | циклический влево                    |
| 1        | 1  | 1   | 0   | $F = F6 F5 \dots F0 Q7 \quad Q = Q6 Q5 \dots Q0 F7$ | циклический двойного слова влево     |
| 1        | 1  | 1   | 1   | $F = F6 F5 \dots F0 Q7 \quad Q = Q6 Q5 \dots Q0 0$  | арифметический двойного слова влево  |

\* при любом значении управляющего сигнала

Арифметические операции в АЛУ выполняются с учетом значения входного переноса  $C_0$ . Необходимая операция выполняется в АЛУ по сигналам от БУ, которые вырабатываются в зависимости от кода, поступающего на входы  $I_5...I_3$ , в соответствии с табл. 2.2. На выходную тристабильную шину  $Y_7...Y_0$  информация может быть передана через СВД с выхода регистра РА РЗУ или с выхода АЛУ. Перевод выходной шины  $Y_7...Y_0$  в состояние «выключено» осуществляется подачей сигнала «Лог. 1» на вход ОЕ буфера данных БД.

**Блок внутренней памяти.** БВП содержит 16 регистров общего назначения, которые составляют регистровое запоминающее устройство (РЗУ), регистры РА и РВ и сдвигатель данных АЛУ (СДА).

Из РЗУ могут быть одновременно считаны два слова, адреса которых заданы на адресных входах  $A_3...A_0$  и  $B_3...B_0$ , при этом, если на адресных входах установлены одинаковые адреса, то на обоих выходах РЗУ появляются идентичные данные. Запись в РЗУ возможна только по адресу В. Данные перед записью в РЗУ могут быть сдвинуты с помощью СДА на один разряд в сторону старших или младших разрядов. Для выполнения сдвигов необходимо установить соответствующий код на входах МК  $I_8...I_6$  (см. табл. 2.3) и подать информацию на один из выводов  $PR_0$  или  $PR_7$  в зависимости от направления сдвига (выводы сдвига младшего разряда  $PR_0$  и старшего разряда  $PR_7$  двунаправленные и могут служить в зависимости от направления сдвига как входами, так и выходами).

Запись данных в РЗУ производится в момент отрицательного фронта тактового сигнала Т. При наличии сигнала «Лог. 1» на тактовом входе информация с входов регистров РА и РВ передается на их выходы. При поступлении сигнала «Лог. 0» на тактовый вход на выходах РА и РВ сохраняются те состояния, которые были на их входах в момент записи данных в РЗУ.

**Блок регистра Q.** БР состоит из 8-разрядного регистра Q (PQ) и сдвигателя регистра (СР). Работа СР аналогична работе СДА в БВП (двунаправленные выводы  $PQ_0$  и  $PQ_7$  аналогичны выводам  $PR_0$  и  $PR_7$  в СДА). Запись информации в PQ с выходов АЛУ или с выходов самого PQ происходит в момент перехода тактового сигнала Т из «0» в «1».

**Блок мультиплексоров.** БМП используется для коммутации выводов PR7, PQ7 и F7 с выводами PR0, PQ0 и позволяет в соответствии с управляющими сигналами на входах I7, I8, SC0, SC1 организовать выполнение в ОУ восьми видов сдвигов, приведенных в табл. 2.4.

**Блок управления.** БУ содержит три дешифратора: источников операндов, функций АЛУ и приемника результата и обеспечивает выполнение в ОУ микроопераций, задаваемых с помощью двоичных кодов на входах I8-I0.

Таким образом, в ОУ данные подаются на вход D, а результат выполнения операции может сниматься с выхода Y, кроме того, формируются признаки результата: Z, C8, OVR, F7. Управляющие сигналы A, B, I, SC, C0, ^OE подаются параллельно на соответствующие входы ОУ.

Формат операционной части микрокоманды для исследования ОУ приведен на рис. 2.4.

| Коды микроопераций |       |       | Данные | Адрес РЗУ |       | Упр. сигналы |     |         |
|--------------------|-------|-------|--------|-----------|-------|--------------|-----|---------|
| I8-I6              | I5-I3 | I2-I0 | D7-D0  | A3-A0     | B3-B0 | C0           | ^OE | SC1-SC0 |

Рис. 2.4. Формат операционной части микрокоманды для ОУ

### 2.3. Лабораторная работа № 1

Выполнение лабораторной работы складывается из домашней подготовки, экспериментальных исследований и оформления отчета. Для домашней подготовки необходимо получить у преподавателя задание на разработку микропрограммы для ОУ. Домашняя подготовка включает: разработку алгоритма и микропрограммы для операции, указанной в задании. Экспериментальные исследования заключаются во вводе, отладке и выполнении разработанной микропрограммы. Отчет по лабораторной работе должен содержать: титульный лист, задание, граф-схему алгоритма, распределение внутренних регистров ОУ, граф-схему микропрограммы, текст отлаженной микропрограммы с комментариями.

Рассмотрим выполнение лабораторной работы на примере задания № 1.

### 2.3.1. Задание № 1

Разработать микропрограмму для операции деления целых положительных чисел нацело:  $Z = ]X / Y[$ , где  $X, Y, Z$  – целые положительные числа в диапазоне от 0 до 127. Кроме результата  $Z$  предусмотреть формирование значения признака переполнения  $P$  ( $P = 1$ , если  $Y = 0$ , иначе  $P = 0$ ). Числа  $X$  и  $Y$  перед выполнением операции заносятся в регистры ОУ по шине D. Результат  $Z$  и значение признака переполнения  $P$  фиксируются после выполнения операции в регистрах ОУ.

### 2.3.2. Разработка алгоритма

Граф-схема алгоритма приведена на рис. 2.5, где  $B$  и  $C$  – вспомогательные переменные. Перед началом цикла деления присваиваются нулевые значения частному ( $Z$ ) и признаку переполнения ( $P$ ). Затем проверяется на равенство нулю делитель. Если делитель равен нулю, то признаку переполнения присваивается единичное значение, и вычисления завершаются, иначе выполняется цикл деления. В цикле деления вспомогательной переменной  $B$ , которая первоначально имеет значение делимого  $X$ , присваивается значение разности этой переменной и делителя  $Y$ . Если полученная разность меньше нуля, то вычисления завершаются, иначе частное увеличивается на единицу, и снова выполняется вычитание делителя  $Y$  из вспомогательной переменной  $B$ .

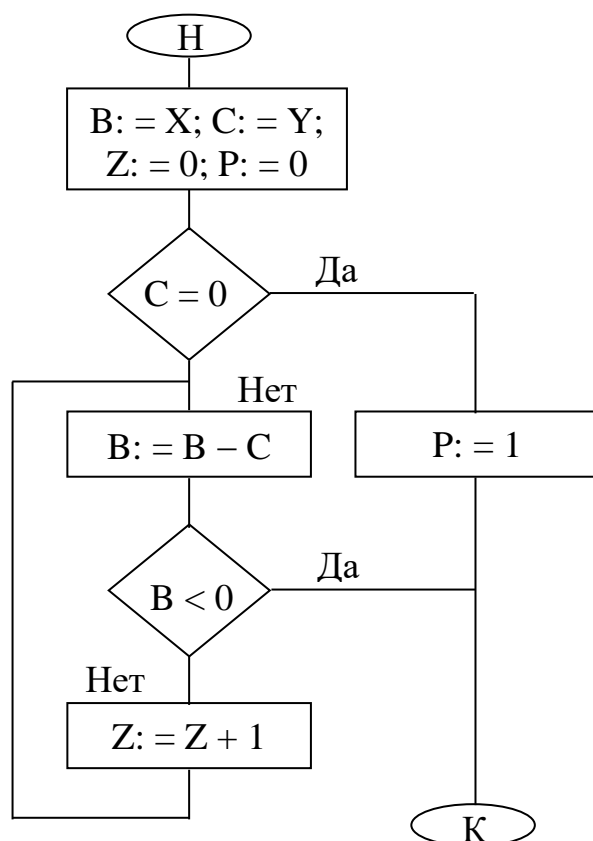


Рис. 2.5. Граф-схема алгоритма деления

### 2.3.3. Распределение регистров и разработка микропрограммы

Распределение внутренних регистров операционного устройства, используемое при выполнении деления чисел нацело, приведено на рис. 2.6, а граф-схема микропрограммы выполнения операции – на рис. 2.7.

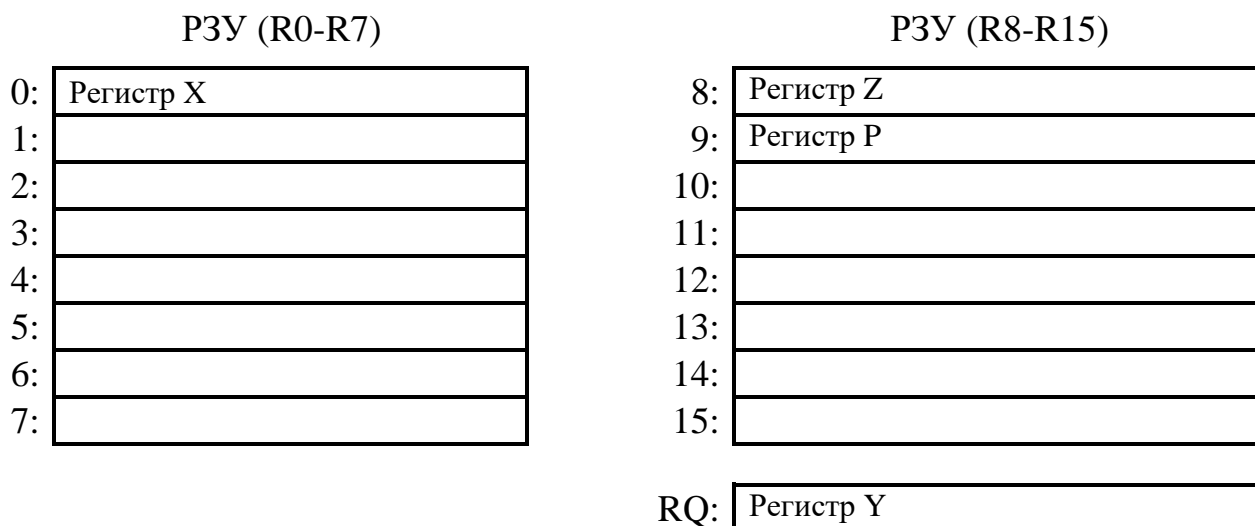


Рис. 2.6. Распределение регистров ОУ для выполнения операции деления

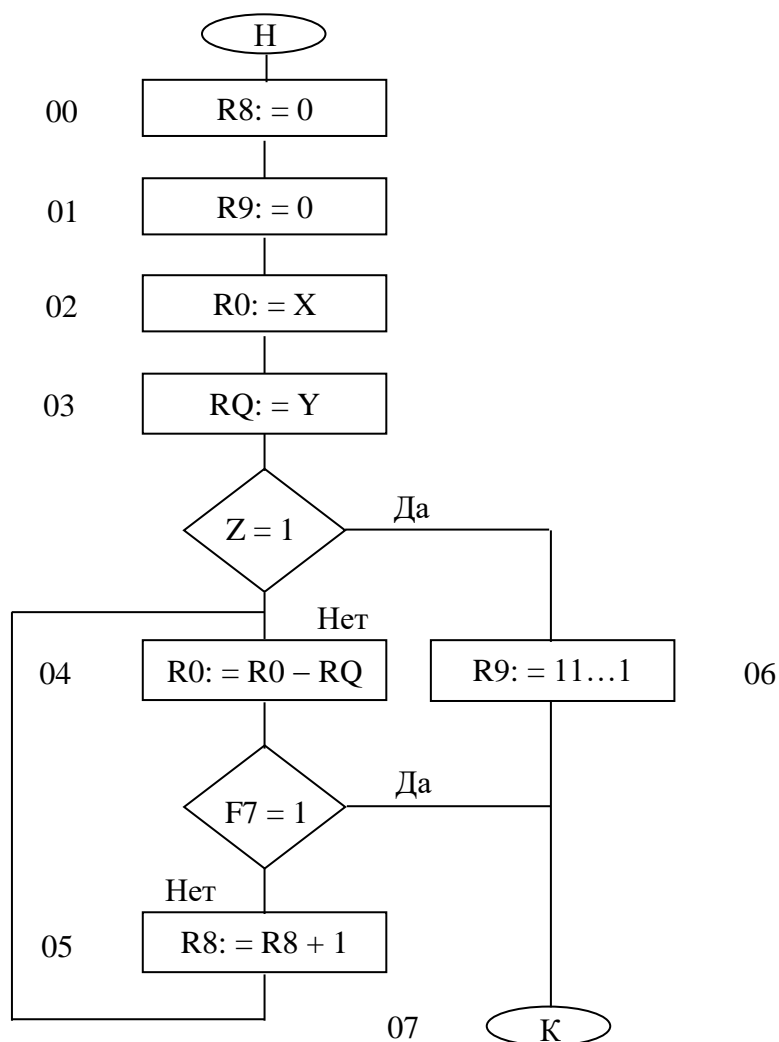


Рис. 2.7. Граф-схема микропрограммы деления

### 2.3.4. Кодирование микропрограммы

Текст микропрограммы в двоичных кодах с комментариями приведен в табл. 2.5. В соответствии с принятым алгоритмом выполнения операции перед началом цикла деления с помощью пары микрокоманд (МК) обнуляется регистр частного R8 и регистр признака переполнения R9 (см. рис. 2.5, МК 00, 01). Затем в регистр делимого R0 и регистр делителя RQ заносятся с внешней шины D исходные данные (МК 02, 03), причем при занесении делителя формируется и проверяется признак нуля (Z). Если признак нуля  $Z = 1$ , то осуществляется переход к МК 6, иначе выполняется следующая по порядку микрокоманда (МК 04), которая является первой микрокомандой цикла деления. С помощью МК 04 содержимое регистра RQ вычитается из содержимого регистра R0. Если получается отрицательный результат ( $F7 = 1$ ), то осуществляется переход к МК 07, иначе выполняется следующая по порядку микрокоманда (МК 05). В этой микрокоманде содержимое регистра частного (R8) увеличивается на единицу и производится переход к микрокоманде начала цикла деления (МК 04).

Таблица 2.5

Микропрограмма деления чисел нацело

| МК   | Операционная часть |       |       |         |           |       |              |             |         |
|--|--------------------|-------|-------|---------|-----------|-------|--------------|-------------|---------|
| №  | Коды микроопераций |       |       | Данные  | Адрес РЗУ |       | Упр. сигналы |             |         |
| N  | I8-I6              | I5-I3 | I2-I0 | D7-D0   | A3-A0     | B3-B0 | C0           | $\wedge$ OE | SC1-SC0 |
| 00   | 011                | 100   | 011   | 0000000 | 0000      | 1000  | 0            | 1           | 00      |
| R8: = 0 (Z: = 0)                                 |                    |       |       |         |           |       |              |             |         |
| 01   | 011                | 100   | 011   | 0000000 | 0000      | 1001  | 0            | 1           | 00      |
| R9: = 0 (P: = 0)                                 |                    |       |       |         |           |       |              |             |         |
| 02   | 011                | 011   | 111   | 0000111 | 0000      | 0000  | 0            | 1           | 00      |
| R0: = X (X: = 7)                                 |                    |       |       |         |           |       |              |             |         |
| 03   | 000                | 011   | 111   | 0000010 | 0000      | 0000  | 0            | 1           | 00      |
| RQ: = Y (Y: = 2) Если Z = 1, то переход к МК 06. |                    |       |       |         |           |       |              |             |         |
| 04   | 011                | 010   | 000   | 0000000 | 0000      | 0000  | 1            | 1           | 00      |
| R0: = R0 – RQ Если F7 = 1, то переход к МК 07.   |                    |       |       |         |           |       |              |             |         |
| 05   | 011                | 000   | 011   | 0000000 | 0000      | 1000  | 1            | 1           | 00      |
| R8: = R8 + 1 Переход к МК 04.                    |                    |       |       |         |           |       |              |             |         |
| 06   | 011                | 111   | 011   | 0000000 | 0000      | 1001  | 0            | 1           | 00      |
| R9: = 11111111                                   |                    |       |       |         |           |       |              |             |         |
| 07   |                    |       |       |         |           |       |              |             |         |

### **2.3.5. Ввод и отладка микропрограммы**

В программе «Имитатор операционного устройства» создается файл с расширением «.mou», представляющий собой двоичный код микропрограммы для операционного устройства. Для ввода и отладки микропрограмм в программе предусмотрены специальные окна и команды, которые подробно рассмотрены в разделе 1.3.

В данной лабораторной работе построение УУ не рассматривается. При разработке микропрограммы используется упрощенный формат микрокоманды, которая содержит только поля, необходимые для управления исследуемым ОУ. Поэтому выполнение функций УУ по анализу логических условий, формируемых в ОУ, и переходу к следующей микрокоманде возлагается на пользователя. Пользователь может выбирать и выполнять микрокоманды с помощью кнопок панели инструментов и команд управляющего меню главного окна программы «Имитатор операционного устройства». В программе также предусмотрено и автоматическое выполнение микропрограммы ОУ, при котором микрокоманды выполняются в порядке возрастания их номеров. С помощью группы команд «Скорость» может быть задана высокая, средняя или низкая скорость выполнения микропрограммы. Используя группу команд «Выполнить», можно задавать точки останова.

## **2.4. Контрольные вопросы**

### *Основные положения*

1. Что используется в качестве математической модели функционирования ОУ?
2. Простейшее преобразование информации, выполняемое в ОУ за один такт, называется:
  - командой;
  - микрокомандой;
  - операцией;
  - микрооперацией.



3. Множество микроопераций, выполняемых в ОУ за один такт, называется:
- командой;
  - микрокомандой;
  - операцией;
  - макрооперацией.
4. Характерной чертой ОУ с общими микрооперациями является:
- реализация всех преобразований информации в единой комбинационной схеме;
  - использование накапливающего сумматора;
  - распределение микроопераций по регистрам операционного устройства;
  - организация связей между регистрами в соответствии с пересылками данных, задаваемых в микропрограммах выполняемых операций.
5. Как определить переполнение разрядной сетки при сложении чисел в дополнительном коде?
6. Чем арифметический сдвиг двоичного кода числа со знаком отличается от логического сдвига?

*Исследуемое устройство*

7. Перечислите одноместные и двухместные операции, выполняемые в ОУ.
8. При сдвиге содержимого регистра РЗУ признаки логических условий формируются для результата или исходного операнда?
9. Что может выступать в качестве источника операнда для АЛУ?
10. Приведите пример операции, в которой источниками операндов являются два различных регистра, а результат записывается в третий регистр.
11. Как обнулить какой-либо регистр РЗУ?
12. В чем состоит различие признаков C8 и OVR?

### 3. УСТРОЙСТВО УПРАВЛЕНИЯ

#### 3.1. Основные положения

##### 3.1.1. Два подхода к построению устройств управления

При исследовании ОУ устройство управления (УУ) было временно исключено из рассмотрения. Возвращаясь к модели дискретного преобразователя

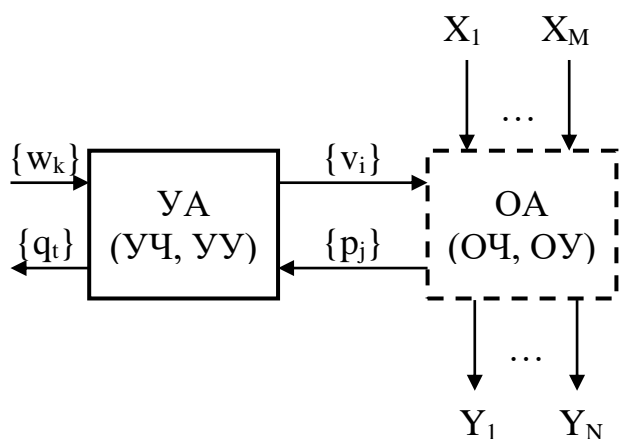


Рис. 3.1. Композиция автоматов

(см. раздел 3.1), сосредоточим внимание на управляющем автомате (рис. 3.1). УУ вырабатывает для операционного устройства множество управляющих сигналов  $\{v_i\}$ , а в УУ из ОУ поступает множество сигналов логических условий  $\{p_j\}$ . Кроме того, в общем случае в УУ поступает множество управляющих сигналов  $\{w_k\}$  из устройства более

высокого уровня управления, для которого устройство управления формирует множество осведомительных сигналов  $\{q_t\}$ . УУ описывается конечным автоматом, имеющим по сравнению с операционным устройством, как правило, относительно небольшое число входов, выходов и внутренних состояний.

На практике используются два основных подхода к построению УУ. Первый подход основан на формальном синтезе управляющего автомата, использующем теорию конечных автоматов. Устройство в этом случае обычно называется устройством управления с «жесткой» логикой, хотя при его реализации могут быть использованы программируемые логические матрицы или ПЗУ. Второй подход ориентируется на микропрограммные автоматы, при разработке которых используются идеи программирования (микропрограммирования). Получаемые при таком подходе устройства получили название устройств управления с «программируемой» логикой. В лабораторной работе по устройствам управления исследуются УУ с «программируемой» логикой.

### 3.1.2. Пример устройства управления

Основу УУ с «программируемой» логикой составляет блок памяти (БП) в котором в закодированном виде хранятся микропрограммы операций, выполняемых в ОУ. Коды микрокоманд (МК) считываются из БП (М) в регистр микрокоманд (РМК) и обеспечивают подачу в ОУ необходимых сигналов микроопераций. Адрес очередной считываемой микрокоманды формируется в регистре адреса (РА) блока памяти. Проиллюстрируем построение и работу УУ на примере простейшей микропрограммы, приведенной на рис. 3.2, а.

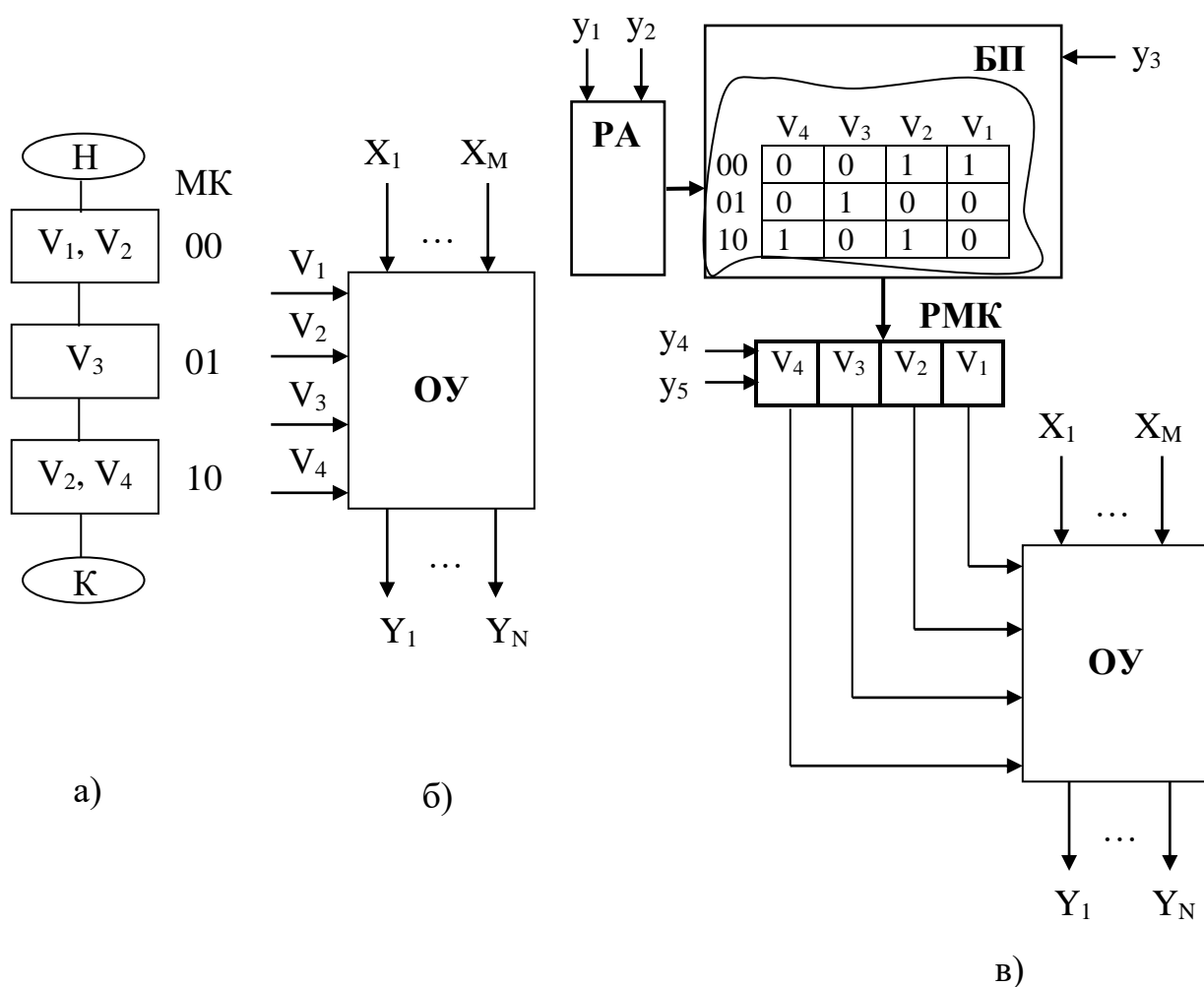


Рис. 3.2. Микропрограмма (а), операционное устройство (б), устройство управления и его подключение к ОУ (в):  $y_1 - \text{РА} = 0$ ;  $y_2 - \text{РА} = \text{РА} + 1$ ;  $y_3$  – чтение из БП;  $y_4 - \text{РМК} = 0$ ;  $y_5 - \text{РМК} = M[\text{РА}]$

Микропрограмма состоит из трех МК и содержит обозначения сигналов четыре различных микрооперации ( $V_1$ ,  $V_2$ ,  $V_3$ ,  $V_4$ ), выполняемых в ОУ (рис. 3.2 б). Коды микрокоманд размещаются в блоке памяти в порядке их выполнения. Каждой микрооперации в коде МК соответствует отдельный разряд. Число разрядов МК в данном случае равно числу различных микроопераций в микропрограмме. Кодирование МК осуществляется следующим образом. Если в микрокоманде должна быть выполнена какая-либо микрооперация, то для выработки необходимого управляющего сигнала в соответствующем ей разряде кода МК записывается «1», а в противном случае – «0». Закодированная микропрограмма для рассматриваемого примера представлена в виде таблицы на рис. 3.2 в, где первый столбец таблицы содержит адреса ячеек памяти, по которым размещаются микрокоманды.

Перед началом работы УУ производится обнуление РА ( $y_1$ ) и РМК ( $y_4$ ). Затем из блока памяти считывается микрокоманда, размещенная в нулевой ячейке ( $y_3$ ). Эта МК помещается в РМК ( $y_5$ ), что обеспечивает подачу управляющих сигналов в ОУ. После выполнения микроопераций в ОУ содержимое РА увеличивается на единицу ( $y_2$ ), а затем считывается и выполняется следующая микрокоманда. В результате УУ подает на ОУ последовательность наборов управляющих сигналов, заданную микропрограммой.

### ***3.1.3. Способы кодирования микроопераций***

На практике применяются два способа кодирования микроопераций. Первый способ использует так называемое «горизонтальное» кодирование МО, а второй – «вертикальное». «Горизонтальное» кодирование микроопераций было рассмотрено выше в примере (см. рис. 3.2). Возвращаясь к этому примеру, «вертикальное» кодирование МО можно пояснить следующим образом (рис. 3.3, а):

- микрокоманда делится на поля (**a**, **b**), число которых, как правило, определяется максимальным числом одновременно выполняемых в микрокомандах МО;

- микрооперации распределяются по полям таким образом, чтобы в одном поле находились только никогда одновременно не выполняемые в микропрограмме МО (**a** –  $V_1, V_3, V_4$ , **b** –  $V_2$ );
- производится кодирование МО в каждом поле, при этом, если в микропрограмме есть микрокоманды, в которых не выполняется ни одна из МО поля, то вводится «пустая» МО ( $\emptyset$ ) с соответствующим ей кодом.

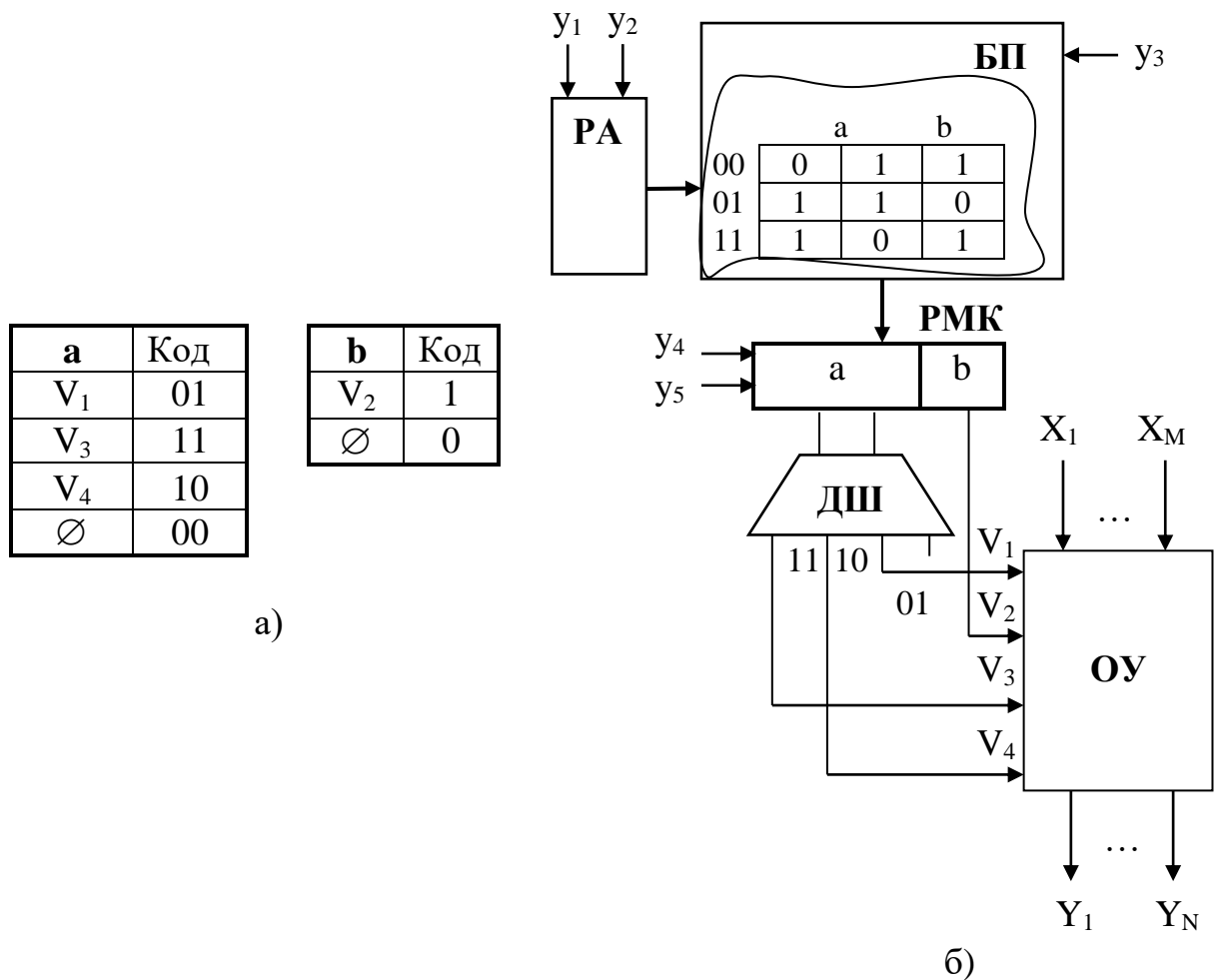


Рис. 3.3. Вертикальное кодирование микроопераций (а), устройство управление и его подключение к ОУ (б)

При построении УУ для преобразования кодов МО, задаваемых в полях микрокоманд, в управляющие сигналы используются дешифраторы микроопераций (рис. 3.3, б). В некоторых случаях дешифраторы микроопераций включают в состав ОУ (см., например, дешифраторы микроинструкций в ОУ из предыдущей лабораторной работы).

### 3.1.4. Формирование адреса микрокоманды с учетом логических условий

В общем случае код микрокоманды можно разделить на две части: операционную (ОЧ) и «адресно-управляющую» (АУЧ). Если ОЧ микрокоманды содержит коды микроопераций, то АУЧ определяет порядок выбора из блока памяти следующей для выполнения МК. В частности, выбор очередной микрокоманды может происходить в зависимости от значений логических условий, поступающих из операционного устройства. Рассмотрим формирование адреса микрокоманды с учетом логических условий на примере микропрограммы, приведенной на рис. 3.4, а (сигналы микроопераций в микропрограмме не показаны, так как формат ОЧ микрокоманды в примере не рассматривается).

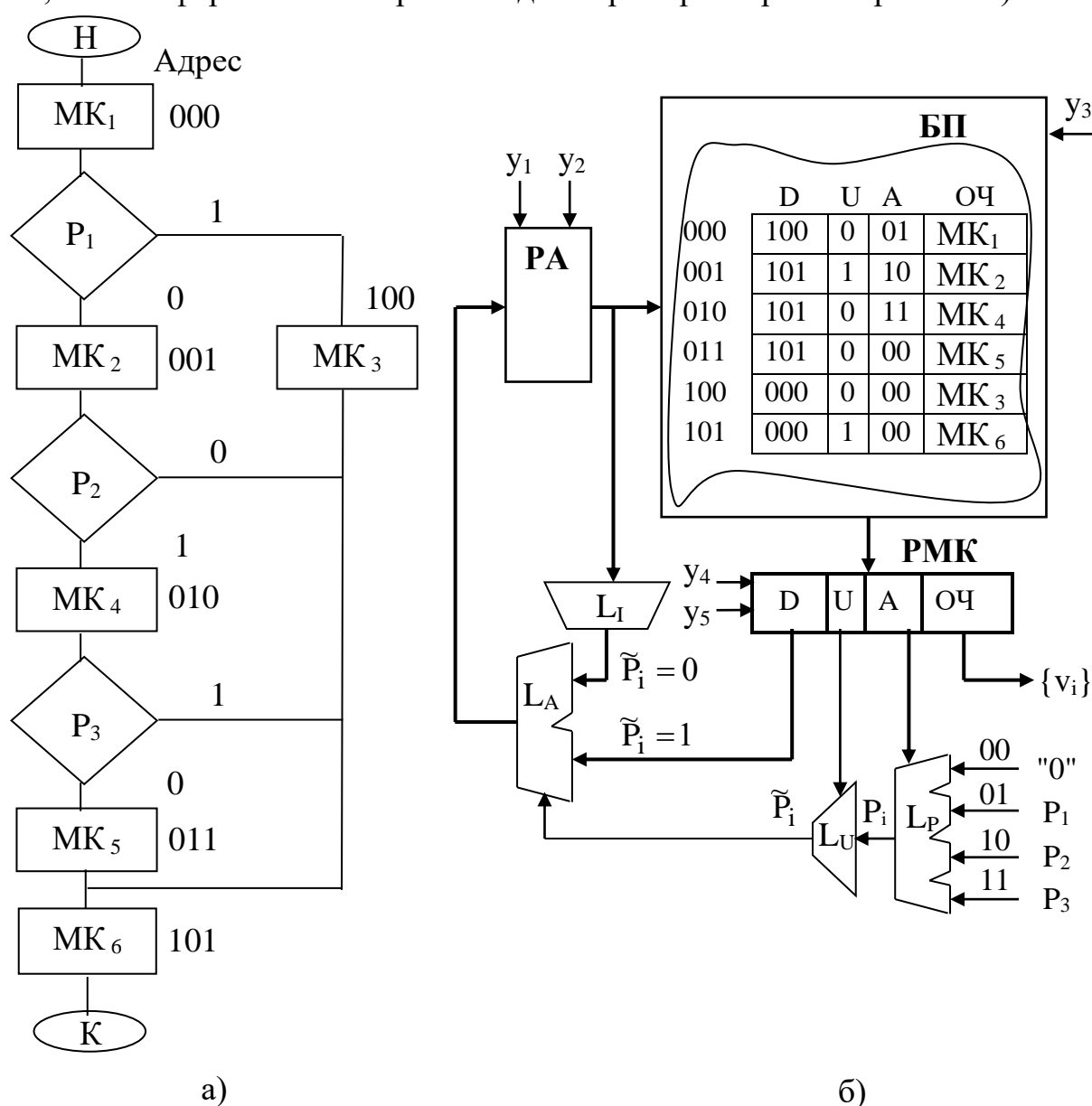


Рис. 3.4. Микропрограмма (а) и структура устройства управления (б):  
 $y_1 - \text{РА} := 0$ ;  $y_2 - \text{РА} := \text{РА} + 1$ ;  $y_3$  – чтение из БП;  $y_4 - \text{PMK} := 0$ ;  $y_5 - \text{PMK} := M[\text{РА}]$

В данном случае АУЧ микрокоманды включает три поля:

- D – адрес микрокоманды, к которой может быть переход;
- A – код выбираемого для анализа логического условия ( $P_i$ ): 00 – «0», 01 –  $P_1$ , 10 –  $P_2$ , 11 –  $P_3$ ;
- U – код инверсии выбранного логического условия: если  $U = 0$ , то значение выбранного логического условия не инвертируется  $\tilde{P}_i = P_i$ , в противном случае  $\tilde{P}_i = \overline{P_i}$ .

Структура УУ приведена на рис. 3.4, б. Выбор логического условия производится с помощью мультиплексора условий  $L_P$ , на управляющие вход которого поступает код A. Инвертор кода условия  $L_U$  позволяет (при  $U = 1$ ) в случае необходимости получить инверсию выбранного условия. Сигнал  $\tilde{P}_i$  с выхода инвертора кода условия поступает на управляющий вход мультиплексора адреса  $L_A$  и определяет адрес следующей микрокоманды. Если  $\tilde{P}_i = 0$ , то адрес следующей МК получается путем увеличения на единицу с помощью схемы инкремента  $L_I$  адреса текущей МК. Если же  $\tilde{P}_i = 1$ , то в регистр адреса записывается адрес перехода из РМК (поле D). Таблица прошивки для рассматриваемой микропрограммы приведена на рис. 3.4, б (первый столбец таблицы содержит адреса ячеек памяти, по которым размещаются микрокоманды).

### **3.1.5. Структура и рабочий цикл устройств управления**

УУ с «программируемой логикой» – это относительно сложное устройство, и к нему также применима модель дискретного преобразователя (рис. 3.5, а). В операционную часть УУ входят: РА, БП, РМК и блок формирования адреса (БФА). Управляющей частью УУ является блок синхронизации, пуска, останова (БСПО), который формирует управляющие сигналы ( $y_1, y_2, \dots, y_6$ ), необходимые для работы устройства. У БСПО могут быть внешние управляющие сигналы, например, «Пуск», «Стоп». Кроме того, он содержит внутренний триггер пуска-останова (ТПО), сброс которого останавливает работу УУ. Примером сигнала логического условия для БСПО может служить сигнал «s» – микрооперации останова, указы-

ваемой в коде микрокоманды. В виду простоты БСПО он строится как автомат с «жесткой» логикой.

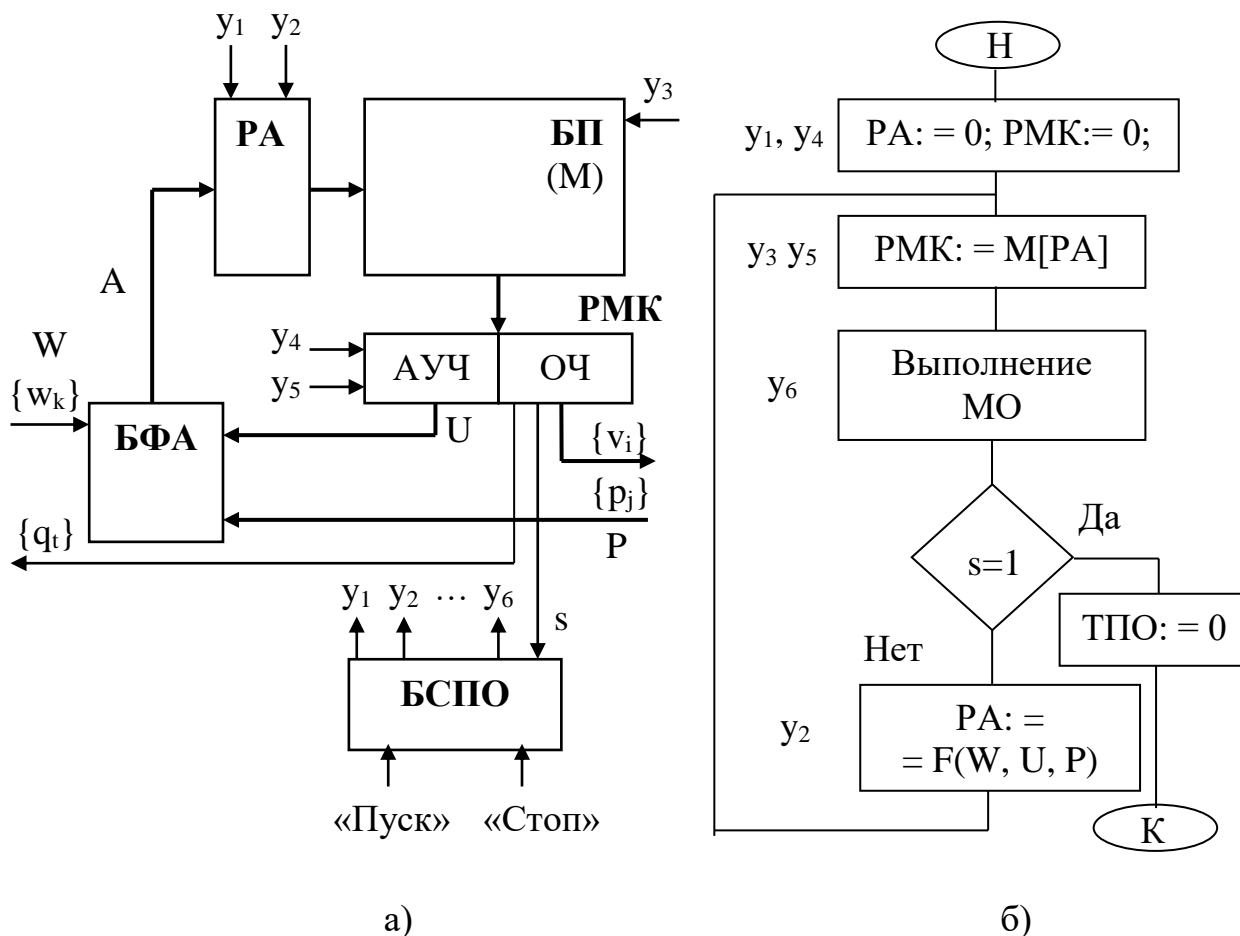


Рис. 3.5. Структура (а) и рабочий цикл (б) устройства управления:  
 $y_1 - PA := 0$ ;  $y_2 - PA := A, A = F(W, U, P)$ ;  $y_3 -$  чтение микрокоманды из БП ( $M[PA]$ );  
 $y_4 - PMK := 0$ ;  $y_5 - PMK := M[PA]$ ;  $y_6 -$  выполнение МО в ОУ;  
 $s -$  сигнал микропрограммного останова

Рабочий цикл УУ содержится в алгоритме работы БСПО. Граф-схема такого алгоритма приведена на рис. 3.5, б. Алгоритм описывает выборку микрокоманды из БП и формирование сигналов, соответствующих ей микроопераций. Иногда этот алгоритм называют «нанопрограммой».



### **3.2. Устройство управления с блоком управления последовательностью микрокоманд**

*Структура.* Структурная схема устройства управления приведена на рис. 3.6.

В состав устройства управления входят следующие блоки:

- преобразователь начального адреса (ПНА), преобразующий код операции в начальный адрес соответствующей микропрограммы;
- преобразователь адреса (ПА), преобразующий код прерывания в начальный адрес соответствующей микропрограммы обработки прерывания;
- блок управления последовательностью микрокоманд (БУПМ);
- мультиплексор кода условий (МПКУ), с помощью которого проверяемое логическое условие выбирается из множества логических условий, используемых в микропрограммах. На информационные входы МПКУ поступают сигналы Z, OVR, F7, C8 с регистра состояния операционного устройства;
- инвертор кода условия (ИКУ), обеспечивающий инверсию значения проверяемого условия;
- блок памяти микропрограмм;
- регистр микрокоманд (РМК).

Цикл работы устройства управления состоит из трех шагов: формирования в БУПМ адреса микрокоманды, считывания микрокоманды из блока памяти микропрограмм и выдачи внешних управляющих сигналов.

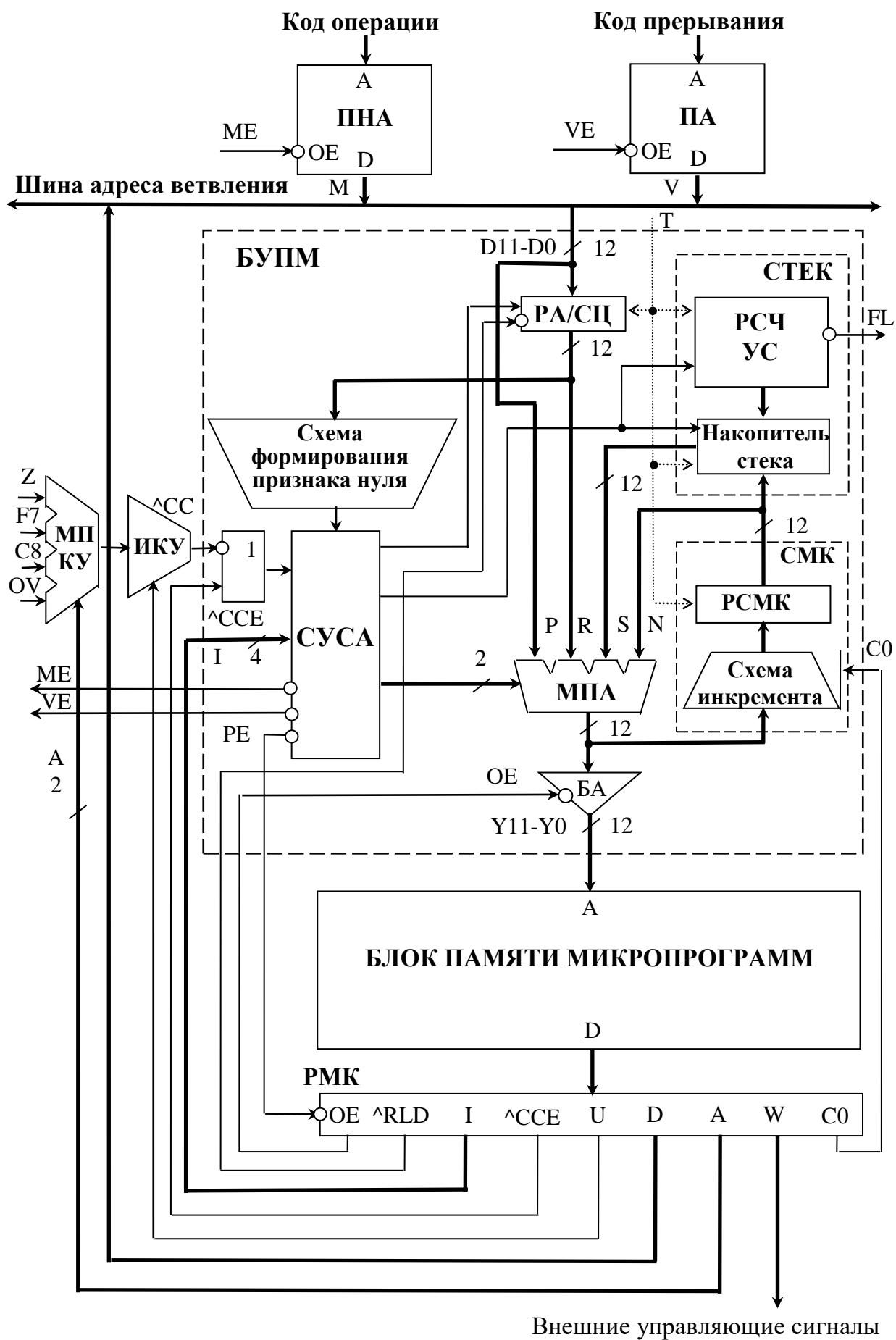


Рис. 3.6. Структурная схема устройства управления

Формат микрокоманды для устройства управления приведен на рис. 3.7.

| Шина данных | Код МИ  | Упр. признаками |      |   | Упр. сигналы |      |     |
|-------------|---------|-----------------|------|---|--------------|------|-----|
| D11...D0    | I3...I0 | A1...A0         | ^CCE | U | C0           | ^RLD | ^OE |

Рис. 3.7. Формат микрокоманды для устройства управления

*Блок управления последовательностью микрокоманд.* Основной функцией БУПМ является формирование адресов микрокоманд, хранящихся в памяти устройства управления. БУПМ позволяет формировать 12-разрядный адрес, выполнять условный или безусловный переход к любому адресу памяти емкостью 4096 слов, использовать стек глубиной пять ячеек. В состав БУПМ входят следующие основные узлы:

- регистр адреса (РА/СЦ), выполняющий также функции счетчика циклов в циклических микропрограммах; к выходам РА/СЦ подключена схема формирования признака нуля;
- стек, содержащий накопитель и реверсивный счетчик указателя стека;
- счетчик микрокоманд (СМК), состоящий из регистра и схемы инкремента;
- мультиплексор адреса (МПА) и буфер адреса (БА);
- схема управления следующим адресом (СУСА).

РА/СЦ состоит из 12 триггеров D-типа. Информация записывается в РА/СЦ с адресных входов D0...D11 по положительному фронту тактового сигнала Т при наличии напряжения низкого уровня на входе RLD или сигнала управления записью, поступающего с СУСА. В зависимости от выполняемой микроинструкции при наличии напряжения высокого уровня на входе RLD содержимое РА/СЦ может быть уменьшено на единицу по положительному фронту тактового сигнала. Если в РА/СЦ загружено число  $n$ , то при соответствующей микроинструкции цикл будет выполнен  $(n + 1)$  раз. Равенство нулю содержимого РА/СЦ используется в качестве условия перехода при выполнении некоторых микроинструкций.

СМК включает регистр счетчика микрокоманд, построенный на 12 триггерах D-типа, и схему приращения. СМК предназначен для хранения и преобразования адреса микрокоманды, поступающего с выхода МПА. Управление СМК осуществляется сигналом C0, причем, если  $C0 = 1$ , в регистр СМК во время следующего такта записывается адрес, представляющий собой текущий выходной адрес, увеличенный на единицу. Если  $C0 = 0$ , то адрес с МПА передается в регистр СМК немодифицированным, и во время следующего такта будет выполняться та же микрокоманда.

Стек предназначен для хранения адреса микрокоманды и обеспечивает выполнение перехода с возвратом при обращении к подмикропрограмме. Изменение указателя и запись адреса в один из пяти 12-разрядных регистров стека происходит по положительному фронту сигнала T. Указатель стека определяет регистр накопителя стека, содержащий информацию, записанную в стек последней. Управление стеком осуществляется сигналами на входах I0...I3, ^CC, ^CCE. Возможны следующие режимы работы стека:

А. Очистка (реверсивный счетчик указателя стека устанавливается в нулевое состояние).

Б. Хранение (значение указателя стека не меняется, производится считывание информации из регистра накопителя, определяемого указателем стека).

В. Запись в стек (значение указателя стека увеличивается на единицу, и производится запись информации в регистр накопителя по новому значению указателя стека, определяемому по правилу: 0-1-2-3-4-5).

Г. Чтение из стека (происходит считывание информации по текущему значению указателя стека, и значение указателя стека уменьшается на единицу по правилу: 5-4-3-2-1-0).

После того, как в стек будет записано пять слов, на выходе ^FL устанавливается сигнал низкого уровня. Любые дальнейшие операции записи не изменяют значения указателя стека и приводят к потере информации, расположенной в вершине стека. Выполнение операции чтения из пустого стека вызывает передачу на шину Y лишенных смысла данных, при этом указатель стека сохраняет нулевое значение.

МПА в зависимости от сигналов, вырабатываемых СУСА, выбирает адрес или из РА/СЦ, или из стека, или из СМК, или с входов D. БА передает адрес с МПА на выходы Y при наличии напряжения низкого уровня на входе  $\wedge OE$ . Напряжение высокого уровня на входе  $\wedge OE$  отключает выходы Y.

СУСА представляет собой комбинационный преобразователь на семь входов, предназначенный для преобразования внешних ( $I_0...I_3$ ,  $\wedge CC$ ,  $\wedge SSE$ ) и внутреннего (от схемы формирования признака нуля) управляющих сигналов БУПМ. Кроме того, СУСА вырабатывает три сигнала ( $\wedge ME$ ,  $\wedge PE$ ,  $\wedge VE$ ), которые используются для отпираания одного из трех внешних источников адреса.

БУПМ обеспечивает выполнение 16 микроинструкций по выбору адреса следующей микрокоманды. Код микроинструкции задается на входах  $I_0...I_3$ . Четыре микроинструкции определяют безусловный переход, и их выполнение не зависит от значения сигналов  $\wedge CC$  и  $\wedge SSE$ . Для остальных микроинструкций при  $\wedge CC = 0$  проверяемое условие считается выполненным, и адрес формируется так, как указано в названии функции микроинструкции, в противном случае, как правило, формируется адрес следующей по порядку микрокоманды. Наличие высокого уровня сигнала  $\wedge SSE$  приводит к безусловному выполнению этих микроинструкций, то есть адрес формируется так, как указано в названии функции микроинструкции, при любом значении логического условия.

Работа мультиплексора и инвертора кода условий описывается в табл. 3.1.

Таблица 3.1

Управление признаками

| A1 | A0 | U | $\wedge CC$  |
|----|----|---|--------------|
| 0  | 0  | 0 | Z            |
| 0  | 0  | 1 | $\wedge Z$   |
| 0  | 1  | 0 | F7           |
| 0  | 1  | 1 | $\wedge F7$  |
| 1  | 0  | 0 | OVR          |
| 1  | 0  | 1 | $\wedge OVR$ |
| 1  | 1  | 0 | C8           |
| 1  | 1  | 1 | $\wedge C8$  |

*Микроинструкции.* Набор микроинструкций приведен в табл. 3.2, где К – код микроинструкции; ОЧ – очистка стека; ХР – хранение; ЗП – запись; ЧТ – чтение из стека; -1 – уменьшение содержимого РА/СЦ на единицу; \* – если  $\overline{CCE} = 0$  и  $\overline{CC} = 1$ , то РА/СЦ находится в режиме хранения, иначе выполняется запись; Х – произвольное состояние РА/СЦ.

Таблица 3.2

Микроинструкции

| К | Мне-мо-ника | Функция   | РА/<br>/СЦ<br>до<br>опера-<br>ции | Результат микроинструкции               |      |   |      |                        |
|---|-------------|---|-----------------------------------|---|------|---|------|------------------------|
|   |             |   |                                   | $\overline{CCE} = 0; \overline{CC} = 1$ |      | $\overline{CCE} = 1 \vee \overline{CC} = 0$ |      | РА/<br>/СЦ,<br>выход   |
|   |             |   |                                   | У                                       | СТЕК | У   | СТЕК |                        |
| 0 | JZ          | Переход по нулевому адресу                              | Х                                 | 0                                       | ОЧ   | 0   | ОЧ   | ХР, $\overline{PE}$    |
| 1 | CJS         | Условный переход к подмикро-программе                   | Х                                 | СМК                                     | ХР   | D   | ЗП   | ХР,<br>$\overline{PE}$ |
| 2 | JMAP        | Переход по адресу из преобразова-теля начального адреса | Х                                 | D                                       | ХР   | D   | ХР   | ХР,<br>$\overline{ME}$ |
| 3 | CJP         | Условный переход по адресу из РМК                       | Х                                 | СМК                                     | ХР   | D   | ХР   | ХР,<br>$\overline{PE}$ |
| 4 | PUSH        | Загрузка СТЕКА и условная загрузка РА/СЦ                | Х                                 | СМК                                     | ЗП   | СМК   | ЗП   | *,<br>$\overline{PE}$  |
| 5 | JSRP        | Условный переход к одной из двух подмик-ропрограмм      | Х                                 | РА/СЦ                                   | ЗП   | D   | ЗП   | ХР,<br>$\overline{PE}$ |
| 6 | CJV         | Условный переход по век-торному адресу                  | Х                                 | СМК                                     | ХР   | D   | ХР   | ХР,<br>$\overline{VE}$ |

|   |      |   |                  |                 |              |                 |              |  |
|---|------|---|------------------|-----------------|--------------|-----------------|--------------|--|
| 7 | JRP  | Условный переход по адресу, выбираемому из РА/СЦ или РМК  | X                | РА/СЦ           | XP           | D               | XP           | XP,<br>$\overline{PE}$                           |
| 8 | RFCT | Повторение цикла, если (РА/СЦ) < > 0<br><br>= 0           | < > 0<br><br>= 0 | СТЕК<br><br>СМК | XP<br><br>ЧТ | СТЕК<br><br>СМК | XP<br><br>ЧТ | -1,<br>$\overline{PE}$<br>XP,<br>$\overline{PE}$ |
| 9 | RPCT | Повторение адреса из РМК пока (РА/СЦ) < > 0               | < > 0<br><br>= 0 | D<br><br>СМК    | XP<br><br>XP | D<br><br>СМК    | XP<br><br>XP | -1,<br>$\overline{PE}$<br>XP,<br>$\overline{PE}$ |
| A | CRPN | Условный возврат из подмикропрограммы                     | X                | СМК             | XP           | СТЕК            | ЧТ           | XP,<br>$\overline{PE}$                           |
| B | CJPP | Условный переход по адресу из РМК и выталкивание из СТЕКА | X                | СМК             | XP           | D               | ЧТ           | XP,<br>$\overline{PE}$                           |
| C | LDCT | Загрузка счетчика и продолжение                           | X                | СМК             | XP           | СМК             | XP           | ЗП,<br>$\overline{PE}$                           |
| D | LOOP | Проверка условия окончания цикла                          | X                | СТЕК            | XP           | СМК             | ЧТ           | XP,<br>$\overline{PE}$                           |
| E | CONT | Продолжить  | X                | СМК             | XP           | СМК             | XP           | XP,<br>$\overline{PE}$                           |
| F | TWB  | Переход по одному из трех направлений                     | < > 0<br><br>= 0 | СТЕК<br><br>D   | XP<br><br>ЧТ | СМК<br><br>СМК  | ЧТ<br><br>ЧТ | -1,<br>$\overline{PE}$<br>XP,<br>$\overline{PE}$ |

Рассмотрим выполнение микроинструкций более подробно:

а) JZ (JUMP TO ADDRESS ZERO) (код 0) – переход к микрокоманде с нулевым адресом. При этом происходит очистка стека.

б) CJS (CONDITIONAL JUMP TO SUBROUTINE) (код 1) – условный переход к подмикропрограмме.

При выполнении условия ( $\wedge CCE = 1$  или  $\wedge CC = 0$ ) осуществляется переход к адресу подмикропрограммы, принимаемому из регистра микрокоманд (РМК) через шину D, точка возврата запоминается в стеке. Если условие не выполняется ( $\wedge CCE = 0$  и  $\wedge CC = 1$ ), то выполняется следующая по порядку микрокоманда, адрес которой поступает из СМК. Адрес Р из РМК устанавливается на входе D, адрес N+1 запоминается в стеке.

в) JMAP (JUMP TO ADDRESS AT MAPPING FROM OUTPUT) (код 2) – переход по адресу из преобразователя начального адреса.

Выполняется безусловный переход к адресу, принимаемому из преобразователя начального адреса через шину D. Реализуется путем формирования отпирающего сигнала  $\wedge ME = 0$  на выходе БУПМ, соединенном с управляющим входом преобразователя. Указанная микроинструкция обычно завершает фазу вызова команды в блок микропрограммного управления, после чего начинается выполнение микропрограммы, соответствующей вызванной команде. Адрес М с выхода преобразователя начального адреса устанавливается на входе D.

г) CJP (CONDITIONAL JUMP TO ADDRESS IN PIPELINE REGISTER) (код 3) – условный переход по адресу из РМК.

При выполнении условия ( $\wedge CCE = 1$  или  $\wedge CC = 0$ ) осуществляется переход по адресу, принимаемому из РМК через шину D. В отличие от микроинструкции CJS, операция со стеком не производится. Если условие не выполняется ( $\wedge CCE = 0$  и  $\wedge CC = 1$ ), то осуществляется переход к следующей по порядку микрокоманде, адрес которой поступает из СМК. Адрес Р из РМК устанавливается на входе D.

д) PUSH (PUSH STACK AND CONDITIONALLY LOAD COUNTER) (код 4) – засылка в стек и условная загрузка счетчика.

При выполнении текущей микрокоманды адрес следующей засылается в стек. Одновременно с этим, если условие выполняется ( $\wedge CCE = 1$  или  $\wedge CC = 0$ ), осуществляется загрузка счетчика. Если условие не выполняется ( $\wedge CCE = 0$  и  $\wedge CC = 1$ ), то загрузка не производится. Данные для загрузки РА/СЦ будут поступать из РМК, так как  $\wedge PE = 0$ . Естественный порядок следования микрокоманд не нарушается, и переход выполняется к следующей по порядку микрокоманде, адрес которой поступает из СМК.



е) JSRP (JUMP TO SUBROUTINE WITH STARTING ADDRESS CONDITIONALLY SELECTED FROM R-REGISTER OR PIPELINE REGISTER) (код 5) – условный переход к одной из двух подмикропрограмм.

При выполнении микроинструкции адрес возврата из СМК засылается в стек и выполняется переход к одной из двух подмикропрограмм, в зависимости от значения условия. Если условие выполняется ( $\wedge\text{CCE} = 1$  или  $\wedge\text{CC} = 0$ ), то осуществляется переход к адресу, содержащемуся в РМК, если условие не выполняется – к адресу, источником которого является РА/СЦ БУПМ. Адрес R должен быть предварительно загружен в РА/СЦ. Адрес  $N + 1$  запоминается в стеке. Адрес P из РМК устанавливается на входе D.

ж) CJV (CONDITIONAL JUMP TO VECTOR ADDRESS) (код 6) – условный переход на адрес вектора.

При выполнении условия ( $\wedge\text{CCE} = 1$  или  $\wedge\text{CC} = 0$ ) осуществляется переход на адрес вектора, источником которого является преобразователь адреса. Реализуется он путем формирования на выходе  $\wedge\text{VE}$  БУПМ отпирающего сигнала  $\wedge\text{VE} = 0$ , соединенном с управляющим входом преобразователя адреса. Если условие не выполняется ( $\wedge\text{CCE} = 0$  и  $\wedge\text{CC} = 1$ ), то производится переход к следующей по порядку микрокоманде, адрес которой принимается из СМК. Данная микроинструкция может быть использована, например, при вызове подмикропрограмм обслуживания прерывания или прямого доступа в память. Адрес V из преобразователя адреса устанавливается на входе D.

з) JRP (JUMP TO ADDRESS CONDITIONALLY SELECTED FROM R-REGISTER OR PIPELINE REGISTER) (код 7) – переход на адрес, условно выбираемый из РА/СЦ, либо из РМК.

При выполнении условия ( $\wedge\text{CCE} = 1$  или  $\wedge\text{CC} = 0$ ) осуществляется переход на адрес, принимаемый из РМК на шину D. Если условие не выполняется ( $\wedge\text{CCE} = 0$  и  $\wedge\text{CC} = 1$ ), то осуществляется переход на адрес, источником которого является РА/СЦ БУПМ. В отличие от аналогичной микроинструкции 5, операция со стеком не производится. Адрес R должен быть предварительно загружен в РА/СЦ. Адрес P из РМК устанавливается на входе D.

и) RFCT (REPEAT LOOP IF COUNTER IS NOT EQUAL TO ZERO) (код 8) – повторить цикл, если содержимое счетчика не равно нулю.

Данная микроинструкция используется для организации выполнения одной или нескольких микрокоманд заданное число раз. Для этого необходимо предварительно записать в стек адрес возврата и число на единицу меньше, чем требуемое число циклов, в РА/СЦ БУПМ. Это реализуется с помощью микроинструкции 4 (PUSH).

При выполнении инструкции RFCT проверяется содержимое РА/СЦ. Если оно не равно нулю, то выполняется цикл. Это осуществляется переходом к адресу возврата, который извлекается из вершины стека, одновременно выполняется уменьшение на единицу содержимого РА/СЦ. Если содержимое РА/СЦ равно нулю, то выполняется переход к следующей по порядку микрокоманде, адрес которой принимается из СМК, при этом адрес возврата выталкивается из стека.

В общем случае адрес S, записанный предварительно в стек, может быть не адресом возврата к началу цикла, а адресом какой-либо произвольной ветви в микропрограммной памяти, тогда рассматриваемая микроинструкция может интерпретироваться как условный переход на адрес из стека по содержимому РА/СЦ.

Содержимое верхней ячейки стека выталкивается из стека, адрес должен быть предварительно запомнен в стеке.

Если  $S > N$ , то получится не цикл, а условный переход к адресу из стека, если содержимое счетчика не равно нулю (RFCT).

к) RPCT (REPEAT PIPELINE ADDRESS IF COUNTER IS NOT EQUAL TO ZERO) (код 9) – повторить цикл из РМК, пока содержимое счетчика не станет равным нулю.

Данная микроинструкция аналогична микроинструкции 8, за исключением того, что адрес ветвления извлекается из РМК (через шину D), а не из стека. Содержимое стека не используется.

Если содержимое РА/СЦ не равно нулю, то осуществляется переход к адресу возврата, принимаемому из РМК через шину D. Одновременно происходит уменьшение содержимого РА/СЦ на единицу. Если содержимое РА/СЦ равно нулю, то выполняется переход к следующей по порядку микрокоманде, адрес которой принимается из СМК.

Эта микроинструкция может быть интерпретирована как расширение стека на одно слово, в том смысле, что при использовании данной микроинструк-

ции цикл может быть выполнен с использованием счетчика, даже если глубина вложенности ячеек достигла пяти.

Содержимое верхней ячейки стека выталкивается из стека, адрес Р из РМК устанавливается на входе D.

л) CRTN (CONDITIONAL RETURN FROM SUBROUTINE) (код 10) – условный возврат из подмикропрограммы.

Если условие выполняется ( $\wedge\text{CCE} = 1$  или  $\wedge\text{CC} = 0$ ), то осуществляется переход по адресу, принимаемому из вершины стека, после чего содержимое этой ячейки выталкивается из стека (поскольку после перехода адрес возврата из подмикропрограммы уже не нужен). Если условие не выполняется ( $\wedge\text{CCE} = 0$  и  $\wedge\text{CC} = 1$ ), то осуществляется переход к следующей по порядку микрокоманде, адрес которой принимается из СМК.

Адрес N+1 должен быть предварительно запомнен в стеке.

м) CJPP (CONDITIONAL JUMP TO PIPELINE ADDRESS AND POP STACK) (код 11) – условный переход к адресу из РМК и выталкивание из стека.

При выполнении условия ( $\wedge\text{CCE} = 1$  или  $\wedge\text{CC} = 0$ ) осуществляется переход к адресу, принимаемому из РМК через шину D. Если условие не выполняется ( $\wedge\text{CCE} = 0$  и  $\wedge\text{CC} = 1$ ), то осуществляется переход к следующей по порядку микрокоманде, адрес которой поступает из СМК.

Данная инструкция может быть использована для выхода из тела цикла и подмикропрограммы до их окончания. При выходе из тела цикла потребность в сохранении адреса возврата исчезает, поэтому он может быть вытолкнут из стека. Микроинструкция позволяет на аппаратно-микропрограммном уровне компактно реализовать таблицы переходов. Для этой цели необходимо в теле цикла последовательно расположить несколько микроинструкций.

н) LDCT (LOAD COUNTER AND CONTINUE) (код 12) – загрузка счетчика и продолжение.

Микроинструкция безусловная, производится загрузка РА/СЦ информацией, поступающей из РМК через шину D, так как  $\wedge\text{PE} = 0$ . Одновременно выполняется переход к следующей по порядку микрокоманде, адрес которой поступает из СМК.

о) LOOP (TEST END OF LOOP) (код 13) – контроль конца цикла.

Данная микроинструкция обеспечивает возможность выхода из цикла по условию. Если условие выполняется ( $\wedge CCE = 1$  или  $\wedge CC = 0$ ), то выполняется следующая по порядку микрокоманда, адрес которой принимается из СМК. Одновременно производится выталкивание из стека. Если условие не выполняется ( $\wedge CCE = 0$  и  $\wedge CC = 1$ ), то осуществляется переход по адресу из вершины стека.

В общем случае адрес  $S$  может быть адресом какой-либо произвольной ветви микропрограммной памяти, тогда рассматриваемая микроинструкция становится микроинструкцией условного перехода к адресу, извлекаемому из вершины стека.

Содержимое верхней ячейки стека выталкивается из стека, адрес  $S$  должен быть предварительно запомнен в стеке.

п) CONT (CONTINUE TO NEXT ADDRESS) (код 14) – продолжить.

Выполняется безусловный переход к следующей микрокоманде, адрес которой поступает из СМК. Никаких других операций не выполняется.

р) TWB (THREE-WAY BRANSH) (код 15) – ветвление на три направления.

Данная микроинструкция осуществляет выбор одного из источников адреса следующей микрокоманды (стека, СМК или РМК через шину  $D$ ) в зависимости от внешнего условия и от содержимого  $PA/CI$ . Для этого необходимо предварительно записать адрес возврата в стек, а число на единицу меньше, чем требуемое число циклов в  $PA/CI$ .

Если содержимое  $PA/CI$  не равно нулю, то при выполнении условия ( $\wedge CCE = 1$  или  $\wedge CC = 0$ ) осуществляется переход к следующей по порядку микрокоманде, адрес которой принимается из СМК, затем происходит выталкивание из стека и уменьшение содержимого  $PA/CI$  на единицу. Если содержимое  $PA/CI$  не равно нулю и условие не выполняется ( $\wedge CCE = 0$  и  $\wedge CC = 1$ ), то осуществляется переход к микрокоманде, адрес которой принимается из стека и происходит уменьшение содержимого  $PA/CI$  на единицу. При этом адрес возврата в вершине стека сохраняется.

Если содержимое  $PA/CI$  равно нулю, то при выполнении условия ( $\wedge CCE = 1$  или  $\wedge CC = 0$ ) осуществляется переход к следующей по порядку микрокоманде, а в противном случае ( $\wedge CCE = 0$  и  $\wedge CC = 1$ ) к микрокоманде, адрес которой прини-

мается из РМК через шину D. При этом независимо от выполнения условия производится выталкивание адреса возврата из стека, а содержимое РА/СЦ не меняется.

Содержимое верхней ячейки стека выталкивается из стека. Адрес S должен быть предварительно запомнен в стеке. Адрес Р из РМК устанавливается на входе D.

Использование данной микроинструкции полезно при выполнении многих машинных команд поиска в памяти, завершающегося нахождением заданного содержимого или достижением предела поиска. Кроме того, она полезна в операциях над полями переменной длины, в которых содержимое необработанной части поля начинается с нулевого значения, при поиске по ключу в дисковых контроллерах, использующих диски переменной емкости, а также при нормализации чисел с плавающей запятой и других.

### **3.3. Лабораторная работа № 2**

Выполнение лабораторной работы складывается из домашней подготовки, экспериментальных исследований и оформления отчета. Домашняя подготовка включает разработку микропрограммы для УУ, которая должна обеспечивать выполнение операции, указанной в задании на лабораторную работу № 2. При разработке микропрограммы для УУ операционные части микрокоманд не рассматриваются.

Экспериментальные исследования заключаются во вводе, отладке и выполнении разработанной микропрограммы. Отчет по лабораторной работе должен содержать: титульный лист, задание, граф-схему микропрограммы, диаграмму распределения ячеек блока памяти микропрограмм, текст отлаженной микропрограммы с комментариями.

Рассмотрим выполнение лабораторной работы на примере задания № 3.

#### **3.3.1. Задание № 2**

Разработать микропрограмму для УУ, обеспечивающую выполнение в ОУ операции деления целых положительных чисел нацело, в соответствии с заданием № 2, приведенным в разделе 2.

### 3.3.2. Разработка микропрограммы и распределение памяти микропрограмм

Граф-схема микропрограммы деления для операционного устройства представлена на рис. 2.7. При разработке микропрограммы для УУ может оказаться целесообразным преобразование исходной микропрограммы с учетом особенностей способов адресации микрокоманд, применяемых в БУПМ. В частности, возможно значительное упрощение циклического участка микропрограммы с заданным числом повторений цикла за счет использования счетчика циклов, расположенного в БУПМ. При выборе микроинструкций БУПМ и распределении памяти микропрограмм удобно воспользоваться специальной диаграммой (рис. 3.8).

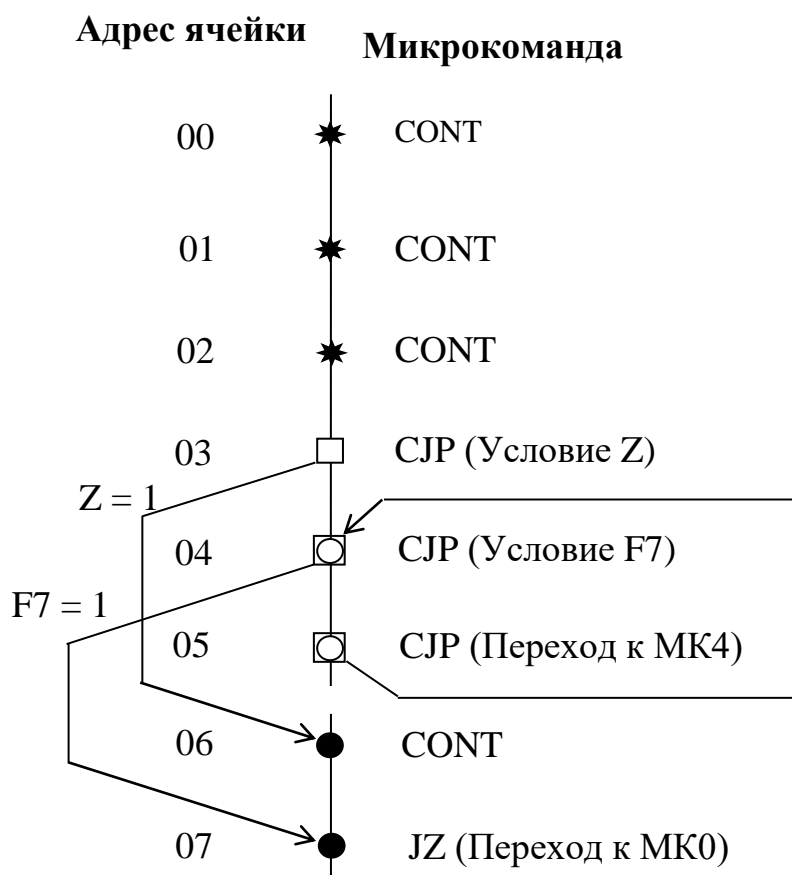


Рис. 3.8. Диаграмма распределения памяти микропрограмм:

- ★ – произвольная микрокоманда;
- – микрокоманда перехода;
- – микрокоманда, к которой осуществляется переход при выполнении проверяемого условия;
- – микрокоманда, к которой осуществляется переход при невыполнении проверяемого условия

### 3.3.3. Кодирование микропрограммы

Кодирование микропрограммы для УУ выполняется аналогично кодированию микропрограммы для ОУ. Текст микропрограммы в двоичных кодах с комментариями приведен в табл. 3.3. Начинается микропрограмма с линейного участка (МК 00, 01, 02), затем идет микрокоманда условного перехода по признаку нуля (МК 03). Если признак нуля  $Z = 1$ , то осуществляется переход к МК 6, иначе выполняется следующая по порядку микрокоманда (МК 04), которая является микрокомандой перехода по признаку знака (F7). Если  $F7 = 1$ , то осуществляется переход к МК 07, иначе выполняется следующая по порядку микрокоманда (МК 05), которая является микрокомандой безусловного перехода к МК 03. После МК 06 выполняется МК 07, в которой для удобства отладки микропрограммы предусмотрен безусловный переход на МК 00.

Таблица 3.3

Микропрограмма деления чисел нацело

| МК                  | Управляющая часть |       |                 |   |                  |              |                  |                 |
|---------------------|-------------------|-------|-----------------|---|------------------|--------------|------------------|-----------------|
| №                   | Шина              | МИ    | Упр. признаками |   |                  | Упр. сигналы |                  |                 |
| N                   | D11-D0            | I3-I0 | A               | U | $\overline{CCE}$ | $C0$         | $\overline{RLD}$ | $\overline{OE}$ |
| 00                  | 0000              | 1110  | 00              | 0 | 0                | 1            | 1                | 0               |
| CONT                |                   |       |                 |   |                  |              |                  |                 |
| 01                  | 0000              | 1110  | 00              | 0 | 0                | 1            | 1                | 0               |
| CONT                |                   |       |                 |   |                  |              |                  |                 |
| 02                  | 0000              | 1110  | 00              | 0 | 0                | 1            | 1                | 0               |
| CONT                |                   |       |                 |   |                  |              |                  |                 |
| 03                  | 0006              | 0011  | 00              | 1 | 0                | 1            | 1                | 0               |
| CJP (Условие Z)     |                   |       |                 |   |                  |              |                  |                 |
| 04                  | 0007              | 0011  | 01              | 1 | 0                | 1            | 1                | 0               |
| CJP (Условие F7)    |                   |       |                 |   |                  |              |                  |                 |
| 05                  | 0004              | 0011  | 00              | 0 | 1                | 1            | 1                | 0               |
| CJP (Переход к МК4) |                   |       |                 |   |                  |              |                  |                 |
| 06                  | 0000              | 1110  | 00              | 0 | 0                | 1            | 1                | 0               |
| CONT                |                   |       |                 |   |                  |              |                  |                 |
| 07                  | 0000              | 0000  | 00              | 0 | 0                | 1            | 1                | 0               |
| JZ (Переход к МК0)  |                   |       |                 |   |                  |              |                  |                 |

### 3.3.4. Ввод и отладка микропрограммы

В программе «Имитатор устройства управления» создается файл с расширением «.мии», представляющий собой двоичный код микропрограммы для устройства управления. Для ввода и отладки микропрограмм в программе предусмотрены специальные окна и команды (рис. 3.9, 3.10), подобные тем, что подробно рассмотрены в разделе 1.3.

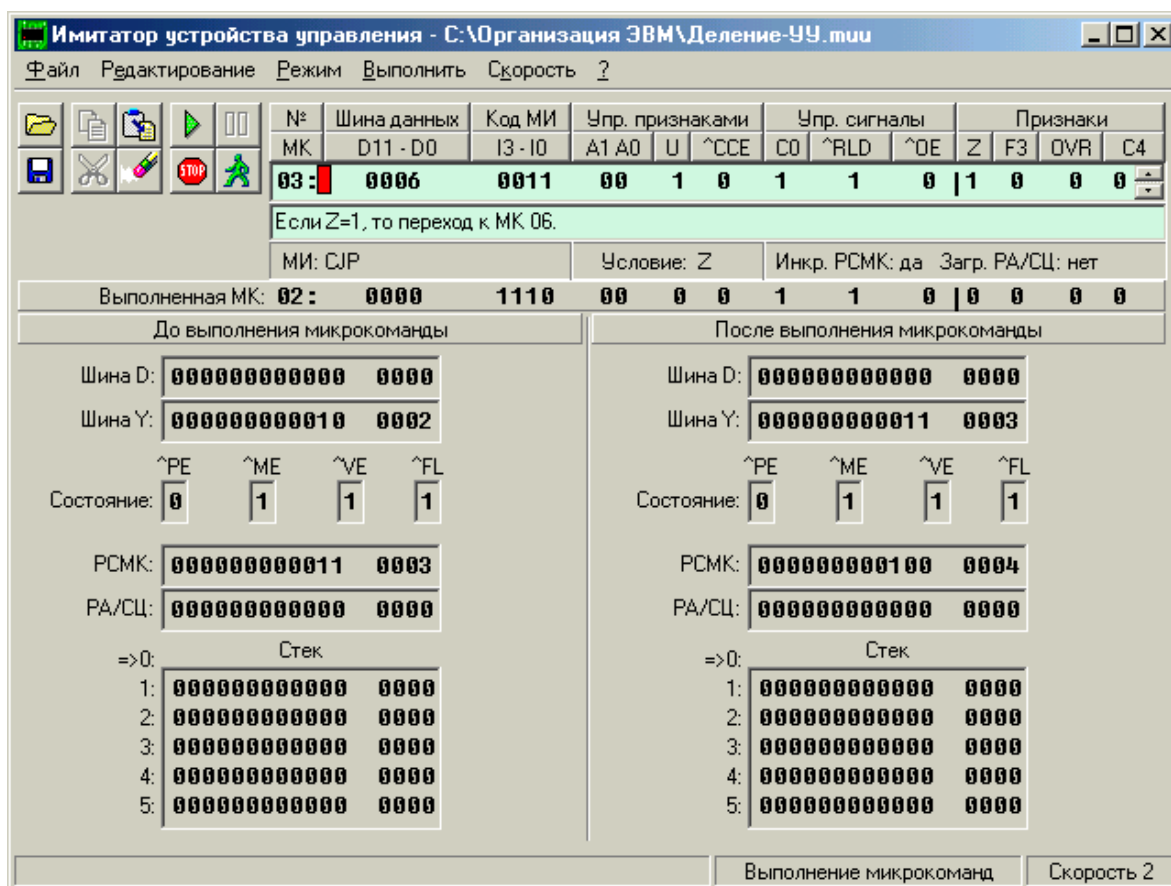


Рис. 3.9. Окно выполнения микрокоманд



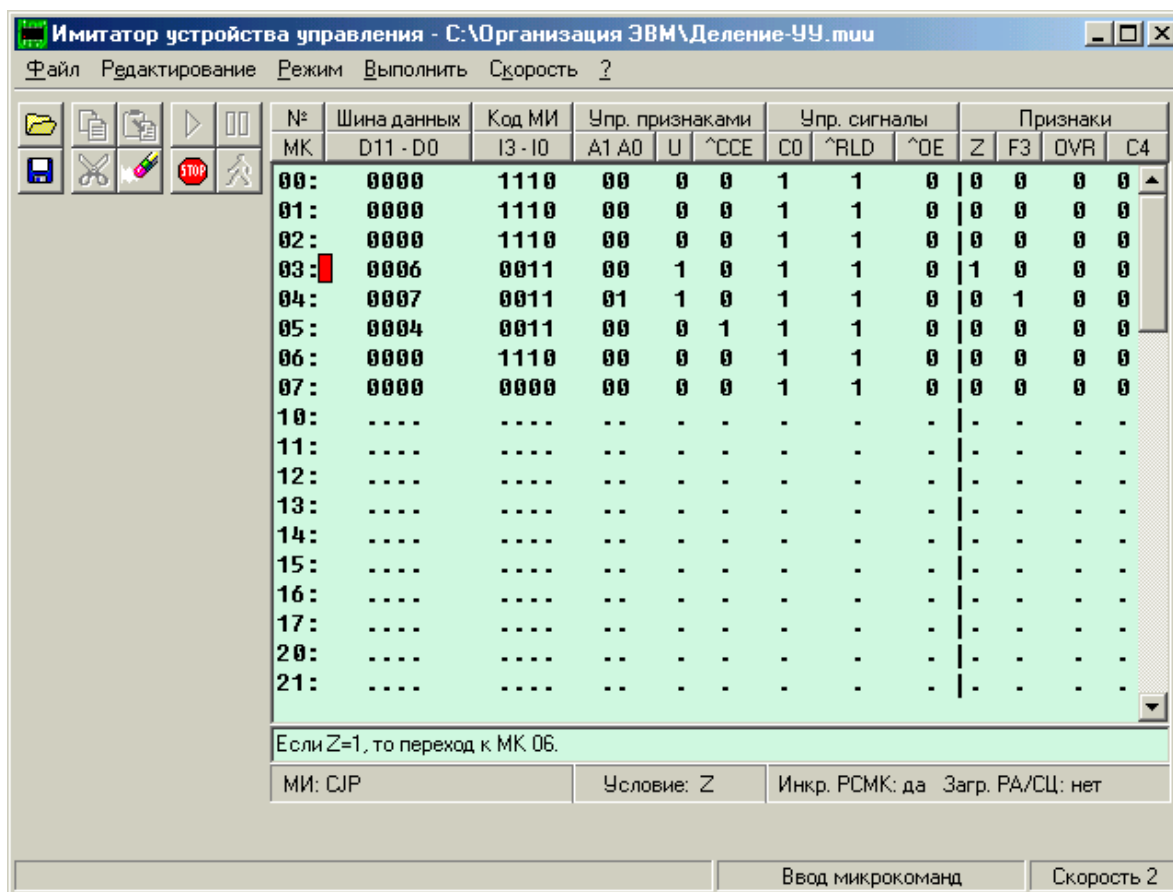


Рис. 3.10. Окно ввода микрокоманд

В данной лабораторной работе построение ОУ не рассматривается. При разработке микропрограммы используется упрощенный формат микрокоманды, которая содержит только поля, необходимые для управления исследуемым УУ. Поэтому выдача в УУ значений логических условий, формируемых в ОУ, возлагается на пользователя. Вводимые пользователем значения логических условий отображаются в специальном служебном поле микрокоманды «Признаки».

Пользователь может выбирать и выполнять микрокоманды с помощью кнопок панели инструментов и команд управляющего меню главного окна программы «Имитатор устройства управления». В программе предусмотрено пошаговое и автоматическое выполнение микропрограммы. Порядок выполнения микрокоманд определяется заданными в них микроинструкциями переходов с учетом введенных пользователем значений логических условий. С помощью группы команд «Скорость» может быть задана высокая, средняя или низкая скорость выполнения микропрограммы. Кроме того, используя группу команд «Выполнить», можно задавать точки останова.

### 3.4. Контрольные вопросы

#### *Основные положения*

1. Из каких шагов складывается рабочий цикл устройства управления (УУ)?
2. Кодирование микроопераций в операционной части микрокоманды УУ, при котором каждой микрооперации соответствует свой разряд, называется:
  - горизонтальным;
  - вертикальным;
  - смешанным;
  - унитарным.
3. Число полей в операционной части микрокоманды УУ при смешанном кодировании микроопераций, как правило, равно:
  - максимальному числу одновременно выполняемых микроопераций;
  - числу различных микроопераций;
  - числу микрокоманд, в которых выполняется более одной микрооперации;
  - минимальному числу одновременно выполняемых микроопераций.
4. Включение поля кода инверсии значения логического условия в состав микрокоманды при кодировании микропрограммы позволяет:
  - исключить дополнительные микрокоманды перехода при размещении микропрограммы в памяти микропрограмм;
  - превратить условный переход в безусловный;
  - исключить логические условия;
  - использовать стек адресов перехода по условиям.
5. Содержимое ячейки ПЗУ, используемого в качестве преобразователя начального адреса при дешифрации кодов операций, по адресу равному дешифрируемому коду операции представляет собой:
  - начальный адрес микропрограммы операции;
  - начальный адрес кода операции;
  - код операции;
  - первую микрокоманду микропрограммы операции.

6. Микропрограмма, состоящая из 200 микрокоманд и содержащая 10 логических условий и 10 различных микроопераций, при размещении в памяти микропрограмм использующая формат микрокоманды, который включает поля: адреса следующей микрокоманды, логического условия, операционной части с горизонтальным кодированием микроопераций имеет объем \_\_\_\_\_ бит.

7. Линейная микропрограмма, содержащая 6 различных микроопераций (V) и состоящая из 3 микрокоманд (M):  $M_1 = \langle V_1, V_2, V_3 \rangle$ ,  $M_2 = \langle V_1, V_3, V_4 \rangle$ ,  $M_3 = \langle V_1, V_3, V_5 \rangle$  при смешанном кодировании микроопераций имеет объем \_\_\_\_\_ бит.

*Исследуемое устройство*

8. Перечислите источники адреса микрокоманды в УУ.
9. Как в УУ осуществляется переход по коду операции?
10. С какой целью в УУ предусмотрена возможность инвертирования кода логического условия?
11. Как в УУ выполняется обращение к подмикропрограммам?
12. В чем состоит преимущество включения в состав УУ счетчика циклов микропрограмм?

## 4. ВЫЧИСЛИТЕЛЬНОЕ УСТРОЙСТВО С ЗАПОМИНАЮЩИМ УСТРОЙСТВОМ

### 4.1. Основные положения

#### 4.1.1. Функционально-структурные параметры вычислительного устройства

После независимого исследования составных частей ВУ: операционного устройства и устройства управления можно перейти к исследованию ВУ как единого устройства. ВУ – это устройство, способное выполнять управляющие и вычислительные алгоритмы, представленные в виде микропрограмм. ВУ можно рассматривать как процессор с микропрограммным уровнем

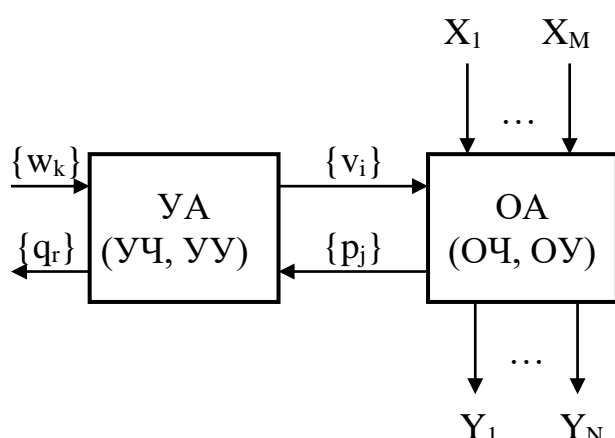


Рис. 4.1. Модель ВУ

управления. Работа ВУ может быть описана моделью дискретного преобразователя, представляющего собой композицию двух конечных автоматов: операционного и управляющего (рис. 4.1). ВУ имеет информационные входы ( $X_1, \dots, X_M$ ) и выходы ( $Y_1, \dots, Y_N$ ). В ВУ поступают управляющие сигналы  $\{w_k\}$  из устройства более высокого уровня управления, для которого ВУ формирует осведомительные сигналы  $\{q_r\}$ . Рабочий цикл ВУ совпадает с рабочим циклом УУ и сводится к выполнению одной микрокоманды. Рабочий цикл ВУ при использовании в качестве управляющей части УУПЛ приведен на рис. 4.5, б и включает в себя формирование адреса микрокоманды, выборку микрокоманды из блока памяти микропрограмм в УУПЛ, выполнение микроопераций выбранной микрокоманды и формирование значений логических условий в ОУ.

ВУ характеризуют следующие функционально-структурные параметры: набор операций; формы представления и форматы данных; число и разрядность внешних входов и выходов; число и разрядность внутренних регистров; набор

внутренних МО и ЛУ; внешние управляющие и осведомительные сигналы; способ построения операционного устройства (с закрепленными или общими микрооперациями); способ построения устройства управления (с жесткой или программируемой логикой). К числу основных временных характеристик ВУ относятся: время выполнения микрокоманды, время выполнения операции, быстродействие.

#### 4.1.2. Два вида параллельной обработки информации

Можно выделить два вида параллельной обработки информации: многоэлементная и многостадийная (конвейерная).

При многоэлементной обработке для вычисления каждой функции (операции)  $y_i = F_i(x_i)$  ( $i = 1, 2, \dots, N$ ) используется отдельный блок  $F_i$ , и  $N$  одинаковых функций  $F_1, F_2, \dots, F_N$  вычисляется одновременно за время  $T$  (рис. 4.2 а).

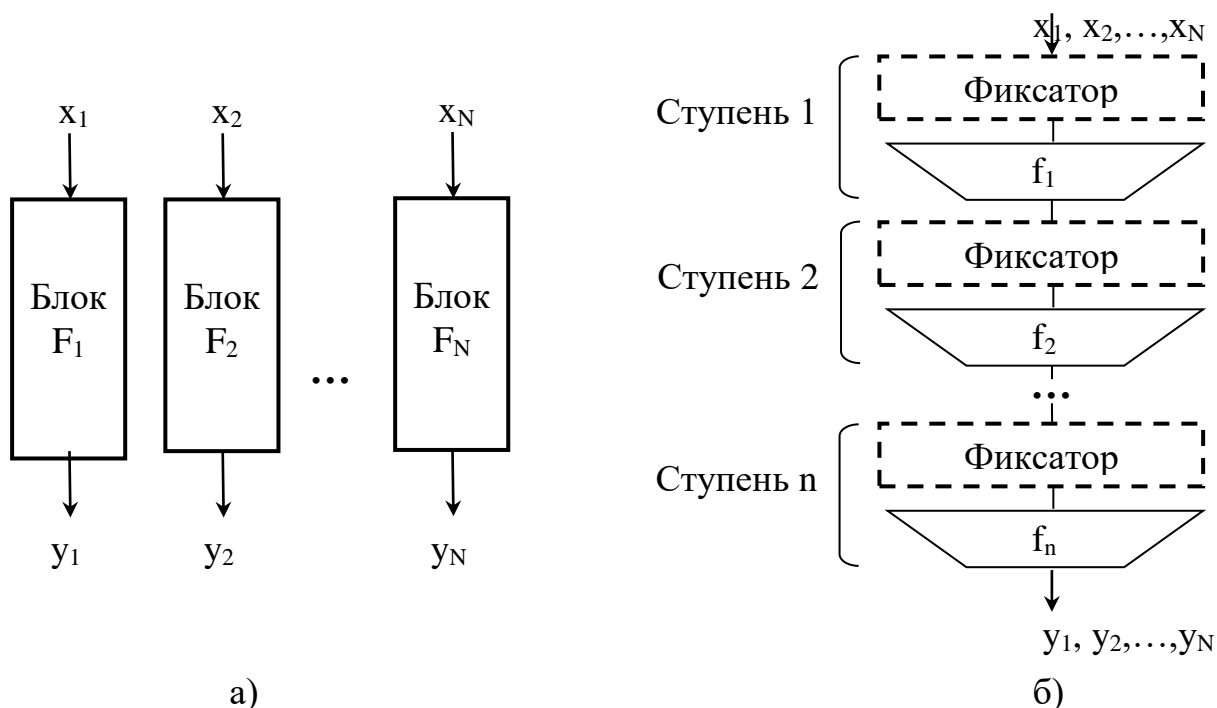


Рис. 4.2. Виды параллельной обработки: многоэлементная (а) и многостадийная (конвейерная) (б)

При конвейерной обработке функция  $F$  представляется в виде суперпозиции функций  $f_1, f_2, \dots, f_n$  (одинаковой сложности). Каждой функции в устройстве соответствует ступень конвейера, содержащая фиксатор (буферный регистр) для приема результатов предыдущей ступени и вычислительный узел  $f$

(рис. 4.2, б). При установившейся работе конвейера на вход устройства в каждый момент времени (такт) поступает одно исходное данное  $x$ , а на выход устройства выдается один результат  $y$ .

Рассмотрим пример декомпозиции функции и запуска конвейера. Предположим, что исходная функция  $F$  может быть представлена в виде суперпозиции четырех функций  $f_1, f_2, f_3, f_4$ , одинаковой сложности:  $F = f_4(f_3(f_2(f_1(x_1))))$ . Тогда конвейер будет иметь четыре ступени и выдаст первый результат в четвертом такте (табл. 4.1.)

Таблица 4.1

Декомпозиция функции и запуск конвейера

| Ступень | Такт (вход) |                 |                      |                               |
|---------|-------------|-----------------|----------------------|-------------------------------|
|         | $T_1$       | $T_2$           | $T_3$                | $T_4$                         |
|         | $x_1$       | $x_2$           | $x_3$                | $x_4$                         |
| 1       | $f_1(x_1)$  | $f_1(x_2)$      | $f_1(x_3)$           | $f_1(x_4)$                    |
| 2       | —           | $f_2(f_1(x_1))$ | $f_2(f_1(x_2))$      | $f_2(f_1(x_3))$               |
| 3       | —           | —               | $f_3(f_2(f_1(x_1)))$ | $f_3(f_2(f_1(x_2)))$          |
| 4       | —           | —               | —                    | $F = f_4(f_3(f_2(f_1(x_1))))$ |

При установившейся конвейерной обработке за время  $T/n$  вычисляется очередная функция. При  $n = N$  (число ступеней равно числу функций)  $N$  функций в пределе будет вычислено за время  $T$ .

#### 4.1.3. Виды конвейеров

Конвейеры делятся на синхронные и асинхронные. В синхронных конвейерах одновременно происходит передача информации во всех ступенях. В асинхронных конвейерах результаты на следующую ступень передаются по мере её готовности.

В общем случае на каждой из ступеней конвейера может выполняться одна из нескольких функций. При этом ступень настраивается на заданную функцию перед выполнением вычисления. Кроме того, могут быть использованы конвейеры с замыкаемыми и размыкаемыми обратными связями.

#### 4.1.4. Конвейерное выполнение микрокоманд

Процесс выполнения микрокоманды обычно делится на два этапа, приблизительно равной продолжительности: формирования адреса и чтения МК в УУ, выполнения микрокоманды и формирования значений логических условий в ОУ. Ступенями конвейера являются УУ и ОУ. Для совмещения во времени процессов в ОУ и УУ между ними устанавливаются фиксаторы – буферные (конвейерные) регистры (БР) (рис. 4.3).

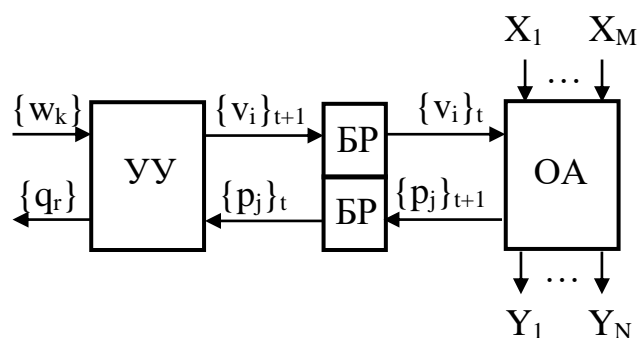


Рис. 4.3. ВУ с конвейерными регистрами

Последовательное и конвейерное выполнение микрокоманд в ВУ иллюстрируется на рис. 4.4.

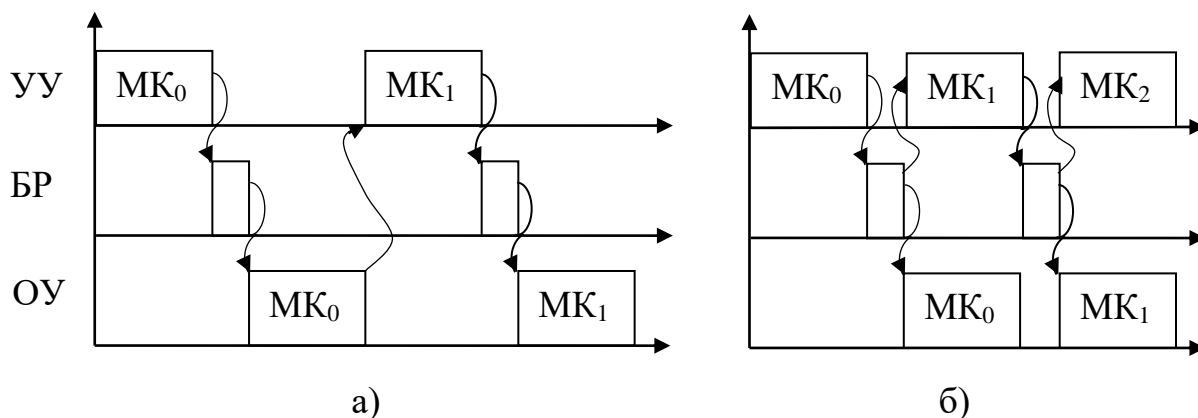


Рис. 4.4. Чтение микрокоманды в УУ и выполнение микроопераций в ОУ: последовательное (а) и параллельное (конвейерное) (б)

#### 4.1.5. Эффективность конвейерного выполнения микрокоманд

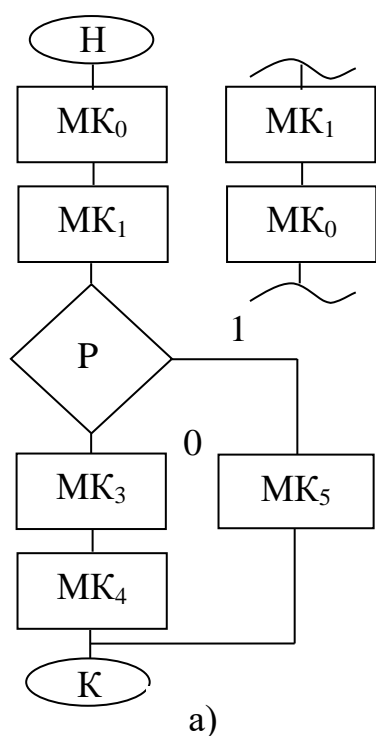
Эффективность работы конвейера максимальна на линейных участках и может снизиться при большом числе ветвлений в микропрограмме (МП).

Для ослабления влияния ветвлений на эффективность работы конвейера может быть использовано предсказание переходов. При предсказании перехода считывается МК, имеющая наибольшую вероятность выполнения. Если значение условия угадать не удалось, то происходит сбой конвейера и повторное считывание необходимой МК.

Другим способом снижения влияния ветвлений в микропрограмме на эффективность работы конвейера является преобразование микропрограммы. Если позволяет алгоритм, то микрокоманда, вычисляющая значение логического условия, перемещается на одну МК к началу МП. При этом вычисление значения логического условия производится заранее и не нарушает работу конвейера.

Если алгоритм не позволяет преобразовывать МП путем перемещения МК, то в МП вводятся пустые МК. Точнее, МК ветвления заменяется двумя МК. Первая МК выполняет необходимые микрооперации и вычисляет значение логического условия, а вторая («пустая») – выполняет только переход по вычисленному первой микрокомандой значению логического условия.

Рассмотрим простейший пример преобразования МП. На рис. 4.5 приведена граф-схема исходной МП, и показано конвейерное выполнение исходной и преобразованной МП. В исходной МП значение логического условия Р формируется в МК<sub>1</sub>. Предполагается, что алгоритм, реализуемый МП, позволяет МК<sub>1</sub> выполнить до МК<sub>0</sub> (рис. 4.5, а).



| Микро-<br>программа  | Степень | Такт |     |     |     |
|----------------------|---------|------|-----|-----|-----|
|                      |         | T1   | T2  | T3  | T4  |
| Исходная             | УУ      | MK0  | MK1 | ?   |     |
|                      | ОУ      | –    | MK0 | MK1 |     |
| Преобра-<br>зованная | УУ      | MK1  | MK0 | MK5 |     |
|                      | ОУ      | –    | MK1 | MK0 | MK5 |

б)

Рис. 4.5. Конвейерное выполнение микропрограмм: граф-схемы исходной МП и ее преобразование (а), выполнение исходной и преобразованной МП (б)



В такте  $T_3$  при выполнении исходной МП в ОУ выполняется  $МК_1$ , а в УУ должна считываться следующая МК (рис. 4.5, б). Но следующая МК не может быть считана, поскольку выбор ее определяется значением логического условия  $P$ , которое будет сформировано только по окончанию такта  $T_3$ . В преобразованной микропрограмме  $МК_1$  выполняется в такте  $T_2$ , поэтому в такте  $T_3$  неопределенности не возникает. Например, если  $P = 1$ , то в такте  $T_3$  будет считана  $МК_5$ , и выполнение МП завершится в такте  $T_4$ .

#### **4.2. Структура устройства и конвейерное выполнение микрокоманд**

Исследуемое вычислительное устройство состоит из операционного устройства и устройства управления и представляет собой микропроцессор (МПР) с микропрограммным уровнем управления. Программный уровень управления в микропроцессоре не рассматривается. Микропрограммы работы МПР размещаются в блоке памяти микропрограмм устройства управления, а для хранения исходных данных и записи результатов к микропроцессору подключается оперативное запоминающее устройство.

Формат микрокоманды МПР образуется путем объединения и соответствующей корректировки форматов микрокоманд, разработанных для операционного и управляющих устройств (из микрокоманды для операционного устройства исключается поле «Данные», и в объединенную микрокоманду добавляется поле для управления запоминающим устройством). Исследуется работа двух вариантов МПР: с обычным выполнением микрокоманд и с конвейерным выполнением микрокоманд. Структурная схема микропроцессора с запоминающим устройством приведена на рис. 4.6 (буферные регистры, включаемые в состав МПР с конвейерным выполнением микрокоманд, показаны на рисунке пунктиром).

Отличия в структуре и работе микропроцессора с конвейерным регистром микрокоманд заключаются в следующем. Для совмещения во времени выполнения микроопераций текущей микрокоманды и чтения из блока памяти микропрограмм следующей микрокоманды в состав УУ включается конвейерный регистр микрокоманд.

В этом случае при формировании адреса следующей микрокоманды используются значения логических условий, сформированные при выполнении предыдущей микрокоманды. Поэтому для сохранения конвейерной обработки микрокоманд перехода необходимо формировать и запоминать значения логических условий до выполнения микрокоманд, содержащих условный переход. Для запоминания значений логических условий в структуру микропроцессора включается буферный регистр состояния (BRS).

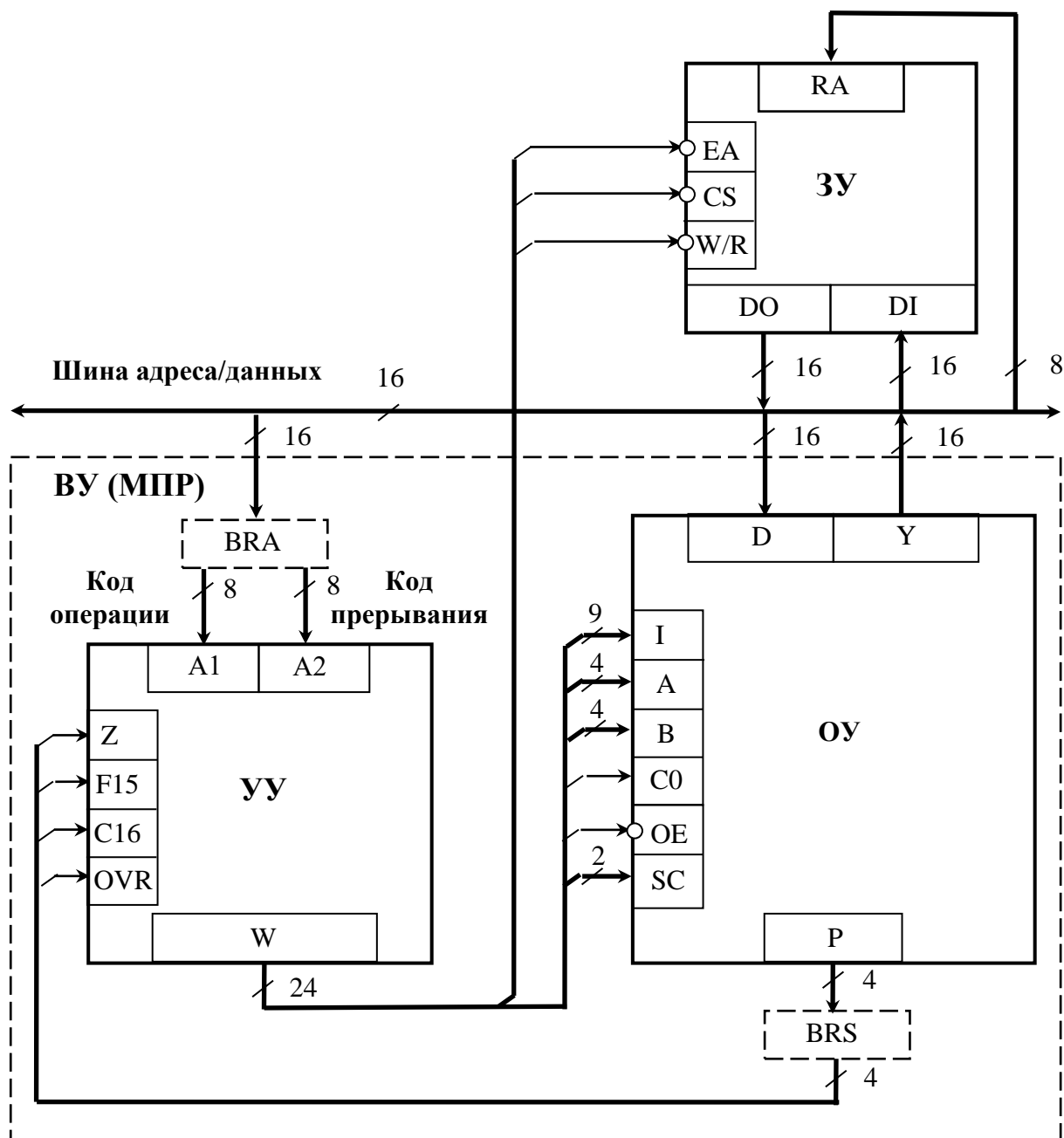


Рис. 4.6. Структурная схема микропроцессора с 3У

Кроме того, чтобы сохранить конвейерную обработку микрокоманд с микроинструкциями перехода JMAP и CJV, в структуру микропроцессора включен буферный регистр адреса (BRA). При поступлении в УУ микроинструкции перехода JMAP или CJV переход осуществляется по содержимому буферного регистра адреса (через ПНА или ПА). Предварительная запись информации (кода операции) в BRA осуществляется в каждом такте автоматически. Информация поступает с шины адреса/данных.

Технические параметры микропроцессора следующие:

- разрядность – 16;
- объем блока памяти микропрограмм – 256 слов;
- объем преобразователя начального адреса – 256 8-разрядных слов;
- объем преобразователя адреса – 256 8-разрядных слов;
- объем ОЗУ – 256 16-разрядных слов.

В моделируемом микропроцессоре разрядность ПНА и ПА равна восьми битам, так как объем памяти блока микропрограмм составляет 256 микрокоманд и для адресации к любой из них достаточно 8-разрядного адреса. Старшие четыре разряда входной шины адреса УУ не используются и равны нулю.

Запоминающее устройство включает в себя блок памяти и регистр адреса. Управление режимом работы ЗУ осуществляется с помощью сигналов  $\overline{CS}$ ,  $\overline{W}/R$ , а управление записью адреса во внутренний регистр адреса сигналом  $\overline{EA}$  (см. табл. 4.2).

Таблица 4.2

Режимы работы оперативного запоминающего устройства

| $\overline{CS}$ | $\overline{W}/R$ | $\overline{EA}$ | Режим работы                   |
|-----------------|------------------|-----------------|--------------------------------|
| 0               | 0                | 0               | некорректная комбинация        |
| 0               | 0                | 1               | запись данных                  |
| 0               | 1                | 0               | некорректная комбинация        |
| 0               | 1                | 1               | чтение данных                  |
| 1               | 0                | 0               | хранение данных, запись адреса |
| 1               | 0                | 1               | хранение данных                |
| 1               | 1                | 0               | хранение данных, запись адреса |
| 1               | 1                | 1               | хранение данных                |

Формат микрокоманды для микропроцессора приведен на рис. 4.7.

| Операционная часть |     |   |          |                 |    |                 |                |                 | Управляющая часть |      |           |   |                  |         |                  |                 |
|--------------------|-----|---|----------|-----------------|----|-----------------|----------------|-----------------|-------------------|------|-----------|---|------------------|---------|------------------|-----------------|
| МИ                 | РЗУ |   | Упр. АЛУ |                 |    | Упр. ОЗУ        |                |                 | Шина              | МИ   | Упр. усл. |   |                  | Упр. УУ |                  |                 |
| I8-0               | A   | B | C0       | $\overline{OE}$ | SC | $\overline{CS}$ | $\overline{W}$ | $\overline{EA}$ | D11-0             | I3-0 | A         | U | $\overline{CCE}$ | C0      | $\overline{RLD}$ | $\overline{OE}$ |

Рис. 4.7. Формат микрокоманды для микропроцессора

### 4.3. Лабораторная работа № 3

Выполнение лабораторной работы складывается из домашней подготовки, экспериментальных исследований и оформления отчета. Домашняя подготовка включает разработку алгоритма и микропрограммы для решения учебной задачи, указанной в задании. При решении учебной задачи предусматривается выборка данных из ЗУ и дешифрация кода операции. В задании предполагается, что в качестве основы для реализации операции используются микропрограммы, разработанные при выполнении лабораторных работ №№ 1, 2. Разрабатываются два варианта микропрограммы решения учебной задачи: без конвейерного выполнения микрокоманд и с конвейерным выполнением микрокоманд. Кроме того, проводится сравнение этих микропрограмм по времени выполнения и объему занимаемой памяти.

Экспериментальные исследования заключаются во вводе, отладке и выполнении разработанных микропрограмм. Экспериментальные исследования производятся в два этапа: сначала исследуется микропрограмма без конвейерного выполнения микрокоманд, а затем с конвейерным выполнением микрокоманд.

Отчет по лабораторной работе должен содержать: титульный лист, задание, граф-схемы микропрограмм, диаграммы распределения памяти микропрограмм и тексты отлаженных микропрограмм для ВУ без конвейерного выполнения микрокоманд и с конвейерным выполнением микрокоманд, а также результаты сравнения этих микропрограмм.

Рассмотрим выполнение лабораторной работы на примере задания № 3.

### 4.3.1. Задание № 3

Разработать микропрограмму для операции деления целых положительных чисел нацело:  $Z = ]X/Y[$ , где  $X, Y, Z$  – целые положительные числа в диапазоне от 0 до 32767. Кроме результата  $Z$  предусмотреть формирование значения признака переполнения  $P$  ( $P = 1$ , если  $Y = 0$ , иначе  $P = 0$ ). Числа  $X$  и  $Y$  перед выполнением операции находятся в ЗУ. Результат  $Z$  и значение признака переполнения  $P$  записываются после выполнения операции в ЗУ. Кроме того, в одной из ячеек ЗУ хранится код операции деления целых положительных чисел нацело, по которому осуществляется переход на микропрограмму выполнения операции.

### 4.3.2. Распределение ячеек ЗУ и регистров микропроцессора

Распределение ячеек ЗУ показано в табл. 4.3.

Таблица 4.3

Распределение ячеек ЗУ

| Адрес | Код  | Мнемоника | Комментарий          |
|-------|------|-----------|----------------------|
| 00    | 0100 | КОП       | Код операции         |
| 01    |      | X         | Делимое              |
| 02    |      | Y         | Делитель             |
| 03    |      | Z         | Частное              |
| 04    |      | P         | Признак переполнения |

Распределение регистров микропроцессора приведено на рис. 4.8. В отличие от распределения регистров, использованного при исследовании ОУ, в МПР дополнительно выделяется регистр счетчика адреса ЗУ (R15).

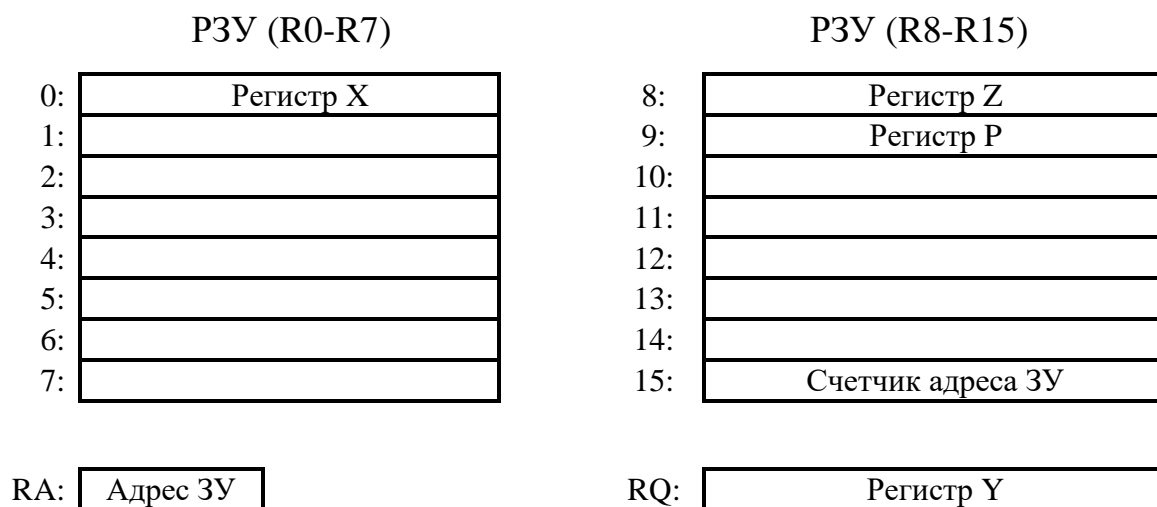


Рис. 4.8. Распределение регистров

#### **4.3.3. Разработка микропрограммы для устройства без конвейерного выполнения микрокоманд**

При разработке микропрограммы предполагается, что она состоит из двух частей (микропрограмм). В первой микропрограмме производится выборка операндов из ЗУ, дешифрация кода операции и запуск второй микропрограммы, а также запись результатов операции. Вторая микропрограмма обеспечивает выполнение операции. Эта микропрограмма работает с данными, уже находящимися в регистрах операционного устройства.

Разработка микропрограммы для устройства без конвейерного выполнения микрокоманд включает выбор кода операции (деления нацело) и распределение ячеек преобразователя начального адреса, разработку граф-схем для первой и второй частей микропрограммы, а также кодирование микропрограммы и составление таблицы прошивок блока памяти микропрограмм.

Для операции деления чисел нацело принят код операции, равный 01. При этом вторая часть микропрограммы, выполняющая деление чисел нацело, размещена в блоке памяти микропрограмм начиная с ячейки, адрес которой равен 0С. Распределение ячеек преобразователя начального адреса, преобразующего код операции (01) в соответствующий начальный адрес микропрограммы операции (0С) представлено в табл. 4.4.

Таблица 4.4

Распределение ячеек преобразователя начального адреса

| КОП<br>(адрес) | Начальный адрес<br>микропрограммы | Комментарий                                    |
|----------------|-----------------------------------|--|
| 00             | ...                               | ...  |
| 01             | 00001100                          | 0С – адрес микропрограммы деления чисел нацело |
| 02             | ...                               | ...  |

Граф-схема микропрограммы для устройства без конвейерного выполнения микрокоманд приведена на рис. 4.9, где M[RA] – ячейка ЗУ, адрес которой указан в регистре адреса RA.

В соответствии с принятым размещением данных в ЗУ (табл. 4.3), из ячейки памяти с адресом 01 в регистр операционного устройства R0 считывается делимое X (МК 01, 02), далее из ЗУ в регистр RQ считывается делитель Y (МК 03, 04). Затем в регистр адреса ЗУ записывается нуль, и производится чтение из ЗУ кода операции (МК 05, 06). Считываемый код операции поступает в преобразователь начального адреса УУ и обеспечивает переход на первую микрокоманду микропрограммы деления чисел нацело. После выполнения микропрограммы деления чисел нацело осуществляется запись результатов в ЗУ: частного Z (МК 07, 08) и признака переполнения P (МК 09, 0A).

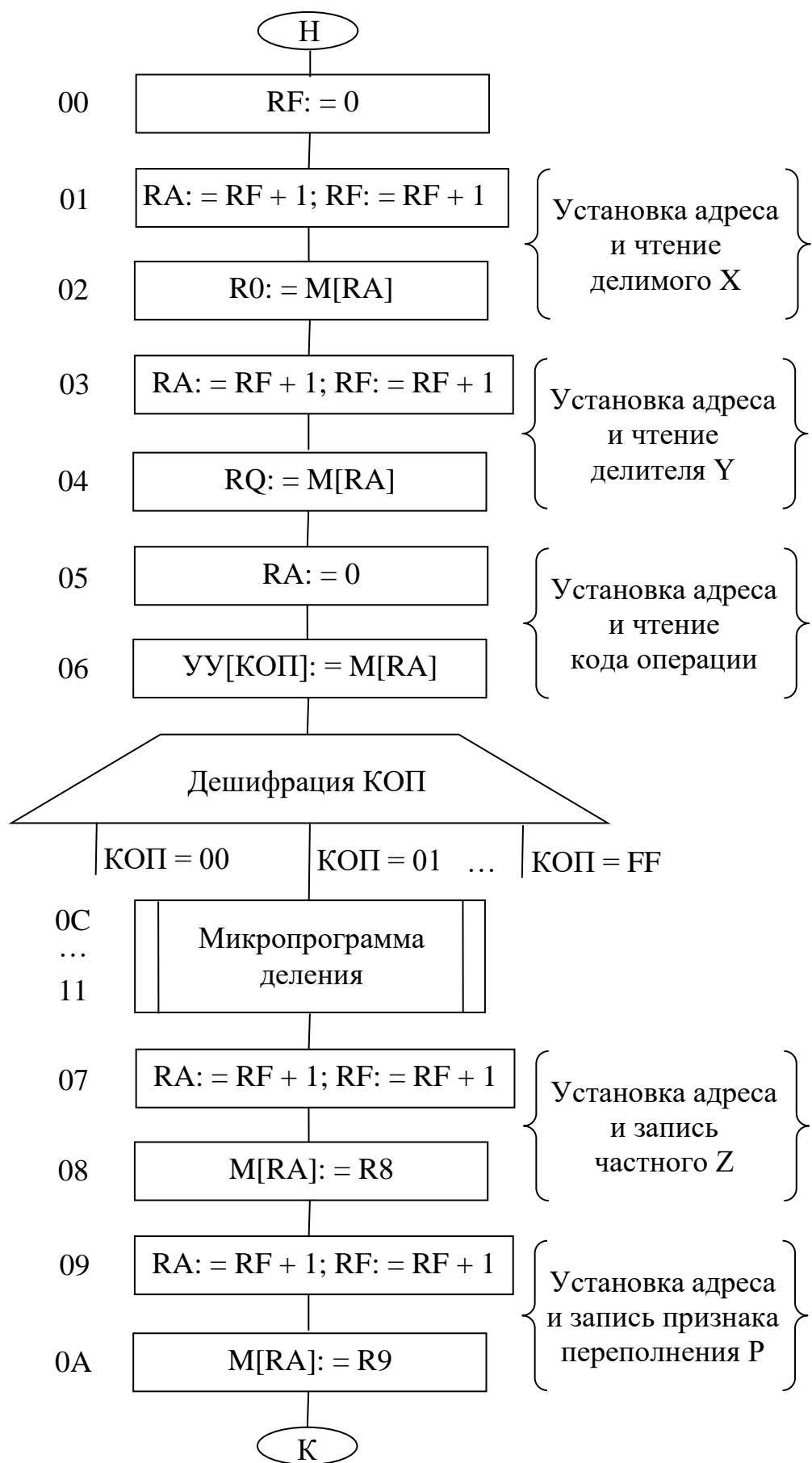


Рис. 4.9. Граф-схема микропрограммы выполнения операции в ВУ



Граф-схема микропрограммы деления чисел нацело приведена на рис. 4.10. В соответствии с принятым алгоритмом выполнения операции, перед началом цикла деления обнуляется регистр частного R8 и регистр признака переполнения R9 (МК 0C, 0D). Затем с помощью МК 0E для регистра RQ, в котором находится делитель, формируется и проверяется признак нуля (Z). Если признак нуля  $Z = 1$ , то осуществляется переход к МК 11, иначе выполняется следующая по порядку микрокоманда (МК 0F), которая является первой микрокомандой цикла деления. С помощью МК 0F содержимое регистра RQ вычитается из содержимого регистра R0. Если получается отрицательный результат ( $F15 = 1$ ), то осуществляется переход к МК 07, иначе выполняется следующая по порядку микрокоманда (МК 10). В этой микрокоманде содержимое регистра частного (R8) увеличивается на единицу, и производится переход к микрокоманде начала цикла деления (МК 0F).

Размещение кода микропрограммы в блоке памяти микропрограмм для устройства без конвейерного выполнения микрокоманд показано в табл. 4.5.

Микропрограмма деления чисел нацело размещается в блоке памяти, начиная с ячейки 0C и заканчивая ячейкой 11. Микрокоманды этой микропрограммы содержат операционные части соответствующих микрокоманд микропрограммы для ОУ, а управляющие части – микрокоманд микропрограммы для УУ (см. разделы 2 и 3).

Основная микропрограмма, в которой производится выборка операндов из ЗУ, дешифрация кода операции и запуск микропрограммы деления, а также запись результатов операции, занимает в блоке памяти микропрограмм диапазон ячеек с 00 по 0A.

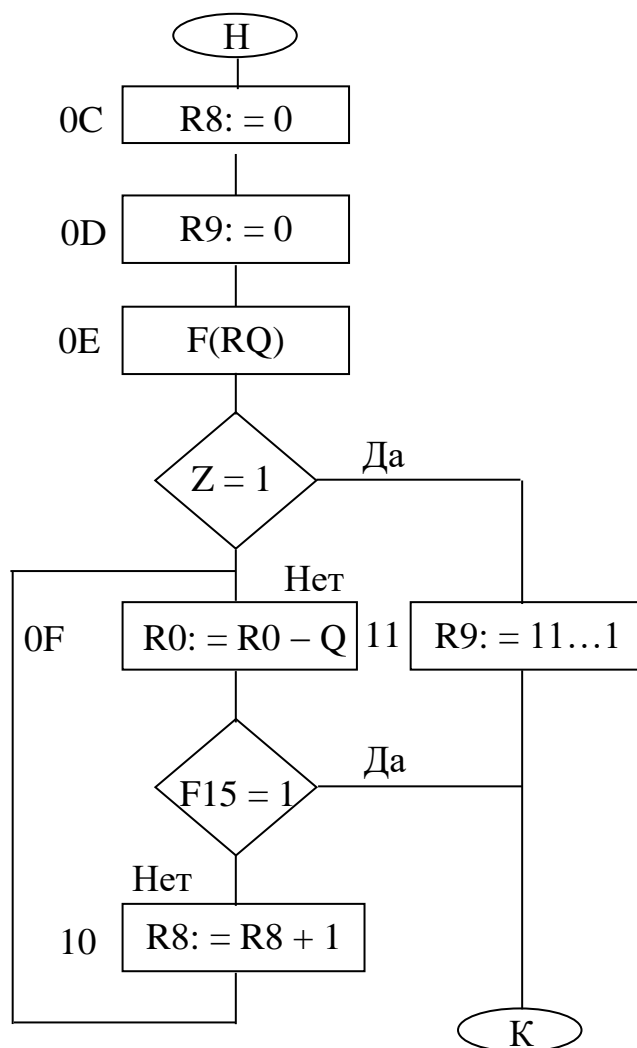


Рис. 4.10. Граф-схема деления чисел нацело (без конвейерного выполнения микрокоманд)

Таблица 4.5

Микропрограмма выполнения операции в ВУ  
без конвейерного выполнения микрокоманд

| №                          | МИ  | РЗУ |   | Упр. АЛУ |    |    | Упр. ОЗУ |   |    | Шина | МИ   | Упр. усл. |   |     | Упр. УУ |     |    |
|----------------------------|-----|-----|---|----------|----|----|----------|---|----|------|------|-----------|---|-----|---------|-----|----|
| N                          | I-9 | A   | B | C0       | OE | SC | CS       | W | EA | D-12 | I3-0 | A         | U | CCE | C0      | RLD | OE |
| 00                         | 343 | 0   | F | 0        | 1  | 00 | 1        | 1 | 1  | 000  | E    | 00        | 0 | 0   | 1       | 1   | 0  |
| RF: = 0                    |     |     |   |          |    |    |          |   |    |      |      |           |   |     |         |     |    |
| 01                         | 303 | 0   | F | 1        | 0  | 00 | 1        | 1 | 0  | 000  | E    | 00        | 0 | 0   | 1       | 1   | 0  |
| RF: = RF + 1; RA: = RF + 1 |     |     |   |          |    |    |          |   |    |      |      |           |   |     |         |     |    |
| 02                         | 337 | 0   | 0 | 0        | 1  | 00 | 0        | 1 | 1  | 000  | E    | 00        | 0 | 0   | 1       | 1   | 0  |
| R0: = X                    |     |     |   |          |    |    |          |   |    |      |      |           |   |     |         |     |    |
| 03                         | 303 | 0   | F | 1        | 0  | 00 | 1        | 1 | 0  | 000  | E    | 00        | 0 | 0   | 1       | 1   | 0  |
| RF: = RF + 1; RA: = RF + 1 |     |     |   |          |    |    |          |   |    |      |      |           |   |     |         |     |    |

| №  | МИ  | РЗУ |   | Упр. АЛУ |                 |    | Упр. ОЗУ        |                |                 | Шина | МИ   | Упр. усл. |   |                  | Упр. УУ |                  |                 |
|--|-----|-----|---|----------|-----------------|----|-----------------|----------------|-----------------|------|------|-----------|---|------------------|---------|------------------|-----------------|
| N  | I-9 | A   | B | C0       | $\overline{OE}$ | SC | $\overline{CS}$ | $\overline{W}$ | $\overline{EA}$ | D-12 | I3-0 | A         | U | $\overline{CCE}$ | C0      | $\overline{RLD}$ | $\overline{OE}$ |
| 04   | 037 | 0   | 0 | 0        | 1               | 00 | 0               | 1              | 1               | 000  | E    | 00        | 0 | 0                | 1       | 1                | 0               |
| RQ: = Y  |     |     |   |          |                 |    |                 |                |                 |      |      |           |   |                  |         |                  |                 |
| 05   | 143 | 0   | E | 0        | 0               | 00 | 1               | 1              | 0               | 000  | E    | 00        | 0 | 0                | 1       | 1                | 0               |
| RA: = 0  |     |     |   |          |                 |    |                 |                |                 |      |      |           |   |                  |         |                  |                 |
| 06   | 237 | 0   | E | 0        | 1               | 00 | 0               | 1              | 1               | 000  | 2    | 00        | 0 | 0                | 1       | 1                | 0               |
| Переход по КОП                                     |     |     |   |          |                 |    |                 |                |                 |      |      |           |   |                  |         |                  |                 |
| 07   | 303 | 0   | F | 1        | 0               | 00 | 1               | 1              | 0               | 000  | E    | 00        | 0 | 0                | 1       | 1                | 0               |
| RF: = RF + 1; RA: = RF + 1                         |     |     |   |          |                 |    |                 |                |                 |      |      |           |   |                  |         |                  |                 |
| 08   | 133 | 0   | 8 | 0        | 0               | 00 | 0               | 0              | 1               | 000  | E    | 00        | 0 | 0                | 1       | 1                | 0               |
| Запись Z   |     |     |   |          |                 |    |                 |                |                 |      |      |           |   |                  |         |                  |                 |
| 09   | 303 | 0   | F | 1        | 0               | 00 | 1               | 1              | 0               | 000  | E    | 00        | 0 | 0                | 1       | 1                | 0               |
| RF: = RF + 1; RA: = RF + 1                         |     |     |   |          |                 |    |                 |                |                 |      |      |           |   |                  |         |                  |                 |
| 0A   | 133 | 0   | 9 | 0        | 0               | 00 | 0               | 0              | 1               | 000  | E    | 00        | 0 | 0                | 1       | 1                | 0               |
| Запись P   |     |     |   |          |                 |    |                 |                |                 |      |      |           |   |                  |         |                  |                 |
| 0B   | ... | .   | . | .        | .               | .. | .               | .              | .               | ...  | .    | ..        | . | .                | .       | .                | .               |
| 0C   | 343 | 0   | 8 | 0        | 1               | 00 | 1               | 1              | 1               | 000  | E    | 00        | 0 | 0                | 1       | 1                | 0               |
| R8: = 0  |     |     |   |          |                 |    |                 |                |                 |      |      |           |   |                  |         |                  |                 |
| 0D   | 343 | 0   | 9 | 0        | 1               | 00 | 1               | 1              | 1               | 000  | E    | 00        | 0 | 0                | 1       | 1                | 0               |
| R9: = 0  |     |     |   |          |                 |    |                 |                |                 |      |      |           |   |                  |         |                  |                 |
| 0E   | 132 | 0   | 0 | 0        | 1               | 00 | 1               | 1              | 1               | 011  | 3    | 00        | 1 | 0                | 1       | 1                | 0               |
| RQ = 0 – ? (Y = 0 – ?)                             |     |     |   |          |                 |    |                 |                |                 |      |      |           |   |                  |         |                  |                 |
| 0F   | 320 | 0   | 0 | 1        | 1               | 00 | 1               | 1              | 1               | 007  | 3    | 01        | 1 | 0                | 1       | 1                | 0               |
| R0: = R0 – RQ (B: = B – C), RQ < 0 – ? (B < 0 – ?) |     |     |   |          |                 |    |                 |                |                 |      |      |           |   |                  |         |                  |                 |
| 10   | 303 | 0   | 8 | 1        | 1               | 00 | 1               | 1              | 1               | 00F  | 3    | 00        | 0 | 1                | 1       | 1                | 0               |
| R8: = R8 + 1;                                      |     |     |   |          |                 |    |                 |                |                 |      |      |           |   |                  |         |                  |                 |
| 11   | 373 | 0   | 9 | 0        | 1               | 00 | 1               | 1              | 1               | 007  | 3    | 00        | 0 | 1                | 1       | 1                | 0               |
| R9: = 11...1                                       |     |     |   |          |                 |    |                 |                |                 |      |      |           |   |                  |         |                  |                 |
| 12   | ... | .   | . | .        | .               | .. | .               | .              | .               | ...  | .    | ..        | . | .                | .       | .                | .               |

#### ***4.3.4. Разработка микропрограммы для устройства с конвейерным выполнением микрокоманд***

Совмещение во времени выполнения микроопераций текущей микрокоманды и считывания следующей позволяет использовать для формирования адреса следующей микрокоманды значения логических условий, определяемых не в текущей микрокоманде. Для сохранения конвейерной обработки микрокоманд перехода необходимо сформировать и запомнить значения логических условий до выполнения микрокоманды, содержащей условный переход. Выполнение этого условия требует преобразования микропрограмм, составленных для ВУ без конвейерного выполнения микрокоманд.

Преобразование заключается в перенесении микроопераций формирования значений логических условий из микрокоманды условного перехода в предшествующую ей микрокоманду и включении в операционную часть микрокоманды перехода микроопераций, которые могут быть выполнены до перехода, но после формирования значения условий. Если такая перестановка микроопераций не допускается алгоритмом обработки, то микрокоманда перехода заменяется двумя микрокомандами: микрокомандой, формирующей значения логических условий, и «пустой» микрокомандой, которая обеспечивает переход по условию, но не выполняет преобразований, предписываемых алгоритмом обработки.

В рассматриваемом примере преобразованию подвергается как основная микропрограмма, так и микропрограмма операции. В основной микрокоманде добавляется «пустая» микрокоманда, выполняющая дешифрацию кода операции. Более значительные изменения вносятся в микропрограмму деления чисел нацело. Граф-схема преобразованной микропрограммы деления чисел нацело изображена на рис. 4.11. Для сохранения конвейерного выполнения микрокоманд значения всех логических условий в микропрограмме операции формируются и запоминаются в регистре состояний с помощью микрокоманд, предшествующих микрокомандам условного перехода. Например, с помощью МК 0D для регистра RQ, в котором находится делитель, формируется и запоминается значение признака нуля (Z). А условный переход по значению этого признака осуществляется в МК 0E, которая записывает нуль в регистр R9.

С помощью МК 0F содержимое регистра RQ вычитается из содержимого регистра R0. При этом формируется и запоминается значение признака отрицательного результата (F7). Переход по значению этого признака выполняется в МК 07, которая увеличивает на единицу содержимое регистра частного (R8). Поскольку такая организация тела цикла приводит к дополнительному увеличению на единицу содержимого регистра R8, то его начальное значение принимается равным «-1» (МК 0C).

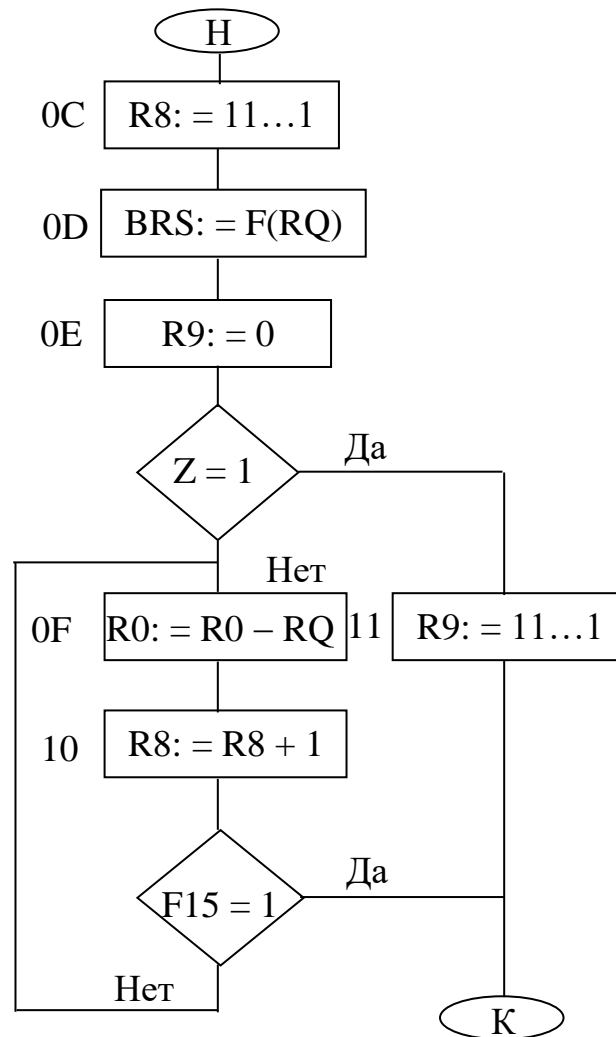


Рис. 4.11. Граф-схема деления чисел нацело (с конвейерным выполнением микрокоманд)

Размещение кода микропрограммы в блоке памяти микропрограмм для устройства с конвейерным выполнением микрокоманд показано в табл. 4.6.

Таблица 4.6

**Микропрограмма выполнения операции в ВУ с конвейерным выполнением  
микрокоманд**

| №  | МИ  | РЗУ |   | Упр. АЛУ |    |    | Упр. ОЗУ |   |    | Шина | МИ   | Упр. усл. |   |     | Упр. УУ |     |    |
|--|-----|-----|---|----------|----|----|----------|---|----|------|------|-----------|---|-----|---------|-----|----|
| N  | I-9 | A   | B | C0       | OE | SC | CS       | W | EA | D-12 | I3-0 | A         | U | CCE | C0      | RLD | OE |
| 00                                       | 343 | 0   | F | 0        | 1  | 00 | 1        | 1 | 1  | 000  | E    | 00        | 0 | 0   | 1       | 1   | 0  |
| RF: = 0                                  |     |     |   |          |    |    |          |   |    |      |      |           |   |     |         |     |    |
| 01                                       | 303 | 0   | F | 1        | 0  | 00 | 1        | 1 | 0  | 000  | E    | 00        | 0 | 0   | 1       | 1   | 0  |
| RF: = RF + 1; RA: = RF + 1               |     |     |   |          |    |    |          |   |    |      |      |           |   |     |         |     |    |
| 02                                       | 337 | 0   | 0 | 0        | 1  | 00 | 0        | 1 | 1  | 000  | E    | 00        | 0 | 0   | 1       | 1   | 0  |
| R0: = X                                  |     |     |   |          |    |    |          |   |    |      |      |           |   |     |         |     |    |
| 03                                       | 303 | 0   | F | 1        | 0  | 00 | 1        | 1 | 0  | 000  | E    | 00        | 0 | 0   | 1       | 1   | 0  |
| RF: = RF + 1; RA:= RF + 1                |     |     |   |          |    |    |          |   |    |      |      |           |   |     |         |     |    |
| 04                                       | 037 | 0   | 0 | 0        | 1  | 00 | 0        | 1 | 1  | 000  | E    | 00        | 0 | 0   | 1       | 1   | 0  |
| RQ: = Y                                  |     |     |   |          |    |    |          |   |    |      |      |           |   |     |         |     |    |
| 05                                       | 143 | 0   | E | 0        | 0  | 00 | 1        | 1 | 0  | 000  | E    | 00        | 0 | 0   | 1       | 1   | 0  |
| RA: = 0                                  |     |     |   |          |    |    |          |   |    |      |      |           |   |     |         |     |    |
| 06                                       | 237 | 0   | E | 0        | 1  | 00 | 0        | 1 | 1  | 000  | E    | 00        | 0 | 0   | 1       | 1   | 0  |
| BRA: = КОП                               |     |     |   |          |    |    |          |   |    |      |      |           |   |     |         |     |    |
| 07                                       | 133 | 0   | E | 0        | 1  | 00 | 1        | 1 | 1  | 000  | 2    | 00        | 0 | 0   | 1       | 1   | 0  |
| Переход по КОП                           |     |     |   |          |    |    |          |   |    |      |      |           |   |     |         |     |    |
| 08                                       | 303 | 0   | F | 1        | 0  | 00 | 1        | 1 | 0  | 000  | E    | 00        | 0 | 0   | 1       | 1   | 0  |
| RF: = RF + 1; RA: = RF + 1               |     |     |   |          |    |    |          |   |    |      |      |           |   |     |         |     |    |
| 09                                       | 133 | 0   | 8 | 0        | 0  | 00 | 0        | 0 | 1  | 000  | E    | 00        | 0 | 0   | 1       | 1   | 0  |
| Запись Z                                 |     |     |   |          |    |    |          |   |    |      |      |           |   |     |         |     |    |
| 0A                                       | 303 | 0   | F | 1        | 0  | 00 | 1        | 1 | 0  | 000  | E    | 00        | 0 | 0   | 1       | 1   | 0  |
| RF: = RF + 1; RA: = RF + 1               |     |     |   |          |    |    |          |   |    |      |      |           |   |     |         |     |    |
| 0B                                       | 133 | 0   | 9 | 0        | 0  | 00 | 0        | 0 | 1  | 012  | 3    | 00        | 0 | 1   | 1       | 1   | 0  |
| Запись P                                 |     |     |   |          |    |    |          |   |    |      |      |           |   |     |         |     |    |
| 0C                                       | 373 | 0   | 8 | 0        | 1  | 00 | 1        | 1 | 1  | 000  | E    | 00        | 0 | 0   | 1       | 1   | 0  |
| R8: = 11...1                             |     |     |   |          |    |    |          |   |    |      |      |           |   |     |         |     |    |
| 0D                                       | 132 | 0   | 0 | 0        | 1  | 00 | 1        | 1 | 1  | 000  | E    | 00        | 0 | 0   | 1       | 1   | 0  |
| BRS: = F(RQ)                             |     |     |   |          |    |    |          |   |    |      |      |           |   |     |         |     |    |
| 0E                                       | 343 | 0   | 9 | 0        | 1  | 00 | 1        | 1 | 1  | 011  | 3    | 00        | 1 | 0   | 1       | 1   | 0  |
| R9: = 0; RQ = 0 - ? (Y = 0 - ?)          |     |     |   |          |    |    |          |   |    |      |      |           |   |     |         |     |    |
| 0F                                       | 320 | 0   | 0 | 1        | 1  | 00 | 1        | 1 | 1  | 00F  | C    | 00        | 0 | 0   | 1       | 1   | 0  |
| R0: = R0 - RQ (B: = B - C); PA/CH: = 00F |     |     |   |          |    |    |          |   |    |      |      |           |   |     |         |     |    |
| 10                                       | 303 | 0   | 8 | 1        | 1  | 00 | 1        | 1 | 1  | 008  | 7    | 01        | 1 | 0   | 1       | 1   | 0  |
| R8: = R8 + 1; RQ < 0 - ? (B < 0 - ?)     |     |     |   |          |    |    |          |   |    |      |      |           |   |     |         |     |    |
| 11                                       | 373 | 0   | 9 | 0        | 1  | 00 | 1        | 1 | 1  | 008  | 3    | 00        | 0 | 1   | 1       | 1   | 0  |
| R9: = 11...1                             |     |     |   |          |    |    |          |   |    |      |      |           |   |     |         |     |    |

Микропрограмма деления чисел нацело размещается в блоке памяти, начиная с ячейки 0С и заканчивая ячейкой 11. Основная микропрограмма, в которой производится выборка операндов из ЗУ, дешифрация кода операции и запуск микропрограммы деления, а также запись результатов операции занимает в блоке памяти микропрограмм диапазон ячеек с 00 по 0В.

В качестве особенности микропрограммы деления чисел нацело следует отметить использование в МК 10 микроинструкции «переход на адрес, условно выбираемый из РА/СЦ либо из РМК» (код 7). При выполнении условия ( $F15 = 1$ ) осуществляется переход на адрес из РМК (08). По этому адресу размещена микрокоманда основной микропрограммы, с которой начинается запись результатов в ЗУ. Если условие не выполняется, то осуществляется переход на адрес, источником которого является РА/СЦ УУ (0F). Переход на МК 0F означает продолжение вычислений в цикле. Адрес 0F предварительно загружается в РА/СЦ с помощью МК 0F.

#### ***4.3.5. Ввод и отладка микропрограмм***

В программе «Имитатор микропрограммируемого микропроцессора» создаются файлы с расширением «*trg*», содержащие двоичные коды микропрограмм, а также коды данных, записанных в ЗУ, ПНА и ПА. Для ввода данных и микропрограмм в программе предусмотрены специальные окна и команды, подобные тем, что рассмотрены в разделах 1.3 и 3.2.

Пользователь может выбирать и выполнять микрокоманды с помощью кнопок панели инструментов и команд управляющего меню главного окна программы «Имитатор микропрограммируемого микропроцессора». В программе предусмотрено пошаговое и автоматическое выполнение микропрограммы. Порядок выполнения микрокоманд определяется заданными в них микроинструкциями переходов с учетом формируемых значений логических условий. С помощью группы команд «Скорость» может быть задана высокая, средняя или низкая скорость выполнения микропрограммы. Кроме того, используя группу команд «Выполнить», можно задавать точки останова.

По умолчанию в программе «Имитатор микропрограммируемого микропроцессора» моделируется работа ВУ без конвейерного выполнения микрокоманд. Для перевода программы в режим моделирования ВУ с конвейерным выполнением микрокоманд используется команда «Режим/Конвейер микрокоманд» (рис. 4.12, а). Повторное выполнение этой команды возвращает программу в предыдущий режим. В программе «Имитатор микропрограммируемого микропроцессора» предусмотрена возможность представления данных на экране не только в двоичной системе счисления, но и в восьмеричной и шестнадцатеричной системах счисления. Переключение системы счисления выполняется нажатием специальной кнопки на панели инструментов («2/8/16сс») или с помощью команды «Выполнить/Переключение системы счисления» (рис. 4.12, б).

| <b><u>Режим</u></b>                         |                 |
|---|-----------------|
| <b><u>В</u>вода микрокоманд</b>             | <b>Alt+1</b>    |
| <b>Вы<u>п</u>олнения <u>м</u>икрокоманд</b> | <b>Alt+2</b>    |
| <b>Редактирования ОЗУ, ПНА и ПА</b>         | <b>Alt+3</b>    |
| <b><u>П</u>ерекл<u>ю</u>чить</b>            | <b>Ctrl+Tab</b> |
| <b><u>К</u>онвейер микрокоманд</b>          |                 |

а)

| <b><u>Выполнить</u></b>                      |                |
|--|----------------|
| <b><u>М</u>икрокоманду</b>                   | <b>F8</b>      |
| <b>Микро<u>п</u>рограмму</b>                 | <b>Ctrl+F9</b> |
| <b><u>О</u>становку выполнения</b>           | <b>Ctrl+F2</b> |
| <b><u>Т</u>очка останова</b>                 | <b>Ctrl+F8</b> |
| <b><u>С</u>брос</b>                          | <b>Ctrl+R</b>  |
| <b>Переключение системы <u>с</u>числения</b> |                |

б)

Рис. 4.12. Группы команд управляющего меню программы «Имитатор микропрограммируемого микропроцессора»: «Режим» (а), «Выполнить» (б)



При запуске программы «Имитатор микропрограммируемого микропроцессора» на экран выводится главное окно приложения (рис. 4.13).

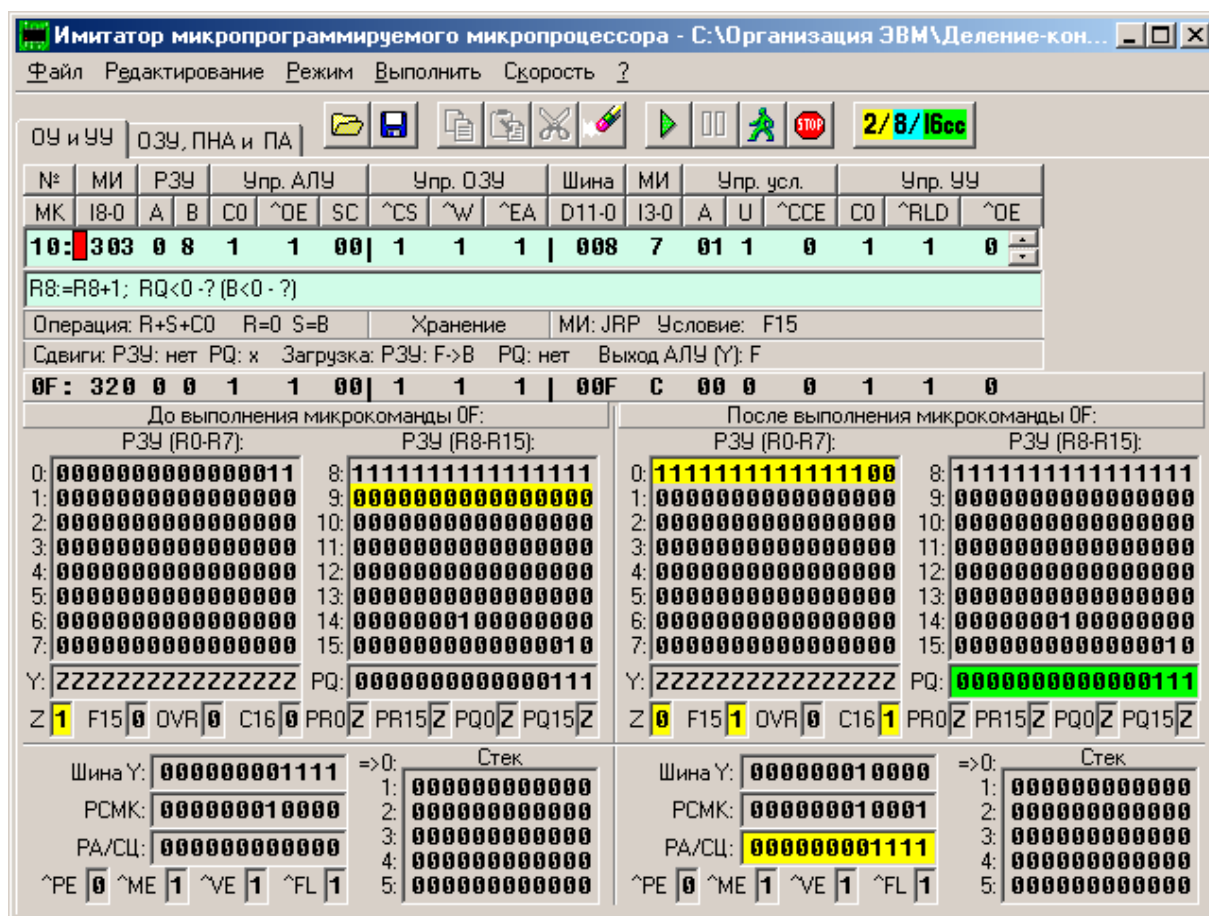


Рис. 4.13. Окно выполнения микрокоманд

Это окно имеет две вкладки: «ОУ и УУ» и «ОЗУ, ПНА и ПА». На вкладке «ОУ и УУ» поддерживается два режима работы: выполнения микрокоманд (рис. 4.8) и ввода микропрограмм (рис. 4.9).

На вкладке «ОУ и УУ» в режиме выполнения микрокоманд и содержится номер и двоичный код подлежащей выполнению микрокоманды с комментариями и расшифровкой заданных микроопераций, а также номер и код выполненной ранее микрокоманды. Кроме того, на экране отображается состояние внутренних регистров и сигналов вычислительного устройства до и после выполнения микрокоманды (рис. 4.13):

– состояние операционного устройства: регистровое ЗУ (R0,...,R15) регистр PQ, выходная шина (Y), регистр состояния RS (Z, F15, OVR, C16), сигналы сдвига (PR15, PQ15, PR0, PQ0);

– состояние управляющего устройства: регистр счетчика микрокоманд РСМК, регистр адреса РА/СЦ, стек, выходная шина адреса (Y), сигналы разрешения выборки адреса (^PE, ^ME, ^VE), сигнал переполнения стека (^FL).

Выполнение микропрограммы начинается с текущей микрокоманды и продолжается до тех пор, пока не будет считана микрокоманда, содержащая точку останова, или не будет обнаружена ошибка.

Средства ввода и редактирования микропрограмм предоставляются пользователю в режиме ввода микрокоманд (рис. 4.14).

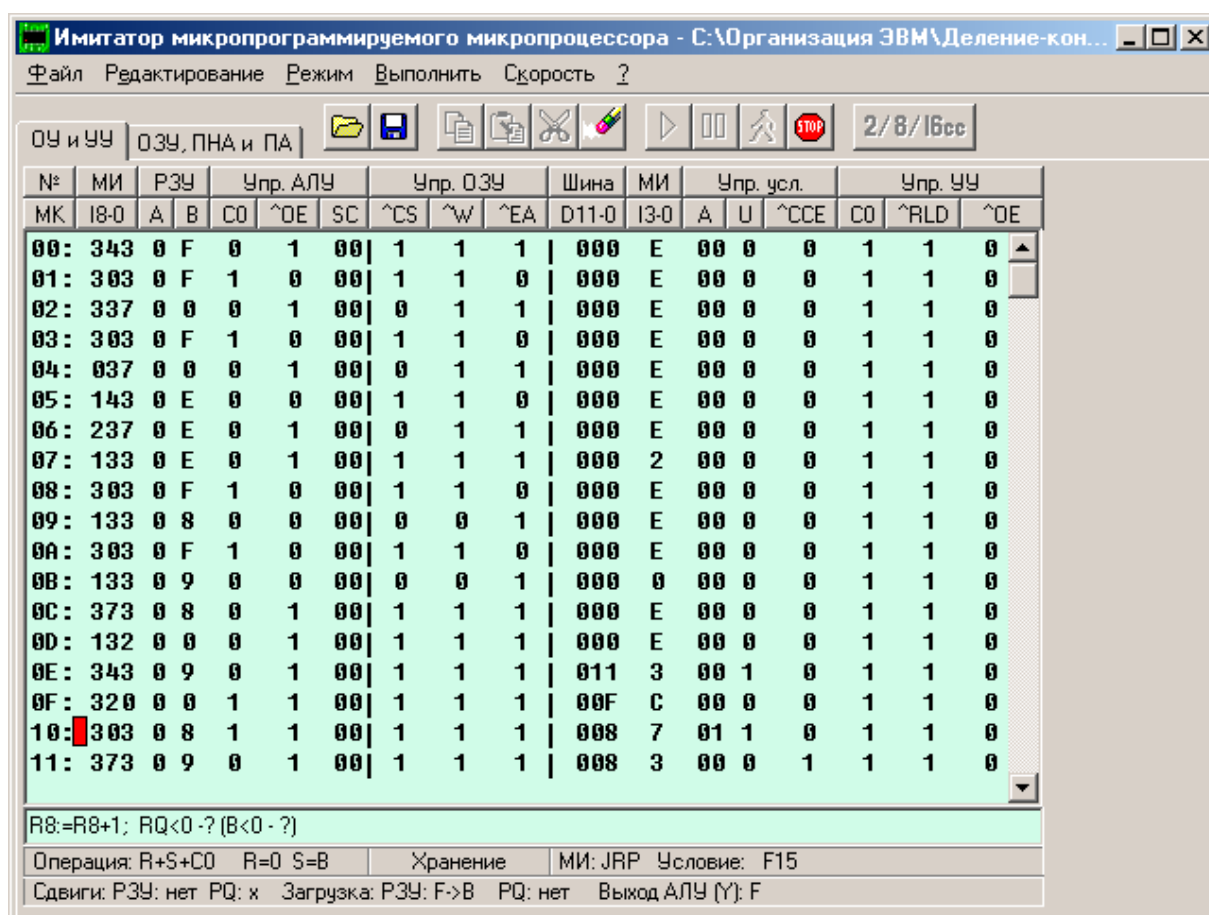


Рис. 4.14. Окно ввода микрокоманд

Переключение программы из режима выполнения в режим ввода микрокоманд и обратно может быть осуществлено с помощью двойного щелчка левой кнопки мыши в любой области ввода микрокоманды.

Ввод и редактирование данных в ЗУ, а также запись начальных адресов микропрограмм в преобразователи начального адреса осуществляется с помощью вкладки «ОЗУ, ПНА и ПА» (рис. 4.15). Здесь для представления данных в окне любого из трех устройств наряду с двоичной системой счисления могут быть также использованы восьмеричная и шестнадцатеричная системы счисления.

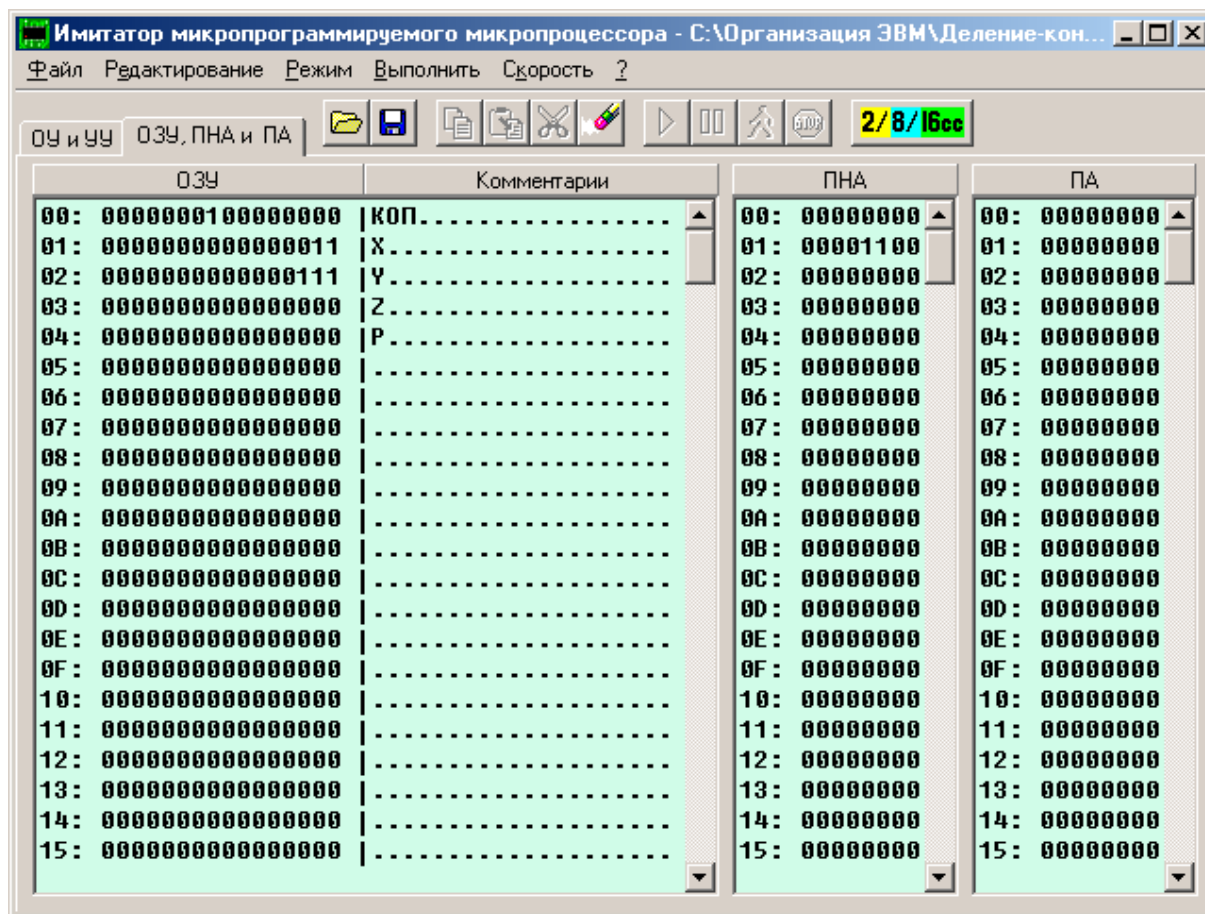


Рис. 4.15. Окно ввода данных

#### 4.3.6. Сравнение микропрограмм

Для сравнения разработанных микропрограмм по времени выполнения и объёму занимаемой памяти можно воспользоваться следующими формулами:

$$T = M \cdot \tau, \quad (4.1)$$

$$V = N \cdot n, \quad (4.2)$$

где  $T$  – среднее время выполнения микропрограммы;  $M$  – среднее число микрокоманд (тактов), выполняемых в микропрограмме;  $\tau$  – время выполнения

одной микрокоманды (длительность тактового периода);  $V$  – объём памяти, необходимый для размещения микропрограммы;  $N$  – число микрокоманд в микропрограмме;  $n$  – разрядность микрокоманды.

Среднее число выполняемых микрокоманд  $M$  определяется с учётом вероятностей ветвлений в микропрограмме. При расчёте среднего числа выполняемых микрокоманд в микропрограмме могут встретиться следующие виды участков: линейные, разветвлённые и циклические.

Для линейного участка микропрограммы среднее число выполняемых микрокоманд  $m_1$  определяется простым подсчётом микрокоманд (рис. 4.16, а):  $m_1 = l$ . Расчёт среднего числа микрокоманд  $m_2$ , выполняемых на участке с логическими условиями, рассмотрим на примере фрагмента микропрограммы, приведённого на рис. 4.16, б, где  $A_i$  – участок микропрограммы, характеризуемый средним числом выполняемых микрокоманд  $l_i$  ( $i = 1, 2, 3, 4$ ), а  $p_j$  – логическое условие, принимающее истинное значение с вероятностью  $q_j$  ( $j = 1, 2$ ). При двух логических условиях среднее число микрокоманд  $m_2$  определяется следующим образом:

$$m_2 = q_1 \cdot l_2 + (1 - q_1) \cdot (l_1 + q_2 \cdot l_4 + (1 - q_2) \cdot l_3)$$

или иначе

$$m_2 = l_1 \cdot (1 - q_1) + l_2 \cdot q_1 + l_3 \cdot (1 - q_1) \cdot (1 - q_2) + l_4 \cdot (1 - q_1) \cdot q_2.$$

В общем случае  $m_2 = \sum_{k=1}^K l_k \cdot Q_k$ , где  $K$  – число выделенных участков

$A_k$ ;  $l_k$  – среднее число выполняемых микрокоманд, характеризующих участок  $A_k$ ;  $Q_k$  – вероятность выполнения участка  $A_k$ .

Для циклического участка микропрограммы (рис. 4.16, в) среднее число выполняемых микрокоманд выполняется следующим образом:

$$m_3 = \frac{l_1 + l_2 \cdot q}{1 - q},$$

где  $q$  – вероятность повторения цикла. Если рассматривается цикл с известным числом  $R$  выполнения основной части  $A_1$ , то  $m_3 = R \cdot l_1 + (R - 1) \cdot l_2$ .

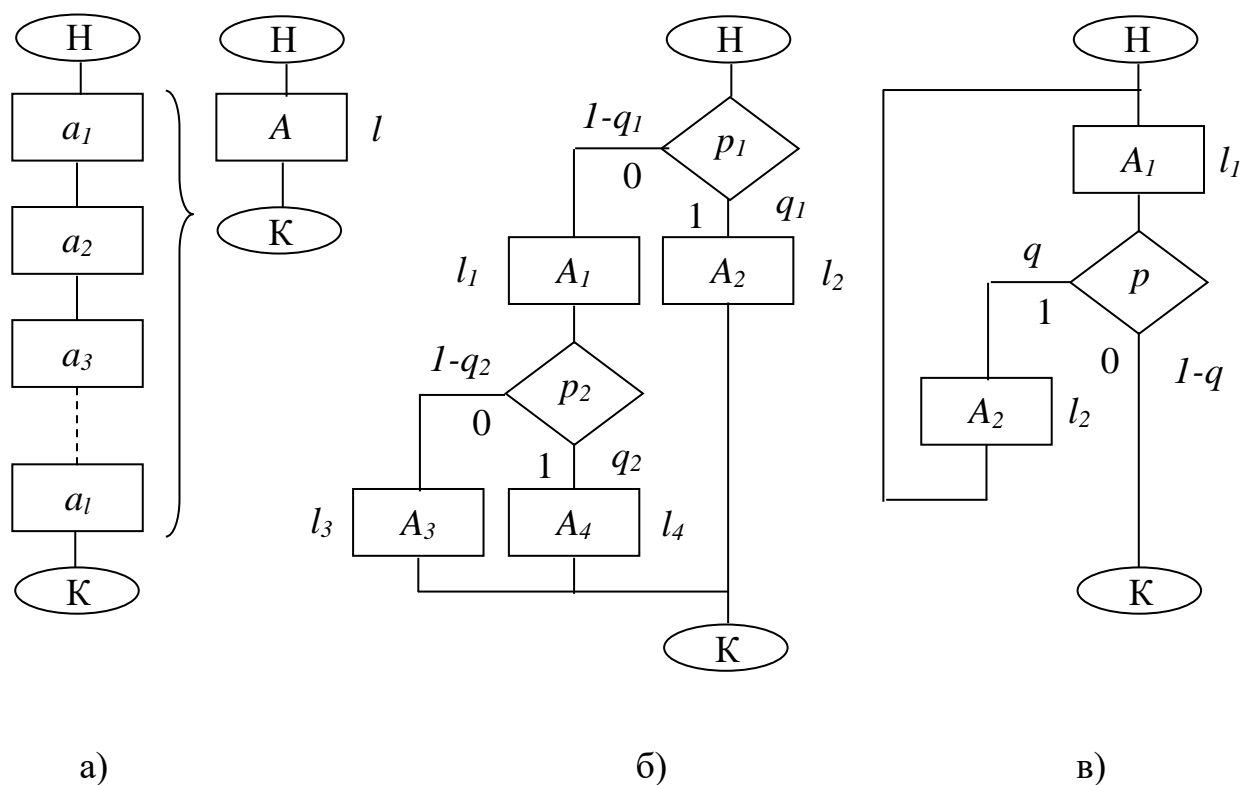


Рис. 4.16. Примеры микропрограмм основных видов: линейной (а), разветвлённой (б) и циклической (в)

Допустим, что в рассматриваемом примере среднее число циклов вычитания в микропрограмме деления равно 10, а вероятность возникновения переполнения – 0,01. Тогда среднее число тактов, необходимых для выполнения микропрограмм, можно определить по граф-схемам микропрограмм, используя приведённые выше рекомендации. В МПР без конвейерного выполнения  $M_1 \approx 32$ , а в МПР с конвейерным выполнением микрокоманд  $M_2 \approx 34$ .

Длительность тактового периода в МПР без конвейерного выполнения микрокоманд  $\tau_1$  складывается из времени задержки сигналов в ОУ –  $\tau_{OU}$ , времени задержки сигналов в УУ –  $\tau_{YU}$  и длительности тактового импульса  $\tau_H$ :  $\tau_1 = \tau_{OU} + \tau_{YU} + \tau_H$  (предполагается, что время обращения к ОЗУ входит во время задержки сигналов в ОУ).

Для МПР с конвейерным выполнением микрокоманд длительность тактового периода определяется следующим образом:  $\tau_2 = \max\{\tau_{OU}, \tau_{YU}\} + \tau_{KP} + \tau_H$ , где

$\tau_{KP}$  – время задержки сигналов на конвейерном регистре. Для расчета времени выполнения микропрограмм примем  $\tau_1 = \tau$ , а  $\tau_2 = k\tau$ .

После определения длительности тактовых периодов  $\tau_1$  и  $\tau_2$  можно рассчитать время выполнения микропрограмм. Примем  $k = 0,7$ , тогда  $T_1 = M_1 \cdot \tau = 32\tau$  и  $T_2 = M_2 \cdot k \cdot \tau = 34 \cdot 0,7 \cdot \tau \approx 24\tau$ .

Объём памяти, необходимой для размещения микропрограмм, определяется по формуле (5.2). Для МПР без конвейерного выполнения микрокоманд:  $N_1 = 17$ ,  $n_1 = 47$  и  $V_1 = 17 \cdot 47 = 779$  бит. Для МПР с конвейерным выполнением микрокоманд:  $N_2 = 18$ ,  $n_2 = 47$ ,  $V_2 = 18 \cdot 47 = 846$  бит.

Таким образом, конвейерная обработка микрокоманд позволила уменьшить время вычислений и привела к незначительному увеличению объёма памяти, необходимого для размещения микропрограммы.

#### 4.4. Контрольные вопросы

##### *Основные положения*

1. Номинальное быстродействие вычислительного устройства не зависит от класса решаемых задач, если:

- время выполнения всех операций одинаково;
- вероятности выполнения всех операций одинаковы;
- сумма вероятностей выполнения всех операций равна единице;
- сумма времени выполнения всех операций постоянна.

2. Вычислительное устройство, состоящее из устройства управления, характеризуемого временем задержки 100 нс, и операционного устройства, в котором может выполняться одна из трех микроопераций (первая микрооперация выполняется за 50, вторая – за 75, третья за – 100 нс) имеет время выполнения микрокоманды \_\_\_\_\_ нс.

3. После введения в вычислительное устройство конвейера микрокоманд число выполняемых микрокоманд увеличилось на 10 %, а время выполнения

микрокоманды сократилось на 40 % нс, при этом время выполнения микропрограммы уменьшилось в \_\_\_\_\_ раза.

4. Выполнение 100 операций сложения с плавающей запятой в 5-ступенчатом конвейерном вычислительном устройстве, имеющем такт работы конвейера 100 нс, требует \_\_\_\_\_ мкс.

5. За счет чего при конвейерном выполнении микрокоманд может быть достигнуто сокращение времени выполнения микропрограммы?

6. Почему условные переходы в микропрограммах могут снизить эффективность от конвейерного выполнения микрокоманд?

7. Какие преобразования микропрограммы помогают уменьшить негативное влияние логических условий на конвейерное выполнение микрокоманд?

*Исследуемое устройство*

8. Какие внешние входы и выходы имеет ВУ?

9. Перечислите режимы работы ЗУ. Какие комбинации управляющих сигналов определяют режимы работы ЗУ?

10. Как в ВУ осуществляется переход по коду операции, размещенному в ЗУ?

11. Как поменять значение кода операции, оставляя неизменным размещение в блоке памяти микропрограмм соответствующей ему микропрограммы?

12. Как осуществляется дешифрация кода операции при конвейерном выполнении микрокоманд?

## 5. УЧЕБНАЯ ЭВМ

### 5.1. Основные положения

#### 5.1.1. Понятие архитектуры ЭВМ

Важнейшим понятием в области вычислительной техники является понятие архитектуры ЭВМ. Первоначально в XX веке архитектуру ЭВМ определяли, как описание ЭВМ с точки зрения пользователя. Это было во времена вычислительных машин первого поколения, когда пользователями были, как правило, профессиональные математики, которым ЭВМ представлялась набором регистров и множеством ячеек памяти, хранящих двоичные коды данных и команд. С тех пор образ ЭВМ, да и сам пользователь сильно изменились.

Например, офисному служащему персональная ЭВМ может предоставить привычный для него интерфейс пользователя, создавая на экране монитора в мультипликативной форме образ рабочего стола со значками документов различных видов и папок с документами. Конечно, то, что видит такой пользователь, ничего общего с архитектурой ЭВМ не имеет. Между тем, сегодня есть специалисты, которые имеют дело с архитектурой ЭВМ.

Архитектуру ЭВМ можно рассматривать как грань, разделяющую аппаратные и программные средства ЭВМ. В этом случае архитектуру «снизу» видят специалисты в области аппаратных средств, а «сверху» – специалисты в области программных средств, непосредственно работающие с аппаратной частью ЭВМ – системные программисты.

В дальнейшем под архитектурой ЭВМ будем понимать описание ЭВМ с точки зрения системного программиста. Это описание включает внутренний язык ЭВМ и программистскую (логическую) структуру (рис. 5.1).



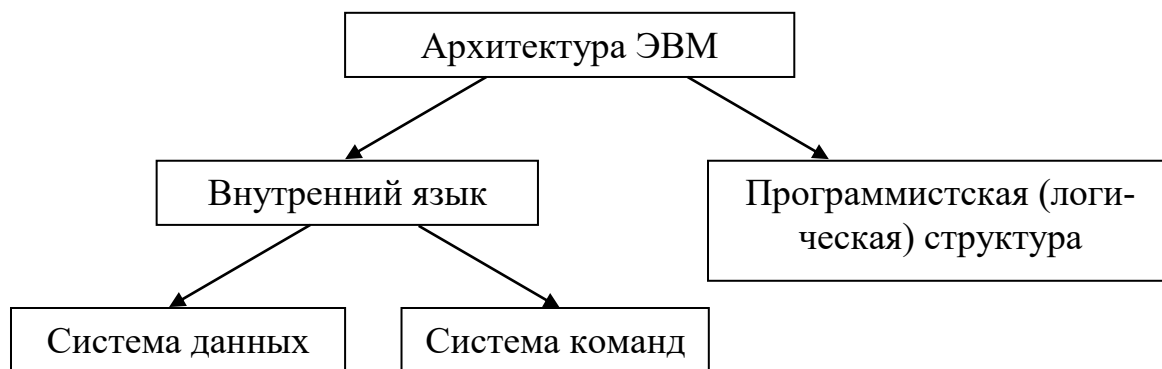


Рис. 5.1. Определение архитектуры ЭВМ

Программистская структура представляет собой множество программно доступных регистров, иногда соединенных в соответствии с направлениями пересылок данных, выполняемых командами.

Внутренний язык, в свою очередь, включает систему данных и систему команд.

Система данных определена, если заданы виды данных (числовые, символьные, графические и другие) и для каждого вида данных определены необходимые характеристики и форматы. Примерами характеристик для числовых данных могут служить: система счисления; формы представления чисел (с фиксированной запятой, с плавающей запятой); используемые коды (прямой код, дополнительный код) и другие.

Система команд определена, если заданы следующие составляющие: набор операций, способы адресации, форматы и модификации команд, а также описаны функции, выполняемые каждой командой.

### ***5.1.2. Система команд: набор операций***

*Арифметические операции.* Арифметические операции над данными в форме с фиксированной запятой: сложение, сложение модулей, вычитание, вычитание модулей, инкремент, декремент, умножение, деление и другие. Арифметические операции над данными в форме с плавающей запятой: сложение, вычитание, умножение, деление или вычисление обратной величины, суммирование произведений, вычисление элементарных функций и другие.

*Сдвиги и вспомогательные операции:* сдвиги логические в сторону младших (старших) разрядов на один разряд или на заданное число разрядов; сдвиги арифметические в сторону младших (старших) разрядов на один разряд или на заданное число разрядов; сдвиги циклические в сторону младших (старших) разрядов на один разряд или на заданное число разрядов; подсчет числа единиц (нулей) в двоичном коде; определение номера первого по порядку разряда, начиная с младших (старших) разрядов двоичного кода, содержащего цифру «0» («1»).

*Логические операции и операции установки значений:* инверсия; конъюнкция; сумма по модулю два и поразрядные операции на основе других логических функций; установка регистра или отдельных его разрядов в единичное (нулевое) состояние.

*Операции пересылки:* пересылки данных из регистра в регистр ( $R - R$ ); пересылки данных из регистра в память ( $R - M$ ), в том числе и запись в стек; пересылки данных из памяти в регистр ( $M - R$ ), в том числе и чтение из стека; пересылки данных из памяти в память ( $M - M$ ).

*Операции управления:* безусловного перехода; условного перехода по прямому (инверсному) значению признака, при этом признак может выбираться с помощью маски; условного (безусловного) обращения к подпрограмме; условного (безусловного) возврата из подпрограммы; инициализации программных прерываний; возврата из прерывающей программ; останова и другие.

*Операции ввода и вывода.* Ввод данных из регистров внешнего устройства (ВУ) осуществляется с помощью специальных команд путем пересылки данных в регистры процессора (ПР) или память. Ввод данных из регистров ВУ производится с помощью команд чтения данных из памяти в регистр ПР (или память). В этом случае регистры ВУ отображаются в адресном пространстве памяти, с которой работает процессор, и рассматриваются как ячейки памяти.

Вывод данных из регистров ПР в регистры ВУ осуществляется с помощью специальных команд путем пересылки данных из регистра ПР в регистр ВУ (из памяти в регистр ВУ). Вывод данных из регистров ПР в регистры ВУ с помощью команд записи данных в память. В этом случае регистры ВУ отображаются в адресном пространстве памяти.

*Специальные операции:* операции диагностики; операции чтения и загрузки системных регистров процессора; операции установки значений определенным полям системных регистров процессора; другие операции привилегированных команд.

В простейшем случае в команде выделяются две части: поле кода операции (КОП) и поле адресной части. Разрядность поля КОП определяется следующим образом:  $n_{КОП} = E(\log_2 K)$ ,  $K$  – число различных операций.

### 5.1.3. Модификации команд

*Одноместные и двухместные арифметико-логические операции.* Арифметико-логические операции можно поделить на двухместные  $c := a * b$  и одноместные  $d := \#a$ , где  $a$  – источник первого;  $b$  – второго операнда;  $c, d$  – приемники результатов;  $*, \#$  – символы операций.

Двухместные операции требуют введения в команду трех указателей:  $a, b, c$ . Число указателей может быть сокращено до двух, если результат операции помещается на место одного из операндов:  $a := a * b$  или  $b := a * b$ .

*Модификации команд двухместных арифметико-логических операций.* Например, если используются одноадресные команды и  $a, b \in \{AC, A\}$  ( $AC$  – идентификатор аккумулятора,  $A$  – адрес ячейки памяти), то возможны четыре модификации команд двухместных арифметико-логических операций:

$$(AC) := (AC) \% M[A],$$

$$M[A] := (AC) \% M[A],$$

$$(AC) := M[A] \% (AC),$$

$$M[A] := M[A] \% (AC);$$

где  $(AC)$  – содержимое  $AC$ , а  $M[A]$  – содержимое ячейки памяти с адресом  $A$ ,  $\%$  – операция.

*Модификации команд одноместных арифметико-логических операций.* Одноместные операции ( $d := \#a$ ) предполагают использование двух указателей:  $a$  и  $d$ . Число указателей можно сократить до одного, если результат операции

помещать на место операнда:  $a := \#a$ . Например, если  $a, d \in \{AC, M[A]\}$ , то будет четыре модификации команд одноместных арифметико-логических операций:

$$(AC) := \#M[A],$$

$$M[A] := \#(AC),$$

$$(AC) := \#(AC),$$

$$M[A] := \#M[A].$$

#### 5.1.4. Число адресных полей в команде

В простейшем случае в команде выделяются две части: поле кода операции и адресная часть. Разрядность адресной части определяется как сумма разрядностей составляющих ее адресных полей. В зависимости от числа адресов в адресной части выделяют следующие виды команд.

*Безадресные команды.* В команде отсутствует адресная часть (рис. 5.2, а). Источник операнда и приемник результата указываются неявно, обычно с помощью кода операции команды. Пример операции, выполняемой командой:  $(AC) := (AC) + 1$ .

*Одноадресные команды.* В команде используется один указатель, в качестве которого выступает номер регистра или адрес ячейки памяти (рис. 5.2, б). При выполнении двухместных операций местоположение второго операнда указывается неявно. Примеры:

1)  $(AC) := (AC) + M[A]$ , здесь  $a = A$  – адрес ячейки памяти, а  $M[A]$  – содержимое ячейки памяти с адресом  $A$ ;

2)  $R[i] := R[i] + (AC)$ , здесь  $a = i$  – номер регистра, а  $R[i]$  – содержимое регистра с номером  $i$ .

*Двухадресные команды.* В команде используется два указателя, каждый из которых может быть номером регистра или адресом ячейки памяти (рис. 5.2, в). При выполнении трехместных операций результат может замещать один из операндов. Примеры.

1)  $M[A1] := M[A1] + M[A2]$ , здесь  $a = A1$ ,  $b = A2$ ;

2)  $R[i1] := R[i2] + (AC)$ , здесь  $a = i1$ ,  $b = i2$ .

*Трехадресные команды.* Команда содержит три указателя, каждый из которых может быть номером регистра или адресом ячейки памяти (рис. 5.2, г).  
Примеры:

1)  $M[A1] := M[A2] + M[A3]$ , здесь  $a = A1$ ,  $b = A2$ ,  $c = A3$ ;

2)  $R[i1] := R[i2] + M[A1]$ , здесь  $a = i1$ ,  $b = i2$ ,  $c = A1$ .

*Четырехадресные команды.* Команда содержит четыре адресных поля: три указателя, каждый из которых может быть номером регистра или адресом ячейки памяти, и поле константы (К) или поле адреса следующей команды (АСК) (рис. 5.2, д). Причем четырехадресные команды с полем адреса следующей команды не получили практического применения.

Пример:  $R[i1] := R[i2] + R[i3] + 1$ , здесь  $a = i1$ ,  $b = i2$ ,  $c = i3$ ,  $K = 1$ .

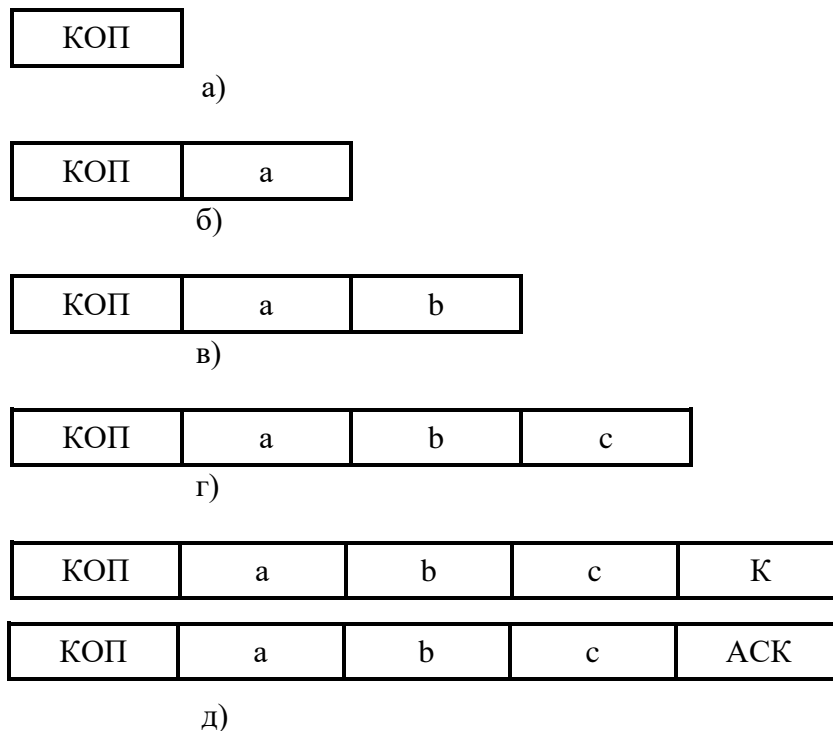


Рис. 5.2. Форматы команд с разным числом адресных полей: безадресных (а), одноадресных (б), двухадресных (в), трехадресных (г) и четырехадресных (д)

### 5.1.5. Виды программистских структур

В зависимости от числа и способов использования программно доступных регистров (ПДР) процессора выделяются следующие виды программистских структур (ПС).

– *Аккумуляторные ПС* – среди небольшого числа (как правило, специализированных) ПДР обычно выделяется один регистр – аккумулятор, который

неявно адресуется командами многих операций в качестве источника операнда и/или приемника результата.

- *ПС с регистрами общего назначения* – все или большая часть ПДР равноправны и могут единообразно использоваться в командах, при большом числе ПДР они образуют регистровый файл.

- *Стековые ПС* – для хранения операндов используется специальный регистровый стек, с которым работают безадресные команды.

#### **5.1.6. Микропрограммная и программная реализация алгоритмов**

Заданное множество алгоритмов  $\Lambda = \{A_1, A_2, \dots, A_N\}$  может быть реализовано микропрограммно с помощью вычислительного устройства. Предположим, что в ВУ используется устройство управления с программируемой логикой (УУПЛ), тогда в блоке памяти микропрограмм (БП МП) будет храниться множество микропрограмм заданных алгоритмов:  $M = \{m_1, m_2, \dots, m_N\}$  (рис. 5.3). Вычисления выполняются в операционном устройстве, а исходные данные и результаты хранятся в запоминающем устройстве. Вычислительное устройство можно рассматривать как микропрограммируемый процессор с микропрограммным уровнем управления. В данном случае программный уровень управления отсутствует.

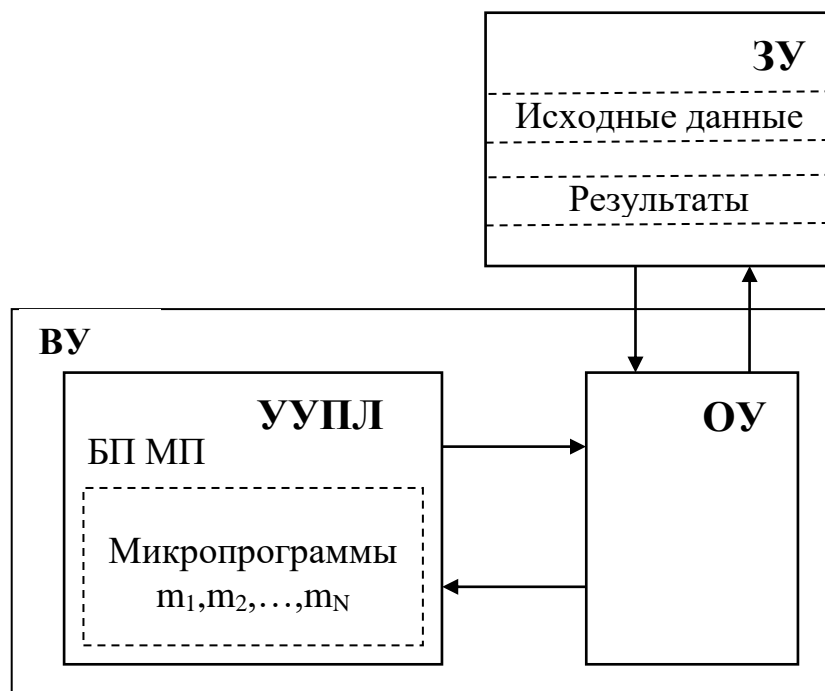


Рис. 5.3. Микропрограммная реализация алгоритмов

Заданное множество алгоритмов  $\Lambda = \{A_1, A_2, \dots, A_N\}$  может быть реализовано программно с помощью процессора (ПР). В этом случае в запоминающем устройстве будут храниться программы алгоритмов  $\Pi = \{P_1, P_2, \dots, P_N\}$  (рис. 5.4). Предположим, что в процессоре используется устройство управления с программируемой логикой (УУПЛ), тогда в памяти микропрограмм будет храниться микропрограмма командного цикла (интерпретатор команд), обеспечивающая последовательное считывание команд из запоминающего устройства и их выполнение. Вычисления выполняются в операционном устройстве, а программы реализуемых алгоритмов, исходные данные и результаты хранятся в запоминающем устройстве. В этом случае вычислительное устройство является микропрограммируемым процессором с программным уровнем управления.

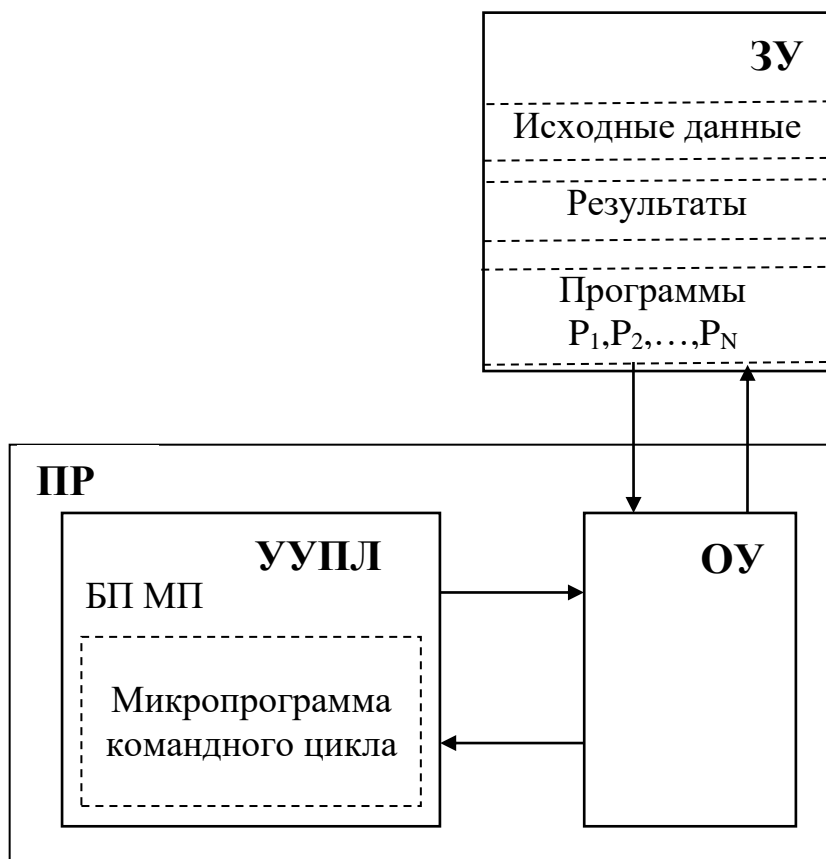


Рис. 5.4. Программная реализация алгоритмов

Как правило, микропрограммная реализация алгоритмов в сравнении с программной уменьшает время их выполнения, но требует больших аппара-

турных затрат. Более высокий программный уровень реализации алгоритмов упрощает их программирование и модификацию.

### 5.1.7. Иерархия уровней внутренних языков

Наибольшее распространение получили три вида внутренних языков процессоров: язык высокого уровня, язык команд и язык микрокоманд. Можно выделить следующие виды интерпретаторов внутренних языков ЭВМ:

- Схемный интерпретатор микроопераций (СИ МО).
- Микропрограммный интерпретатор команд (МПИ К).
- Программный интерпретатор команд.
- Программный интерпретатор предложений языка высокого уровня (ПИ ПЯВУ).
- Микропрограммный интерпретатор предложений языка высокого уровня (МПИ ПЯВУ).

Основные виды внутренних языков процессоров и их интерпретаторов показаны на рис. 5.5. Наиболее простые преобразования информации – микрооперации, задаваемые в микрокомандах, интерпретируются (выполняются) схемно с помощью узлов и блоков ЭВМ.

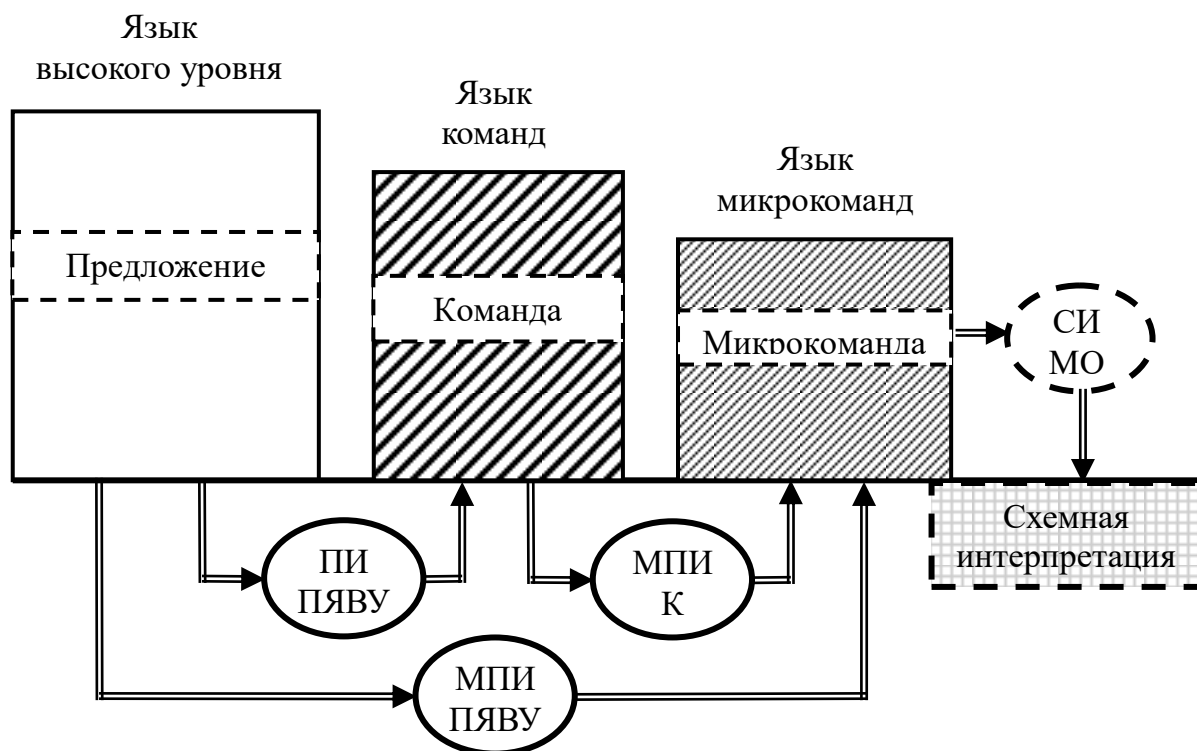


Рис. 5.5. Внутренние языки процессоров и их интерпретаторы



В зависимости от уровня внутреннего языка и способа его интерпретации можно выделить архитектуры ЭВМ, приведенные в табл. 5.1.

Таблица 5.1

Архитектуры ЭВМ в зависимости от внутреннего языка и способа интерпретации

| Уровень<br>внутреннего<br>языка               | Способ интерпретации  |   |  |
|---|---|---|--|
|   | Схемный   | Микро-<br>программный   | Программный  |
| Команды                                       | ЭВМ<br>с RISC-архитектурой<br>(с сокращенным<br>набором команд) | ЭВМ с CISC-<br>архитектурой<br>(со сложным<br>набором команд) | Виртуальные ЭВМ                                    |
| Предложения<br>языка высокого<br>уровня (ЯВУ) | —   | ЭВМ с ЯВУ<br>в качестве<br>внутреннего                        | Виртуальные ЭВМ<br>с ЯВУ в качестве<br>внутреннего |

## 5.2. Аппаратные средства для микропрограммной реализации ЭВМ

В процессе выполнения лабораторной работы разрабатывается и исследуется учебная ЭВМ. При этом рассматриваются два уровня управления: микропрограммный и программный. Разрабатываемая ЭВМ реализуется с помощью специальной микропрограммы – интерпретатора команд (микропрограммы командного цикла) на базе вычислительного устройства, которое исследовалось в предыдущей лабораторной работе. Вычислительное устройство представляет собой микропрограммируемый микропроцессор, состоящий из операционного устройства и устройства управления. Для хранения программ, исходных данных и записи результатов к микропроцессору подключается оперативное запоминающее устройство. Структурная схема микропроцессора с запоминающим устройством приведена на рис. 5.6.

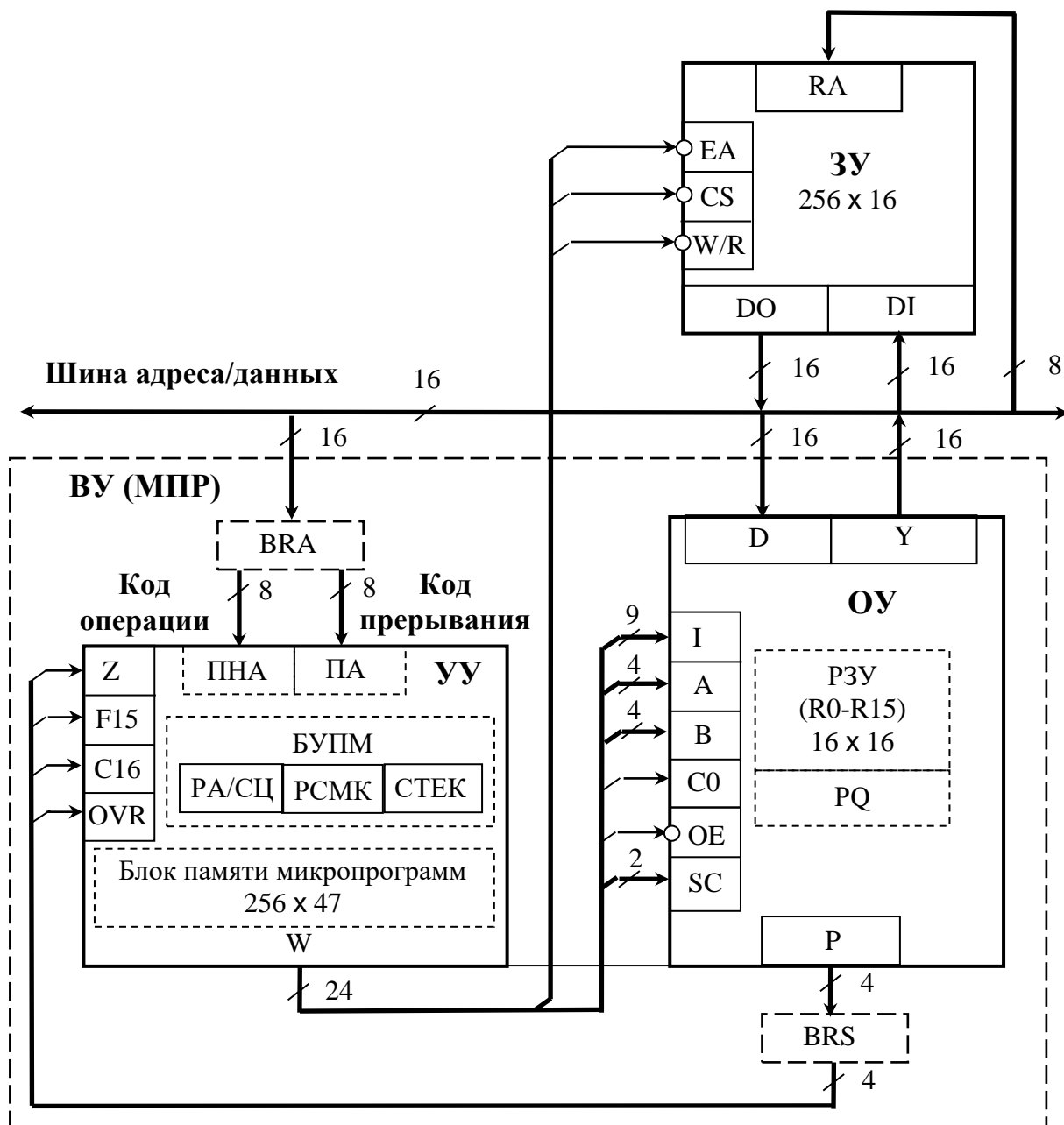


Рис. 5.6. Структурная схема микропроцессора с 3У

### 5.3. Лабораторная работа № 4

Выполнение лабораторной работы складывается из домашней подготовки, экспериментальных исследований на лабораторной установке и оформления отчета.

Для домашней подготовки необходимо получить у преподавателя задание. Домашняя подготовка включает:

- повторение лекционного материала по теме «Процессоры» и содержания практических занятий по микропрограммируемым процессорам;

- определение архитектуры учебной ЭВМ и разработку программы для решения задачи, приведенной в задании на лабораторную работу;
- разработку структурной схемы ЭВМ и граф-схемы алгоритма командного цикла;
- разработку микропрограммы командного цикла с использованием микроинструкций микропроцессора;
- расчет производительности и быстродействия ЭВМ.

Экспериментальные исследования включают: ввод, отладку и выполнение микропрограммы командного цикла по тактам, – а программы решения задачи – по командным циклам. Отчет по лабораторной работе должен содержать: титульный лист, задание, краткое описание архитектуры ЭВМ и текст отлаженной программы, структурную схему ЭВМ и граф-схему алгоритма командного цикла, распределение внутренних регистров микропроцессора, микропрограмму командного цикла, результаты расчетов среднего времени выполнения команды и программы решения задачи. Задания по лабораторной работе предполагают разработку учебной ЭВМ, ориентированной на решение простейших задач. Примером такого задания может служить задание № 4.

#### **5.3.1. Задание № 4**

Определить архитектуру ЭВМ, система команд которой состоит из одноадресных команд, использующих прямую адресацию; разработать структурную схему и алгоритм работы ЭВМ; составить и отладить микропрограмму командного цикла ЭВМ; составить и выполнить программу деления двух чисел нацело:  $Z = \lfloor X/Y \rfloor$ , где  $X$ ,  $Y$ ,  $Z$  – целые положительные числа в диапазоне от 0 до 32747, размещаемые в ЗУ. Кроме результата  $Z$ , при делении необходимо формировать и записывать в ЗУ значение признака переполнения  $P$  ( $P = 1$ , если  $Y = 0$ ;  $P = 0$ , если  $Y$  не равно 0).

#### **5.3.2. Определение архитектуры и программирование**

Разработка архитектуры ЭВМ при выполнении лабораторной работы включает: выбор форматов данных, определение системы команд и выбор состава программно-доступных регистров. Рассмотрим решение выделенных задач на примере задания № 4.

Форматы данных. Данные в примере являются целыми числами, изменяющимися в пределах от 0 до 32767. В этом случае любое число можно представить 16-разрядным двоичным кодом, старший разряд которого определяет знак числа. Значения признака переполнения  $P$  предлагается кодировать следующим образом: значению  $P = 0$  (нет переполнения) соответствует код  $000...0$ , а значению  $P = 1$  (есть переполнение) – код  $111...1$ .

Программно-доступные регистры. Программно-доступными регистрами ЭВМ, система команд которой состоит из одноадресных команд, можно считать: аккумулятор – АС, программный счетчик – РС и регистр признаков – RP, содержащий в простейшем случае разряды двух признаков: нуля (PZ) и знака (PS).

Система команд. Разработка системы команд предполагает определение и кодирование операций, способов адресаций, модификаций и форматов команд. При выполнении лабораторной работы предварительный набор операций может быть определен в процессе составления и анализа алгоритма учебной задачи, приведенной в задании. Граф-схема алгоритма деления чисел изображена на рис. 2.5.

Для системы команд, состоящей из одноадресных команд с прямой адресацией, достаточно одного формата команды, содержащего два поля: кода операции  $K$  и адреса  $A$ . Полное описание системы команд формируется в процессе разработки и анализа программы решения задачи. Для рассматриваемого примера система команд ЭВМ приведена в табл. 5.2, где  $M[A]$  – ячейка памяти с адресом  $A$ , знак «+» в описании признаков означает, что устанавливается новое значение признака по результату выполнения команды, а знак «–» свидетельствует о сохранении старого значения признака.

Таблица 5.2

## Система команд

| Наименование           | Мнемоника | Описание   | Признаки |    |
|------------------------|-----------|--|----------|----|
|                        |           |  | PZ       | PS |
| ВЫЧИТАНИЕ              | SUB A     | $AC: = AC - M[A], PC: = PC + 1$                        | +        | +  |
| ОЧИСТКА                | CLM A     | $M[A]: = 00...0, PC: = PC + 1$                         | –        | –  |
| ЗАПИСЬ ЕДИНИЦ          | MVC A     | $M[A]: = 11...1, PC: = PC + 1$                         | –        | –  |
| ЗАПИСЬ АС              | MOV A     | $M[A]: = AC, PC: = PC + 1$                             | –        | –  |
| ИНКРЕМЕНТ              | INC A     | $M[A]: = M[A] + 1, PC: = PC + 1$                       | –        | –  |
| ЗАГРУЗКА АС            | LDA A     | $AC: = M[A], PC: = PC + 1$                             | +        | +  |
| НЕТ ОПЕРАЦИИ           | NOP       | $PC: = PC + 1$   | –        | –  |
| ПЕРЕХОД                | BR A      | $PC: = A$  | –        | –  |
| ПЕРЕХОД,<br>ЕСЛИ НУЛЬ  | BEQ A     | Если $PZ = 1$ , то $PC: = A$ ,<br>иначе $PC: = PC + 1$ | –        | –  |
| ПЕРЕХОД,<br>ЕСЛИ МИНУС | BMI A     | Если $PS = 1$ , то $PC: = A$ ,<br>иначе $PC: = PC + 1$ | –        | –  |
| ОСТАНОВ                | HLT A     | $PC: = A$ , останов                                    | –        | –  |

Команды ЭВМ в зависимости от выполнения операций можно разделить на три группы.

*Арифметико-логические команды:* вычитание, очистка, инкремент и запись единицы. Операция вычитания является двухместной и может быть представлена общим выражением:  $c: = a * b$ , где  $a$  – источник первого;  $b$  – второго операнда;  $c$  – приемник результата;  $*$  – символ операции. В общем случае двухместные операции требуют введения в команду трех указателей:  $a, b, c$ . Число указателей может быть сокращено до двух, если результат операции помещается на место одного из операндов:  $a: = a * b$  или  $b: = a * b$ . Для рассматриваемых одноадресных команд  $a, b \in \{AC, M[A]\}$ , причем из четырех возможных модификаций команд:  $AC: = AC - M[A]$ ,  $M[A]: = AC - M[A]$ ,  $AC: = M[A] - AC$ ,  $M[A]: = M[A] - AC$  в команде вычитания используется только одна – первая.

Операции очистки, записи единиц и инкремента являются одноместными:  $a: = *(a)$ . Для рассматриваемых команд  $a \in \{AC, M[A]\}$ , поэтому возможны две

модификации:  $AC := *(AC)$ ,  $M[A] := *(M[A])$ . Во всех командах одноместных операций используется вторая модификация.

*Команды пересылки.* Эти команды представлены в системе команд ЭВМ командой загрузки аккумулятора и командой записи содержимого аккумулятора в память. Операции пересылки могут быть описаны общим выражением:  $a := b$ ;  $a, b \in \{R[N], M[A]\}$ , где  $R[N]$  – один из внутренних регистров ЭВМ с номером (идентификатором)  $N$ . Возможны четыре модификации команд пересылки:  $R[N] := M[A]$  – чтение, загрузка регистра;  $M[A] := R[N]$  – запись;  $R[N1] := R[N2]$  – пересылка из регистра в регистр;  $M[A1] := M[A2]$  – пересылка из ячейки в ячейку памяти. В рассматриваемой системе команд  $R[N] = AC$  и используется две модификации:  $AC := M[A]$ ,  $M[A] := AC$ .

*Команды управления:* переход по нулю, переход по минусу, переход безусловный, останов с загрузкой программного счетчика.

Для рассматриваемых одноадресных команд операции перехода можно представить следующим выражением:  $N_{J+1} = F(N_J, a, p)$ , где  $N_J$  – номер (адрес) текущей;  $N_{J+1}$  – следующей команды;  $a$  – указатель;  $p$  – условие перехода;  $F(N_J, a, p)$  – функция формирования номера (адреса) следующей команды. Тогда для операции условного перехода по нулю (знаку)  $N_J, N_{J+1} \in \{PC\}$ ,  $a \in \{A\}$ ,  $p \in \{PZ\}$  ( $p \in \{PS\}$ ) и

$$PC := \begin{cases} A, & \text{если } PZ = 1 \text{ (PS = 1),} \\ PC + 1, & \text{если } PZ = 0 \text{ (PS = 0).} \end{cases}$$

$A$  в операции безусловного перехода  $PC := A$ .

В сложных командах управления операции могут объединяться с другими операциями (формирование признака, окончание цикла, записи в стек и т. п.). В рассмотренной системе команд такой является команда останова, в которой выполнению основной операции «ОСТАНОВ» предшествует загрузка программного счетчика.

В системе команд имеются также пустые команды, осуществляющие выполнение операций безусловного перехода к следующей по порядку команде:  $PC := PC + 1$ . Следует заметить, что такая операция выполняется при выполнении каждой команды первой и второй группы.

Программа. Программа деления чисел нацело с использованием симво-  
лики ассемблера приведена на рис. 5.7.

|           |   |
|-----------|---|
| CLM AZ    | Очистка ЯП для частного Z                   |
| LDA AY    | Загрузка в АС делителя Y                    |
| BEQ m1    | Если PZ = 1 (Y = 0), то переход на метку m1 |
| LDA AX    | Загрузка в АС делимого X                    |
| m3 SUB AY | Вычитание из делимого X делителя Y          |
| BMI m2    | Если PS = 1, то переход на метку m2         |
| INC AZ    | Увеличение на единицу частного Z            |
| BR m3     | Переход на метку m3                         |
| m1 MVC AP | Запись единиц в ячейку признака             |
| BR m4     | Переход на метку m4                         |
| m2 CLM AP | Очистка ЯП для признака P                   |
| m4 HLT SA | Загрузка РС и останов                       |

Рис. 5.7. Программа деления чисел

В программе приняты следующие обозначения:

AX – адрес ячейки памяти (ЯП), в которой находится делимое;

AY – адрес ячейки ЯП, в которой находится делитель;

AZ – адрес ячейки ЯП, в которую помещается частное;

AP – адрес ячейки ЯП, в которую помещается значение признака переполнения;

SA – начальный адрес программы.

### **5.3.3. Кодирование программы и распределение памяти программ и данных**

Наименования, мнемонические обозначения и коды операций приведены в табл. 5.3.

Таблица 5.3

## Коды операций

| Наименование        | Мнемоника | Код операции |
|---------------------|-----------|--------------|
| ВЫЧИТАНИЕ           | SUB       | 00000001     |
| ОЧИСТКА             | CLM       | 00000010     |
| ЗАПИСЬ ЕДИНИЦ       | MVC       | 00000011     |
| ЗАПИСЬ АС           | MOV       | 00000100     |
| ИНКРЕМЕНТ           | INC       | 00000101     |
| ЗАГРУЗКА АС         | LDA       | 00000110     |
| НЕТ ОПЕРАЦИИ        | NOP       | 00000111     |
| ПЕРЕХОД             | BR        | 00001000     |
| ПЕРЕХОД, ЕСЛИ НУЛЬ  | BEQ       | 00001001     |
| ПЕРЕХОД, ЕСЛИ МИНУС | BMI       | 00001010     |
| ОСТАНОВ             | HLT       | 00000000     |

Распределение памяти программ и данных приведено в табл. 5.4.

Таблица 5.4

## Распределение памяти программ и данных

| Адрес | Код  | Мнемоника | Комментарий                                 |
|-------|------|-----------|---|
| 00    | 0006 | SA        | Начальный адрес программы                   |
| 01    |      | X         | Делимое                                     |
| 02    |      | Y         | Делитель                                    |
| 03    |      | Z         | Частное                                     |
| 04    |      | P         | Признак переполнения                        |
| 05    |      |           | Свободная ячейка памяти (ЯП)                |
| 06    | 0203 | CLM AZ    | Очистка ЯП для частного Z                   |
| 07    | 0602 | LDA AY    | Загрузка в АС делителя Y                    |
| 08    | 090E | BEQ m1    | Если PZ = 1 (Y = 0), то переход на метку m1 |
| 09    | 0601 | LDA AX    | Загрузка в АС делимого X                    |
| 0A    | 0102 | SUB AY    | Вычитание из делимого X делителя Y          |
| 0B    | 0A10 | BMI m2    | Если PS = 1, то переход на метку m2         |
| 0C    | 0503 | INC AZ    | Увеличение на единицу частного Z            |
| 0D    | 080A | BR m3     | Переход на метку m3                         |
| 0E    | 0304 | MVC AP    | Запись единиц в ячейку признака             |
| 0F    | 0811 | BR m4     | Переход на метку m4                         |
| 10    | 0204 | CLM AP    | Очистка ЯП для признака P                   |
| 11    | 0006 | HLT SA    | Загрузка PC и останов                       |



#### ***5.3.4. Разработка структуры и алгоритма работы***

Задача разработки структуры и алгоритма работы ЭВМ может быть сформулирована как оптимизационная задача. Например, достижение минимальных аппаратных затрат при заданном быстродействии или достижение максимального быстродействия при ограниченных аппаратных затратах. В общем случае число ограничений, накладываемых на различные технико-экономические показатели ЭВМ, достаточно велико, а их согласованное аналитическое представление весьма затруднительно, поэтому на практике редко удается свести решение задач проектирования ЭВМ к формальному решению оптимизационных задач. Вместе с тем проектирование обычно представляет собой итерационный процесс, на каждом шаге которого производится проверка выполнения заданных ограничений и преобразование структуры и алгоритма работы ЭВМ, обеспечивающего уменьшение (увеличение) минимизируемого (максимизируемого) показателя. В лабораторной работе рассматривается первый шаг такого проектирования – разработка начального варианта структуры и алгоритма работы ЭВМ, не предполагающего использование каких-либо способов повышения быстродействия и сокращения аппаратных затрат. Оценка разработанного варианта ЭВМ производится только по временным показателям.

Основные исходные данные для разработки структуры и алгоритма работы ЭВМ содержатся в описании архитектуры, определенной в предыдущих разделах. При разработке структуры, кроме программно-доступных регистров АС, РС, РР, в состав ЭВМ включаются программно-недоступные регистры. Это, например, регистр команд (RK) и регистры операндов (RY, RB), используемые для временного хранения команд и операндов в процессе выполнения команд. Связи между регистрами ЭВМ, а также необходимые схемы преобразования данных определяются при составлении алгоритма работы ЭВМ с учетом требуемых пересылок и преобразований данных и команд. Упрощенная структурная схема ЭВМ для рассматриваемого примера приведена на рис. 5.8, где FS, FZ – флаги соответственно «знака» и «нуля»;  $V_1, V_2, \dots, V_N$  – управляющие сигналы, вырабатываемые УУ при выполнении команды. В отличие от значе-

ний разрядов PZ, PS регистра RP, устанавливаемых после выполнения определенных команд, состояния флагов FZ, FS изменяются после выполнения каждой микрокоманды в зависимости от результата операции в АЛУ.

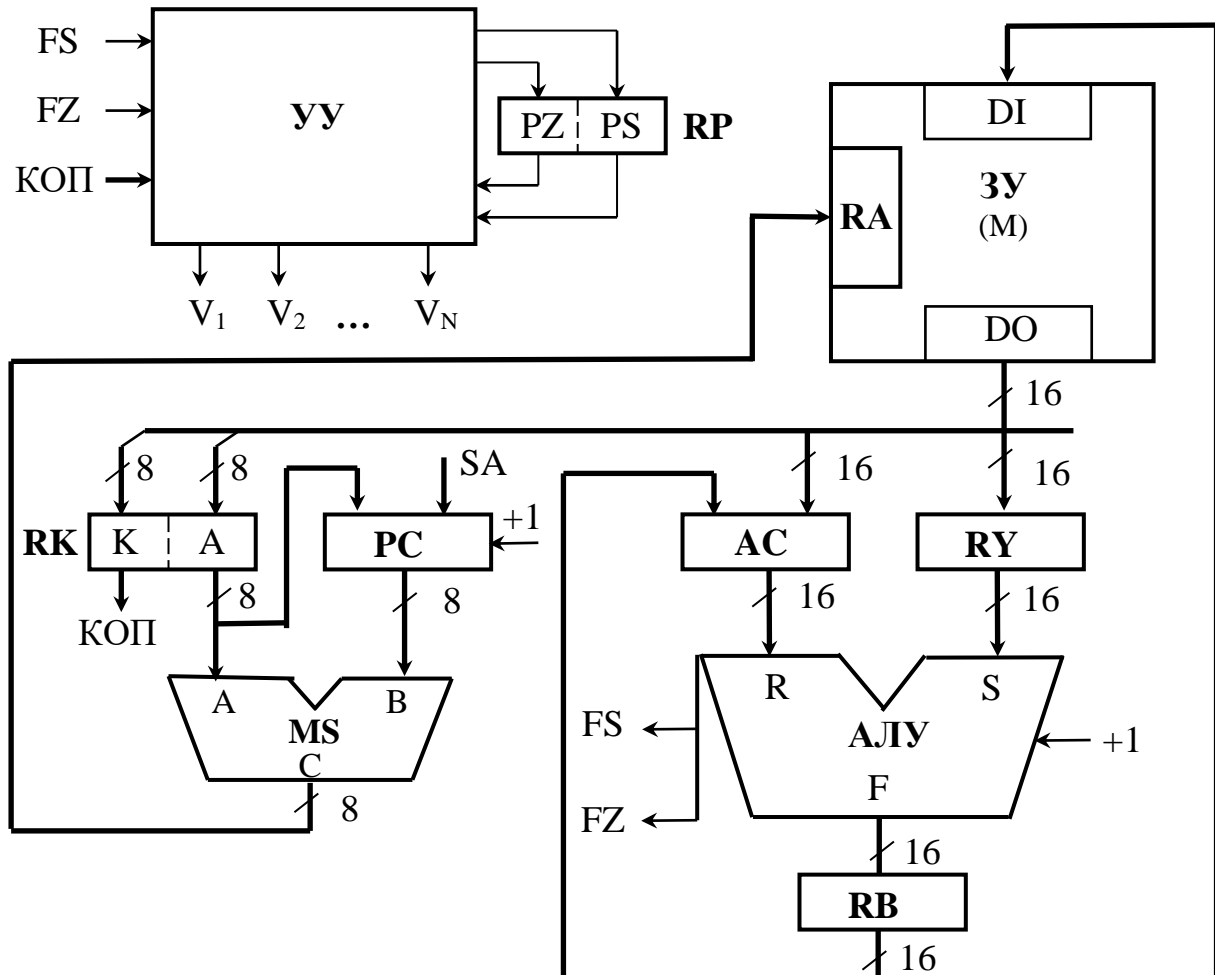


Рис. 5.8. Структурная схема учебной ЭВМ

Алгоритм работы ЭВМ представлен на рис. 5.9 в виде укрупненной граф-схемы микропрограммы командного цикла, содержащей подмикропрограммы выборки команды (ВК) и подмикропрограммы заданных операций.

Подготовка к циклу включает состояние ожидания сигнала пуска (установки специального флага:  $TST = 1$ ) и загрузку начального адреса программы (SA) в программный счетчик. В рассматриваемом примере командный цикл включает три этапа: выборку команды, дешифрацию кода операции и выполне-

ние заданной операции. Выход из командного цикла производится при выполнении команды HLT.

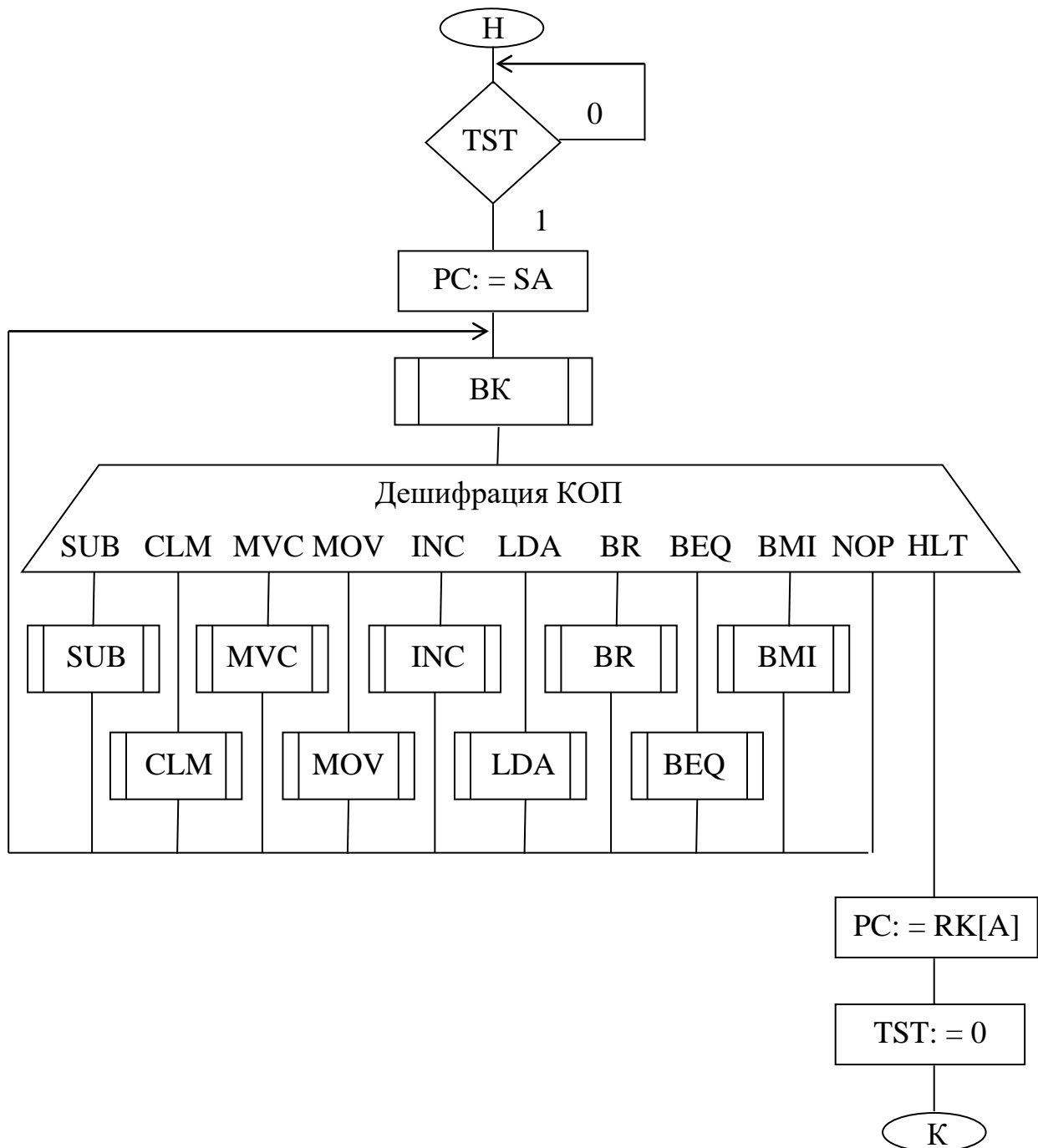


Рис. 5.9. Граф-схема микропрограммы командного цикла учебной ЭВМ

Граф-схемы подмикропрограмм, выделенных в микропрограмме командного цикла, приведены на рис. 5.10, где PZS – подмикропрограмма установки признаков PZ и PS в регистре признаков RP.

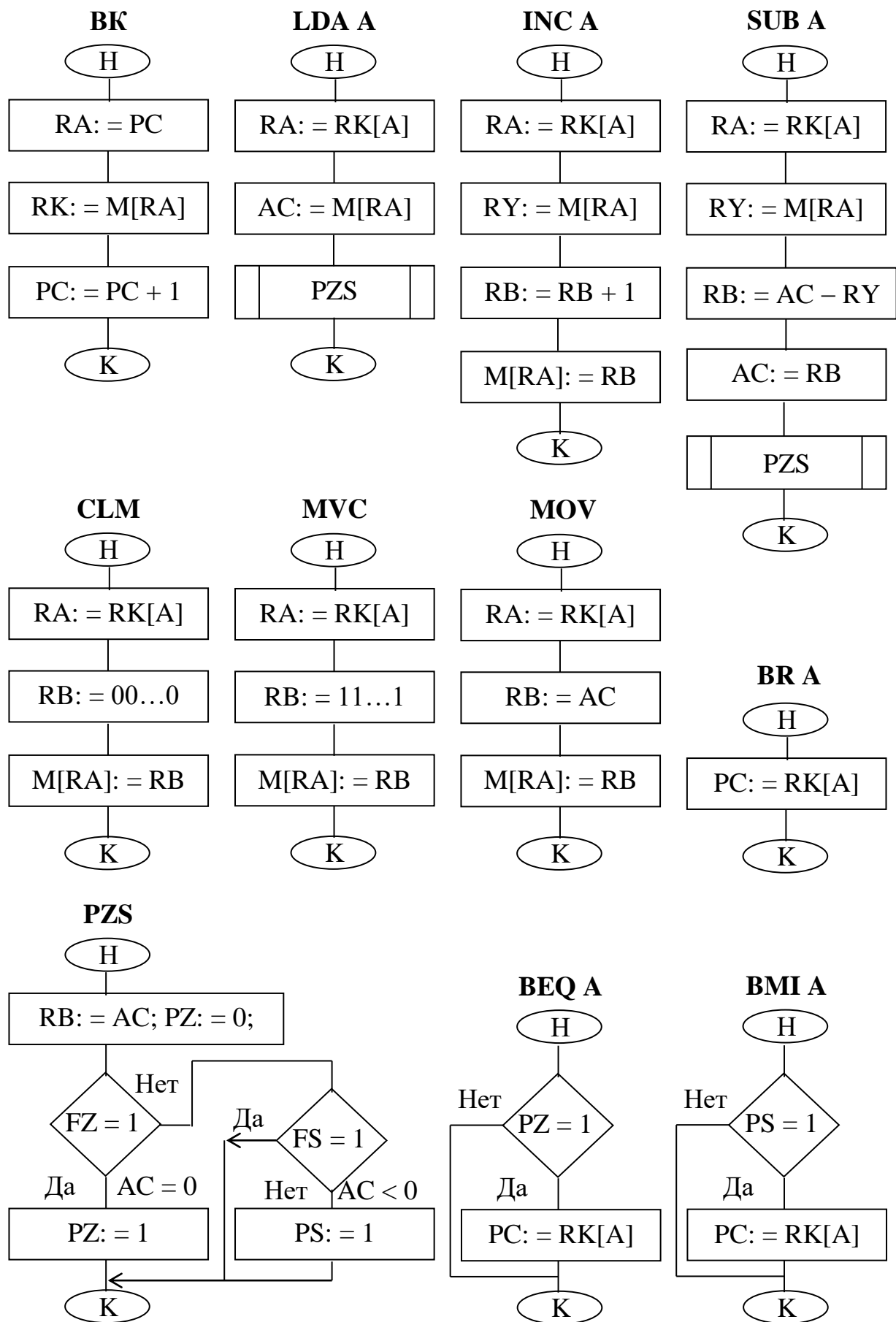


Рис. 5.10. Граф-схемы подмикروпрограмм командного цикла

### 5.3.5. Микропрограммная реализация ЭВМ

Микропрограммная реализация ЭВМ осуществляется на 16-разрядном микропроцессоре и включает: распределение внутренних регистров микропроцессора, разработку граф-схемы микропрограммы командного цикла, кодирование микропрограммы командного цикла и распределение памяти микропрограмм. Рассмотрим решение выделенных задач на примере задания № 4.

Каждому программно-доступному регистру (AC, RP, PC) ЭВМ (рис. 5.8) ставится в соответствие один из внутренних регистров микропроцессора. Программно-недоступные регистры выделяются с учетом функциональных возможностей микропроцессора и максимальным числом одновременно хранимых промежуточных результатов выполнения команд. Распределение регистров показано на рис. 5.11. Функции триггеров FS и FZ ЭВМ выполняют разряды S и Z регистра состояний операционного устройства микропроцессора.

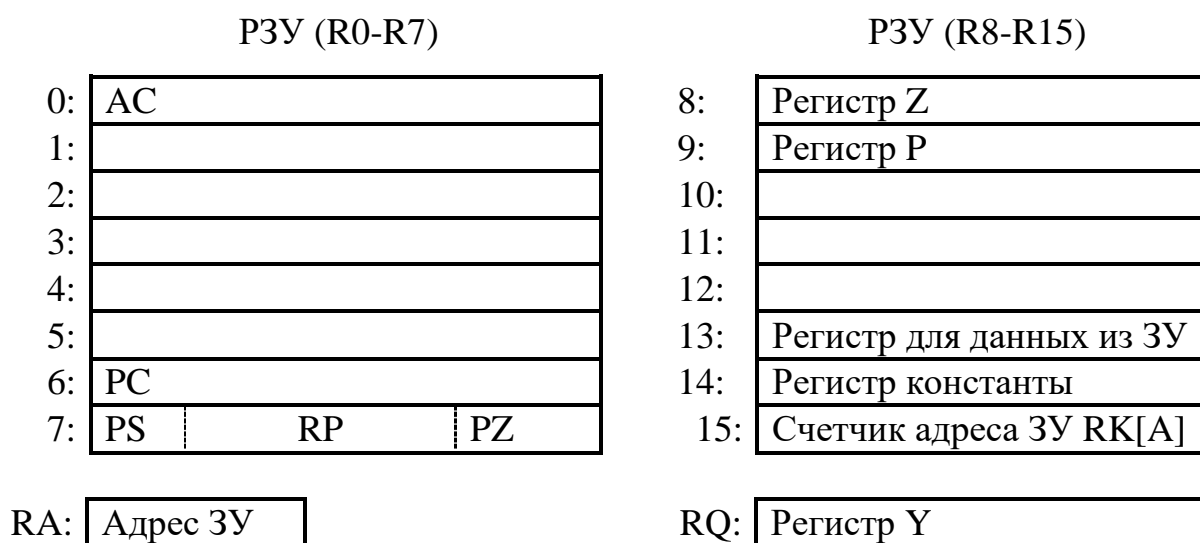


Рис. 5.11. Распределение регистров

Разрабатывается граф-схема микропрограммы командного цикла ЭВМ с использованием внутренних регистров и с учетом функциональных возможностей микропроцессора. Основу этой микропрограммы составляет микропрограмма, разработанная без ориентации на микропроцессор (рис. 5.9, 5.10). В то же время отличия логической структуры и функциональных возможностей используемого микропроцессора приводят к необходимости изменения ранее раз-

работанной микропрограммы командного цикла. Основные изменения в рассматриваемом примере касаются подмикропрограмм командного цикла (рис. 5.10). Граф-схемы подмикропрограмм операций, разработанные с ориентацией на аппаратные средства микропрограммной реализации ЭВМ, приведены на рис. 5.12, где PZS – подмикропрограмма установки признаков PZ и PS в регистре признаков RP (R7).

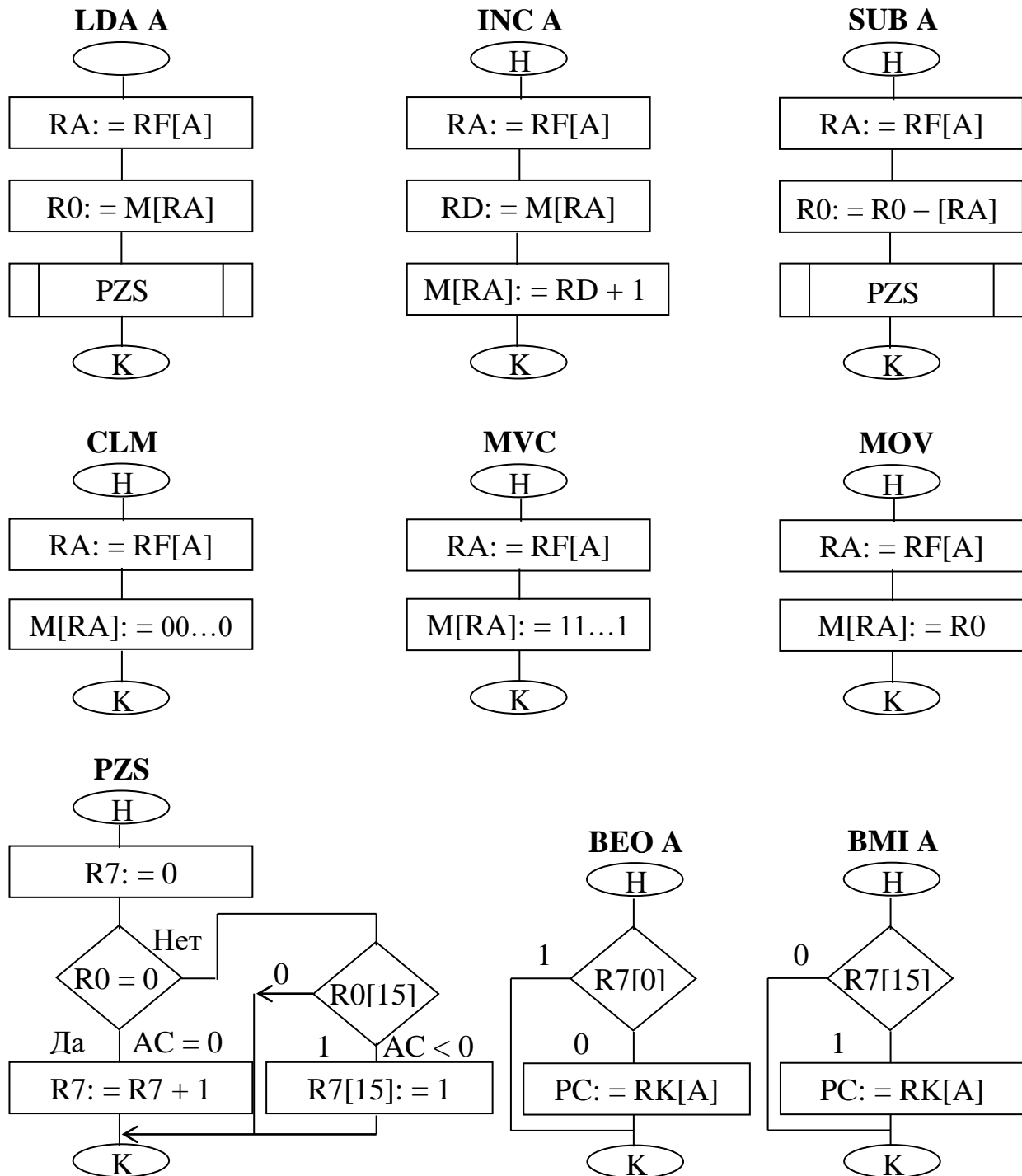


Рис. 5.12. Граф-схемы подмикропрограмм операций

Совместно решаются задачи кодирования микропрограммы командного цикла и распределения памяти микропрограмм. При этом необходимо учитывать, что микропроцессор, используемый для реализации ЭВМ, при «включении питания» начинает работу с выборки и выполнения микрокоманды, хранящейся в памяти микропрограмм по нулевому адресу. В рассматриваемом примере в ячейке памяти микропрограмм с нулевым адресом размещается первая микрокоманда микропрограммы командного цикла. Коды операций однозначно определяют адреса первых микрокоманд подмикропрограмм операций. Соответствие между кодами операций и начальными адресами подмикропрограмм в памяти микропрограмм показано в табл. 5.5.

Таблица 5.5

Коды операции и начальные адреса подмикропрограмм

| Мнемоника | Код операции | Адрес первой микрокоманды |
|-----------|--------------|---------------------------|
| SUB       | 00000001     | 00001110                  |
| CLM       | 00000010     | 00001100                  |
| MVC       | 00000011     | 00010000                  |
| MOV       | 00000100     | 00010010                  |
| INC       | 00000101     | 00010100                  |
| LDA       | 00000110     | 00010111                  |
| NOP       | 00000111     | 00011001                  |
| BR        | 00001000     | 00010011                  |
| BEQ       | 00001001     | 00011010                  |
| BMI       | 00001010     | 00011100                  |
| HLT       | 00000000     | 00011110                  |

Кодирование микропрограммы командного цикла для ЭВМ производится аналогично кодированию микропрограммы арифметической операции для вычислительного устройства на микропроцессоре (см. раздел 4). Текст микропрограммы командного цикла рассматриваемой ЭВМ, отражающий распределение памяти микропрограмм, приведен в табл. 5.6 (выборка команды, дешифрация кода операции и установка признаков) и табл. 5.7 (выполнение операций).

Микропрограмма командного цикла (табл. 5.6) начинается с формирования в регистре RE вспомогательной константы 00FF, используемой для выделения младшего байта команды, в котором находится адрес операнда (микрокоманды с номерами 00 и 01). Микрокоманды (МК) 02-04 обеспечивают загрузку начального адреса программы в программный счетчик и увеличение содержимого программного счетчика на единицу. Чтение и дешифрацию команды выполняет МК06, которая также выделяет адресную часть команды и заносит ее в регистр RF. Подмикропрограмма PZS формирования значений признаков нуля (PZ) и знака (PS) состоит из пяти микрокоманд (МК06-МК0A).

Таблица 5.6

Микропрограмма командного цикла  
(выборка команды и установка признаков)

| №  | МИ  | РЗУ |   | Упр. АЛУ |                 |    | Упр. ОЗУ        |                |                 | Шина | МИ  | Упр. усл. |   |                  | Упр. УУ |                  |                 |
|--|-----|-----|---|----------|-----------------|----|-----------------|----------------|-----------------|------|-----|-----------|---|------------------|---------|------------------|-----------------|
| МК   | I-9 | A   | B | C0       | $\overline{OE}$ | SC | $\overline{CS}$ | $\overline{W}$ | $\overline{EA}$ | D-12 | I-4 | A         | U | $\overline{CCE}$ | C0      | $\overline{RLD}$ | $\overline{OE}$ |
| 00   | 571 | E   | E | 0        | 0               | 00 | 1               | 1              | 1               | 006  | C   | 00        | 0 | 0                | 1       | 1                | 0               |
| RE: = 0111111111111111; PA/ЦИ: = 6               |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                  |                 |
| 01   | 533 | 0   | E | 0        | 0               | 00 | 1               | 1              | 1               | 001  | 9   | 00        | 0 | 0                | 1       | 1                | 0               |
| RE – сдвиг вправо; PA/ЦИ: = PA/ЦИ-1              |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                  |                 |
| 02   | 143 | 0   | 6 | 0        | 0               | 00 | 1               | 1              | 0               | 000  | E   | 00        | 0 | 0                | 1       | 1                | 0               |
| RA: = 0  |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                  |                 |
| 03   | 337 | 0   | 6 | 0        | 1               | 00 | 0               | 1              | 1               | 000  | E   | 00        | 0 | 0                | 1       | 1                | 0               |
| R6: = SA (PC: = SA)                              |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                  |                 |
| 04   | 203 | 6   | 6 | 1        | 0               | 00 | 1               | 1              | 0               | 000  | E   | 00        | 0 | 0                | 1       | 1                | 0               |
| RA: = R6; R6: = R6 + 1 (RA: = PC; PC: = PC + 1)  |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                  |                 |
| 05   | 245 | E   | F | 0        | 1               | 00 | 0               | 1              | 1               | 000  | 2   | 00        | 0 | 0                | 1       | 1                | 0               |
| Переход по КОП, RF: = K[A] (RK[A]: = K[A])       |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                  |                 |
| 06   | 343 | 0   | 7 | 0        | 1               | 00 | 1               | 1              | 1               | 000  | E   | 00        | 0 | 0                | 1       | 1                | 0               |
| R7: = 0 (RP: = 0) (PZS)                          |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                  |                 |
| 07   | 133 | 0   | 0 | 0        | 1               | 00 | 1               | 1              | 1               | 00A  | 3   | 00        | 1 | 0                | 1       | 1                | 0               |
| Переход, если R0 = 0 (AC = 0)                    |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                  |                 |
| 08   | 133 | 0   | 0 | 0        | 1               | 00 | 1               | 1              | 1               | 004  | 3   | 01        | 0 | 0                | 1       | 1                | 0               |
| Переход, если R0[15] = 0 (AC[15] = 0)            |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                  |                 |
| 09   | 503 | 0   | 7 | 1        | 1               | 01 | 1               | 1              | 1               | 004  | 3   | 00        | 0 | 1                | 1       | 1                | 0               |
| R7: = R7 + 1; циклический сдвиг вправо (PS: = 1) |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                  |                 |
| 0A   | 303 | 0   | 7 | 1        | 1               | 00 | 1               | 1              | 1               | 004  | 3   | 00        | 0 | 1                | 1       | 1                | 0               |
| R7: = R7 + 1 (PZ: = 1)                           |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                  |                 |
| 0B   | ... | .   | . | .        | .               | .. | .               | .              | .               | ...  | .   | ..        | . | .                | .       | .                | .               |



После дешифрации кода операции (МК5) управление передается одной из одиннадцати микрокоманд (табл. 5.7), с которой начинается подмикропрограмма соответствующей операции. С целью уменьшения необходимого объема памяти микропрограмм объединены заключительные части подмикропрограмм отдельных операций. Для операций SUB и LDA общей является подмикропрограмма формирования признаков PZC.

Таблица 5.7

Микропрограмма командного цикла (выполнение операций)

| №                         | МИ  | РЗУ |   | Упр. АЛУ |                 |    | Упр. ОЗУ        |                |                 | Шина | МИ  | Упр. усл. |   |                  | Упр. УУ |     |                 |
|---------------------------|-----|-----|---|----------|-----------------|----|-----------------|----------------|-----------------|------|-----|-----------|---|------------------|---------|-----|-----------------|
| МК                        | I-9 | A   | B | C0       | $\overline{OE}$ | SC | $\overline{CS}$ | $\overline{W}$ | $\overline{EA}$ | D-12 | I-4 | A         | U | $\overline{CCE}$ | C0      | RLD | $\overline{OE}$ |
| 0C                        | 133 | 0   | F | 0        | 0               | 00 | 1               | 1              | 0               | 000  | E   | 00        | 0 | 0                | 1       | 1   | 0               |
| RA: = RF[A] (CLM A)       |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |     |                 |
| 0D                        | 143 | 0   | F | 0        | 0               | 00 | 0               | 0              | 1               | 004  | 3   | 00        | 0 | 1                | 1       | 1   | 0               |
| M[RA]: = 0                |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |     |                 |
| 0E                        | 133 | 0   | F | 0        | 0               | 00 | 1               | 1              | 0               | 000  | E   | 00        | 0 | 0                | 1       | 1   | 0               |
| RA: = RF[A] (SUB A)       |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |     |                 |
| 0F                        | 315 | 0   | 0 | 1        | 1               | 00 | 0               | 1              | 1               | 006  | 3   | 00        | 0 | 1                | 1       | 1   | 0               |
| R0: = R0 – M[RA]          |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |     |                 |
| 10                        | 133 | 0   | F | 0        | 0               | 00 | 1               | 1              | 0               | 000  | E   | 00        | 0 | 0                | 1       | 1   | 0               |
| RA: = RF[A] (MVC A)       |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |     |                 |
| 11                        | 171 | E   | F | 0        | 0               | 00 | 0               | 0              | 1               | 004  | 3   | 00        | 0 | 1                | 1       | 1   | 0               |
| M[RA]: = 1111111111111111 |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |     |                 |
| 12                        | 133 | 0   | F | 0        | 0               | 00 | 1               | 1              | 0               | 000  | E   | 00        | 0 | 0                | 1       | 1   | 0               |
| RA: = RF[A] (MOV A)       |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |     |                 |
| 13                        | 134 | 0   | 0 | 0        | 1               | 00 | 0               | 0              | 1               | 004  | 3   | 00        | 0 | 1                | 1       | 1   | 0               |
| M[RA]: = R0               |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |     |                 |
| 14                        | 133 | 0   | F | 0        | 0               | 00 | 1               | 1              | 0               | 000  | E   | 00        | 0 | 0                | 1       | 1   | 0               |
| RA: = RF[A] (INC A)       |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |     |                 |
| 15                        | 337 | 0   | D | 0        | 1               | 00 | 0               | 1              | 1               | 000  | E   | 00        | 0 | 0                | 1       | 1   | 0               |
| RD: = M[RA]               |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |     |                 |
| 16                        | 303 | 0   | D | 1        | 0               | 00 | 0               | 0              | 1               | 004  | 3   | 00        | 0 | 1                | 1       | 1   | 0               |
| M[RA]: = RD + 1           |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |     |                 |
| 17                        | 133 | 0   | F | 0        | 0               | 00 | 1               | 1              | 0               | 000  | E   | 00        | 0 | 0                | 1       | 1   | 0               |
| RA: = RF[A] (LDA A)       |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |     |                 |
| 18                        | 337 | 0   | 0 | 0        | 1               | 00 | 0               | 1              | 1               | 006  | 3   | 00        | 0 | 1                | 1       | 1   | 0               |
| R0: = M[RA]               |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |     |                 |
| 19                        | 133 | 0   | 0 | 0        | 1               | 00 | 1               | 1              | 1               | 004  | 3   | 00        | 0 | 1                | 1       | 1   | 0               |

| №  | МИ  | РЗУ |   | Упр. АЛУ |                 |    | Упр. ОЗУ        |                |                 | Шина | МИ  | Упр. усл. |   |                  | Упр. УУ |                  |                 |
|--|-----|-----|---|----------|-----------------|----|-----------------|----------------|-----------------|------|-----|-----------|---|------------------|---------|------------------|-----------------|
| МК   | I-9 | A   | B | C0       | $\overline{OE}$ | SC | $\overline{CS}$ | $\overline{W}$ | $\overline{EA}$ | D-12 | I-4 | A         | U | $\overline{CCE}$ | C0      | $\overline{RLD}$ | $\overline{OE}$ |
| <b>(NOP)</b>   |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                  |                 |
| 1A   | 113 | 0   | 7 | 0        | 1               | 00 | 1               | 1              | 1               | 004  | 3   | 00        | 0 | 0                | 1       | 1                | 0               |
| (AC = 0 – ?) Переход, если R7[0] = 0 ( <b>BEQ A</b> )  |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                  |                 |
| 1B   | 334 | F   | 6 | 0        | 1               | 00 | 1               | 1              | 1               | 004  | 3   | 00        | 0 | 1                | 1       | 1                | 0               |
| R6: = RF (PC: = RK[A]) ( <b>BR A</b> )                 |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                  |                 |
| 1C   | 133 | 0   | 7 | 0        | 1               | 00 | 1               | 1              | 1               | 004  | 3   | 01        | 0 | 0                | 1       | 1                | 0               |
| (AC < 0 – ?) Переход, если R7[15] = 0 ( <b>BMI A</b> ) |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                  |                 |
| 1D   | 334 | F   | 6 | 0        | 1               | 00 | 1               | 1              | 1               | 004  | 3   | 00        | 0 | 1                | 1       | 1                | 0               |
| R6: = RF (PC: = RK[A])                                 |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                  |                 |
| 1E   | 334 | F   | 6 | 0        | 1               | 00 | 1               | 1              | 1               | 000  | 3   | 00        | 0 | 1                | 1       | 1                | 0               |
| R6: = RF (PC: = RK[A]) ( <b>HLT A</b> )                |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                  |                 |

### 5.3.6. Ввод и отладка микропрограммы командного цикла и программы решения задачи

В лабораторной работе используется программа «Имитатор микропрограммируемого микропроцессора», возможности которой достаточно подробно рассмотрены в разделе 5. После загрузки и запуска программы для лабораторных исследований она переводится в режим ввода микропрограмм, в котором производится ввод разработанной микропрограммы командного цикла. Отладка и выполнение микропрограммы осуществляется в режиме выполнения микрокоманд. Ввод и редактирование данных в ЗУ, а также запись в преобразователь начального адреса начальных адресов микропрограмм операций осуществляется с помощью вкладки «ОЗУ, ПНА и ПА» (рис. 5.13).

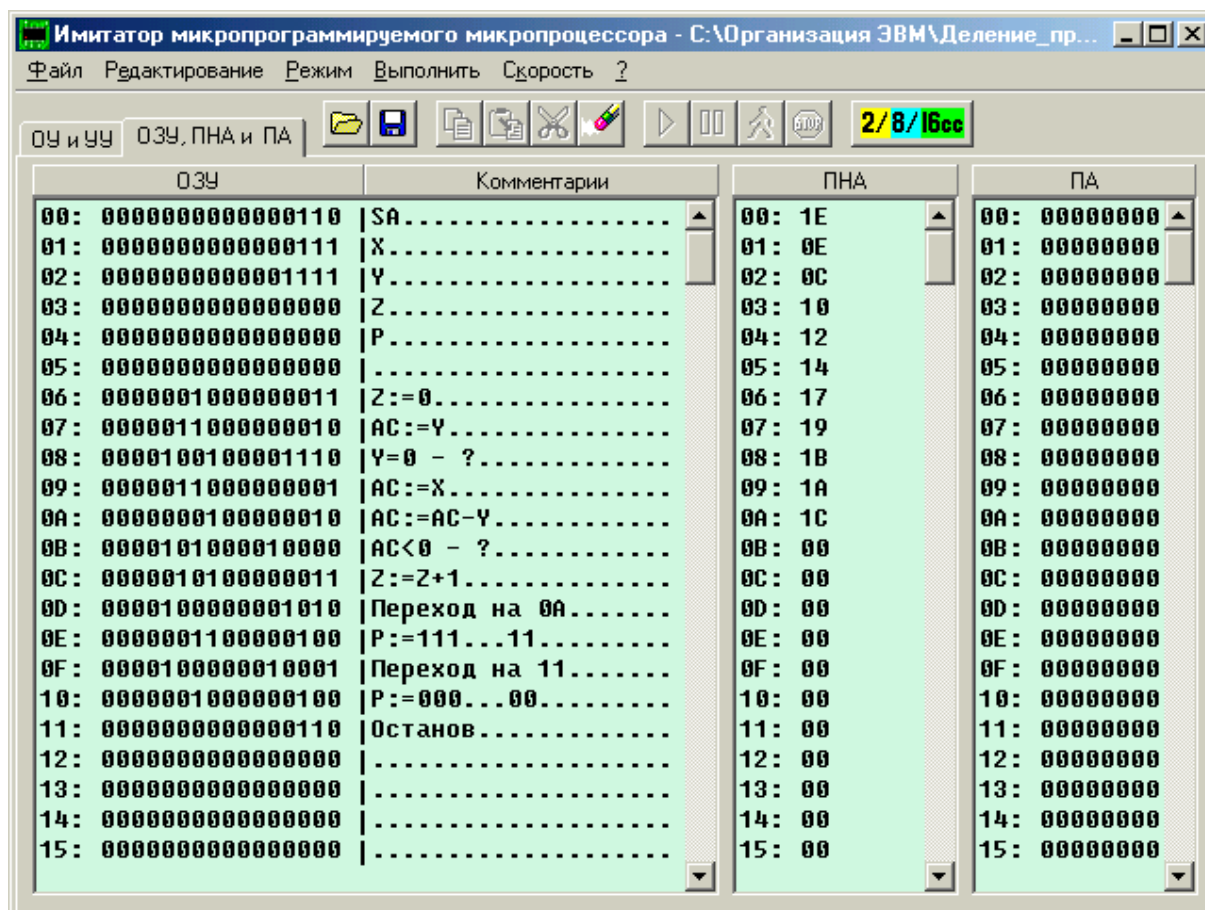


Рис. 5.13. Вкладка «ОЗУ, ПНА и ПА»

### 5.3.7. Расчет производительности и быстродействия

При выполнении лабораторной работы рассчитываются два показателя, характеризующие среднюю скорость обработки информации в ЭВМ: производительность  $W$  и быстродействие  $V$ . Производительность определяется числом задач, решаемых ЭВМ в единицу времени:  $W = 1/T_w$  (задач/сек), где  $T_w$  – среднее время решения задачи. Это время в общем случае может быть вычислено по формуле:

$$T_w = \sum_{j=1}^M P_j T_j, \quad (5.1)$$

где  $P_j$  – вероятность;  $T_j$  – среднее время решения задачи  $j$ -го типа;  $M$  – число типов задач. Если производительность рассчитывается для ЭВМ, решающей однотипные задачи, имеющие одинаковое время решения  $T$ , то  $T_w = T$ . Среднее время решения задачи можно определить с использованием граф-схемы программы по формуле:

$$T = \sum_{i=1}^M h_i t_i, \quad (5.2)$$

где  $h_i$  – среднее число команд  $i$ -го типа, выполняемых в программе при заданных вероятностях ветвлений и числе повторений циклов;  $t_i$  – среднее время выполнения команды  $i$ -го типа;  $M$  – число различных команд. Среднее время выполнения команды  $i$ -го типа может быть рассчитано по микропрограмме командного цикла следующим образом:

$$t_i = b_i \tau^* + d_i T_{zy}, \quad (5.3)$$

где  $b_i$  – среднее число микрокоманд, выполняемых в команде  $i$ -го типа;  $\tau^*$  – время выполнения микрокоманды;  $d_i$  – среднее число обращений к ЗУ при выполнении команды  $i$ -го типа;  $T_{zy}$  – время обращения к ЗУ.

Если время обращения к ЗУ включается во время выполнения микрокоманды, то  $t_i = b_i \tau$ , где  $\tau$  – время выполнения микрокоманды, учитывающее время обращения к ЗУ. Тогда среднее время решение задачи можно определить как  $T = R \tau$ , где  $R$  вычисляется по формуле:

$$R = \sum_{i=1}^M h_i b_i \quad (5.4)$$

Быстродействие определяется числом команд, выполняемых ЭВМ в единицу времени:  $V = \frac{1}{t_V}$  (команд/сек), где  $t_V$  – среднее время выполнения команд.

Время  $t_V$  определяется аналогично времени  $T_w$  по формуле (5.1)

$$t_V = \sum_{i=1}^M p_i t_i, \quad (5.5)$$

где  $p_i = \frac{h_i}{\sum_{i=1}^M h_i}$  – вероятность;  $t_i$  – время выполнения команды  $i$ -го типа;  $i = 1, M$ .

Поскольку время  $t_i$  вычисляется по формуле  $t_i = b_i \tau$ , то среднее время выполнения команды можно представить следующим образом:  $t_V = r \tau$ , где  $r$  вычисляется по формуле:

$$r = \sum_{i=1}^M p_i b_i \quad (5.6)$$

Рассмотрим расчет производительности и быстродействия ЭВМ на примере задания № 4. Допустим, что среднее число циклов в программе деления  $N = 10$ , а вероятность возникновения переполнения  $g = 0,01$ ; кроме того, будем считать, что время обращения к ЗУ включено во время выполнения микрокоманд.

Для расчетов используются граф-схема программы деления (рис. 5.14) и микропрограмма командного цикла (рис. 5.12, табл. 5.6, 5.7).

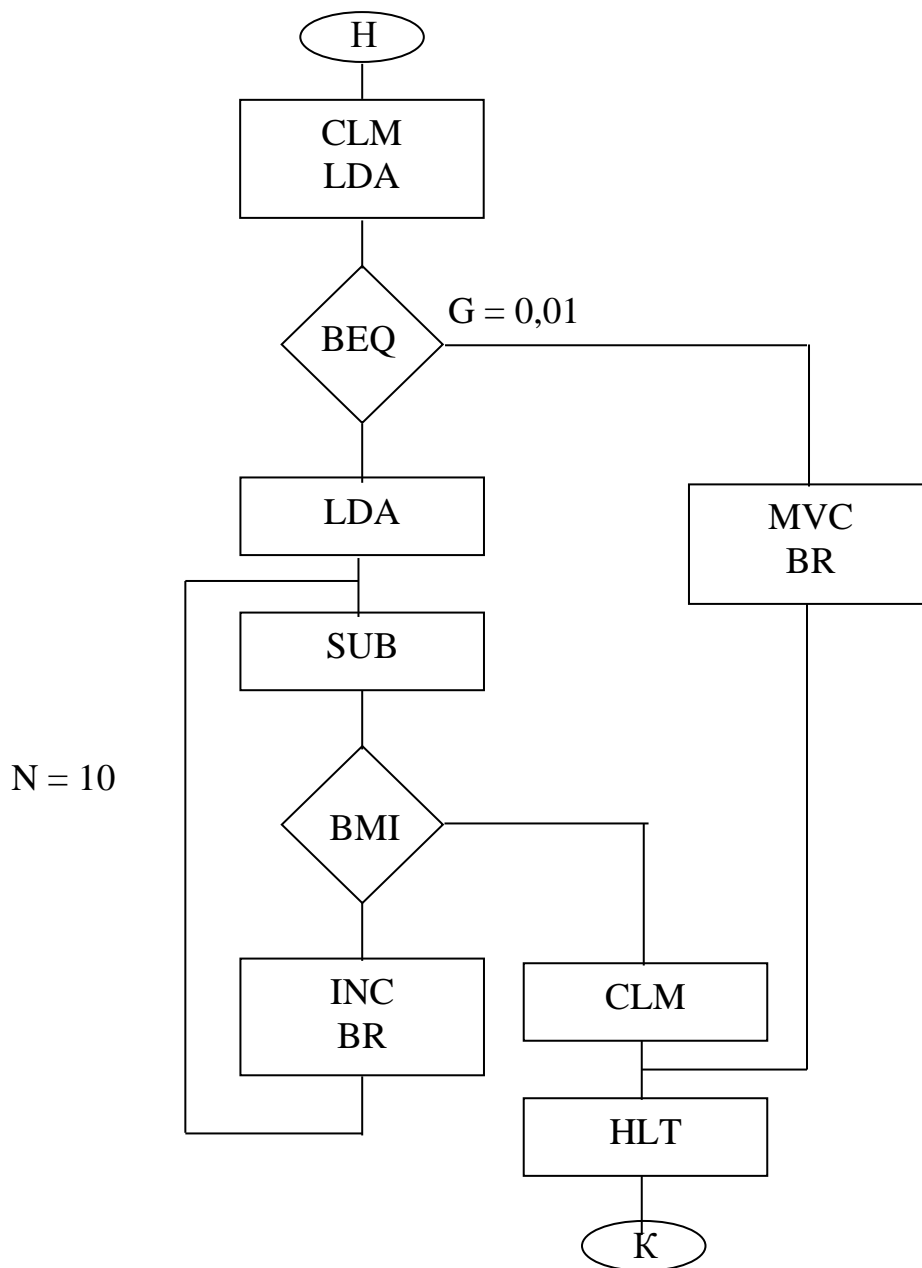


Рис. 5.14. Граф-схема программы деления

Все промежуточные результаты расчетов удобно представить в виде таблицы (табл. 5.8). Очевидно, что для итоговых сумм, приведенных в таблице, должно выполняться равенство  $R = rH$ .

Таблица 5.8

Данные к расчету быстродействия

| Тип команды<br>(мнемоника) | Среднее число команд, $h_i$ | Среднее число микрокоманд, $b_i$ | Произведение $h_i \times b_i$ | Вероятность команды $p_i$ | Произведение $b_i \times p_i$ |
|----------------------------|-----------------------------|----------------------------------|-------------------------------|---------------------------|-------------------------------|
| CLM                        | 1,99                        | 10,00                            | 19,90                         | 0,05                      | 0,46                          |
| LDA                        | 1,99                        | 12,10                            | 24,08                         | 0,05                      | 0,55                          |
| MVC                        | 0,01                        | 10,00                            | 0,10                          | 0,00                      | 0,00                          |
| SUB                        | 9,90                        | 12,10                            | 119,79                        | 0,23                      | 2,75                          |
| INC                        | 8,91                        | 11,00                            | 98,01                         | 0,20                      | 2,25                          |
| BR                         | 8,92                        | 9,00                             | 80,28                         | 0,20                      | 1,84                          |
| BMI                        | 9,90                        | 10,00                            | 99,00                         | 0,23                      | 2,27                          |
| BEQ                        | 1,00                        | 10,00                            | 10,00                         | 0,02                      | 0,23                          |
| HLT                        | 1,00                        | 9,00                             | 9,00                          | 0,02                      | 0,21                          |
| ИТОГО                      | $H \approx 43,62$           |                                  | $R \approx 460,16$            |                           | $r \approx 10,55$             |

Примем  $\tau = 100$  нс. Тогда среднее время выполнения команды составит  $t_V \approx 1$  (мкс), а быстродействие  $V = 1$  (млн. команд/сек). Аналогично среднее время решения задачи –  $T \approx 46$  (мкс), производительность –  $W \approx 22$  (тыс. задач/сек).

## 5.4. Контрольные вопросы

### Основные положения

1. Внутренний язык, включающий систему команд и данных, и программистская структура определяют в процессоре:

- архитектуру;
- организацию;
- программную систему;
- спецификацию программного обеспечения.

2. Состояние регистра признаков (флагов) процессора меняется после выполнения:

- команды определенного вида;
- каждой команды;
- микрокоманды определенного вида;
- каждой микрокоманды.

3. Последовательность из шести цифр, каждая из которых соответствует одному из этапов выполнения команды: 1 – исполнение операции, указанной в команде; 2 – запись результата в память; 3 – выборка операндов из памяти; 4 – формирование адресов операндов; 5 – декодирование команды (определение кода операции и способов адресации); 6 – выборка команды, причем цифры располагаются в порядке выполнения этапов, имеет следующий вид: \_\_\_\_\_ .

4. Длина команд ЭВМ, состоящих из поля кода операции и трех полей номеров регистров при условии, что в командах реализовано 250 операций, а для чтения и записи операндов используется регистровый файл из 256 регистров, составляет \_\_\_\_\_ двоичных разрядов.

5. Минимальное число различных указателей (источников операндов и приемников результатов) в командах, выполняющих двухместные операции (\*) вида:  $A = B * C$  (здесь  $B$  и  $C$  различные числа), равно \_\_\_\_\_ .

6. Минимальное число различных указателей (источников операндов и приемников результатов) в командах, выполняющих одностепенные операции (#) вида:  $A = \#B$ , равно \_\_\_\_\_ .

7. Чем отличается номинальное быстродействие процессора от производительности?

*Исследуемое устройство*

8. Чем отличаются признаки нуля и знака (PZ и PS), формируемые в регистре признаков RP, от признаков нуля и знака (FZ и FS), формируемых в АЛУ?

9. Как осуществляется переход по коду операции, указанному в команде, на соответствующую операции микропрограмму?

10. Производительность ЭВМ, решающей задачи первого класса с вероятностью  $P_1 = 0,1$  за время  $T_1 = 110$  мкс, а задачи второго класса – с вероятностью  $P_2 = 0,9$  за время  $T_2 = 10$  мкс составляет \_\_\_\_\_ тыс. задач / сек.

11. Выполнение каких команд рассматриваемой ЭВМ не зависит от состояния регистра признаков RP?

12. Как в программе деления чисел нацело производится очистка ячеек памяти?





Пример операции, выполняемой командой:  $(AC) := (AC) + M[A_E]$ , где  $M[A_E]$  – содержимое ячейки памяти с адресом  $A_E$ .

*Регистровая (прямая регистровая) адресация.* В коде команды указывается номер  $N_R$  регистра, содержащего операнд (рис. 6.3).

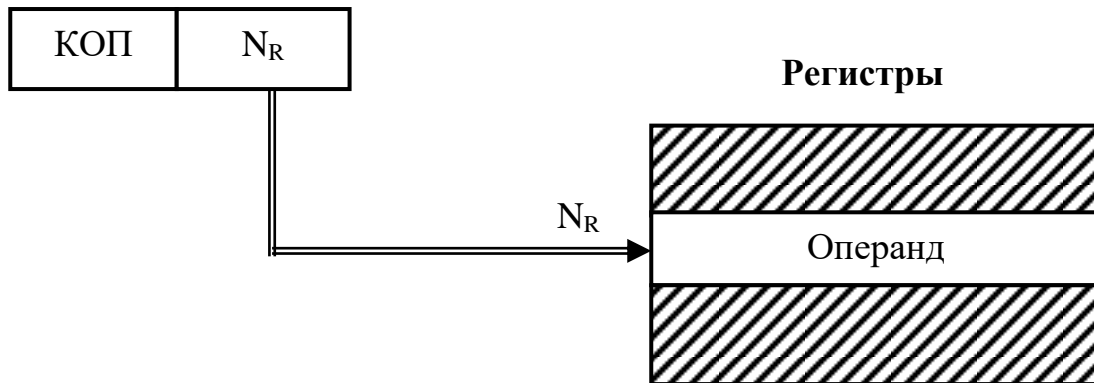


Рис. 6.3. Регистровая (прямая регистровая) адресация

Пример операции, выполняемой командой:  $(AC) := (AC) + R[N_E]$ , где  $R[N_E]$  – содержимое регистра с номером  $N_R$  ( $A_E = N_R$ ).

*Косвенная адресация.* В коде команды при одноступенчатой косвенной адресации задается адрес ( $A$ ) ячейки памяти, в которой хранится адрес операнда ( $A_O$ ) (рис. 6.4).

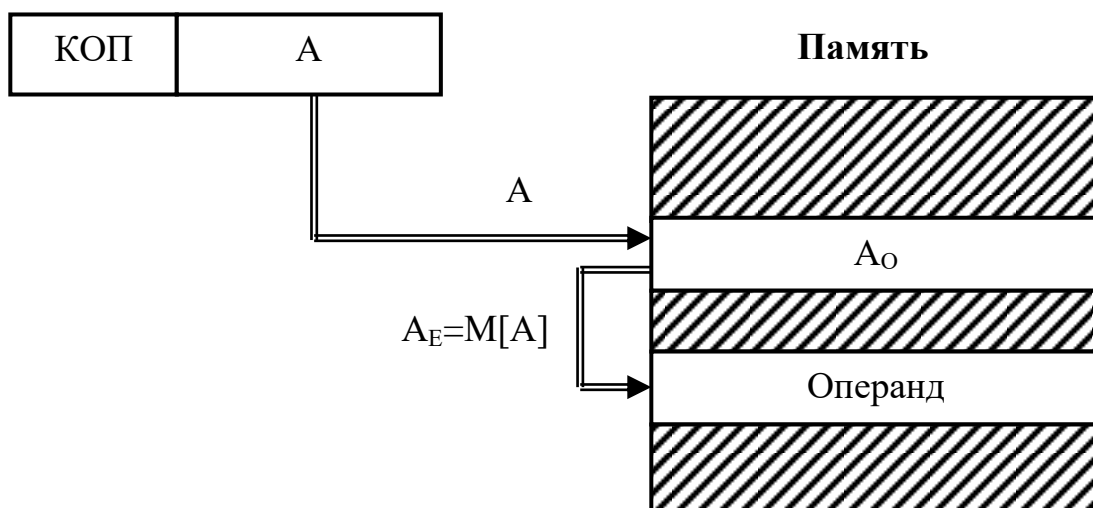


Рис. 6.4. Косвенная адресация

Пример операции, выполняемой командой:  $(AC): = (AC) + M[M[A]]$ , где  $M[M[A]]$  – содержимое ячейки памяти, адрес которой находится в ячейке памяти с адресом  $A$ .

Косвенная адресация может быть многоступенчатой, в том числе и с произвольным числом ступеней:  $(AC): = (AC) + M[\dots M[A]]$ . В последнем случае в ячейке памяти может быть выделен специальный бит, определяющий, что содержит ячейка: адрес операнда или операнд.

*Косвенная регистровая адресация.* В коде команды задается номер  $N_R$  регистра, содержащего адрес ( $A_O$ ) ячейки памяти, в которой хранится операнд (рис. 6.5).

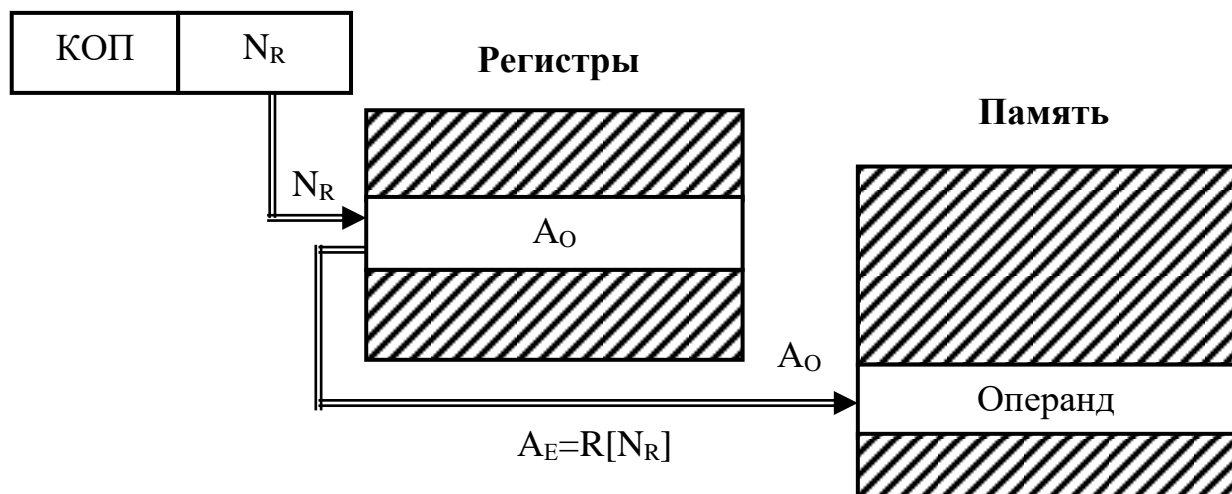


Рис. 6.5. Косвенная регистровая адресация

Пример операции, выполняемой командой:  $(AC): = (AC) + M[R[N_R]]$ , где  $M[R[N_R]]$  – содержимое ячейки памяти, адрес которой указан в регистре  $R[N_R]$  с номером  $N_R$  ( $A_E = R[N_R]$ ).

*Базовая адресация.* Исполнительный адрес вычисляется путем суммирования адреса ( $D$ ) из команды (смещения) и базы ( $B$ ), которая берется из специального регистра базы ( $RB$ ) или регистра общего назначения ( $РОН$ ), выбираемого по номеру, указанному в команде:  $A_E = (RB) + D$  или  $A_E = R[N_R] + D$  (рис. 6.6).

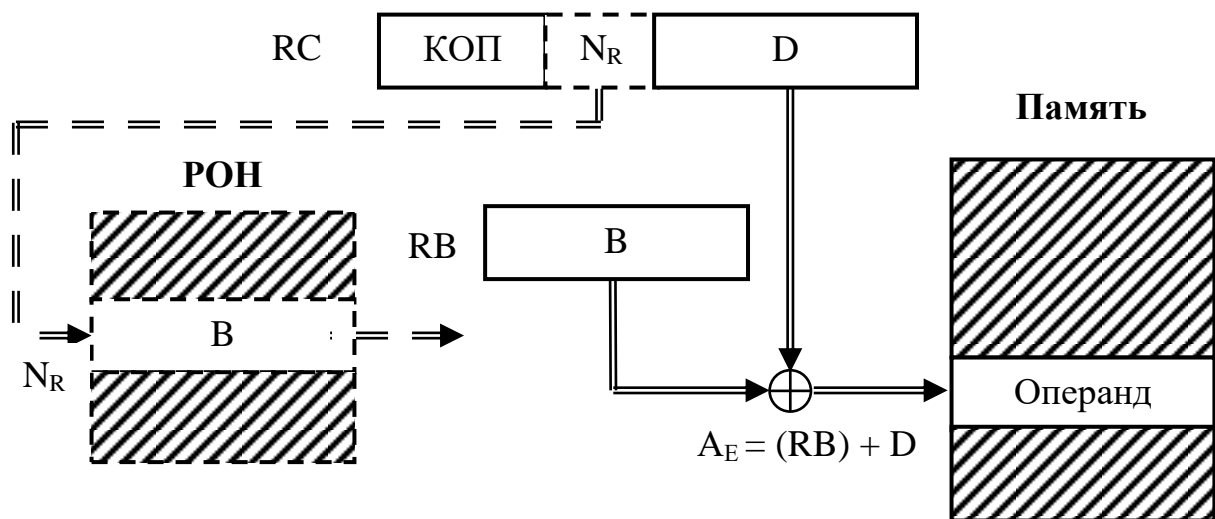


Рис. 6.6. Базовая адресация

Пример операции, выполняемой командой:  $(AC): = (AC) + M[(RB) + D]$  или  $(AC): = (AC) + M[R[N_R] + D]$ .

*Страничная адресация.* Исполнительный адрес вычисляется путем конкатенации (присоединения)  $K$ -разрядного адреса ( $D$ ) операнда на странице, указанного в команде к старшим разрядам адреса ( $A_P$ ), которые рассматриваются как адрес страницы и хранятся в специальном регистре адреса страницы ( $RA_P$ ) (рис. 6.7).

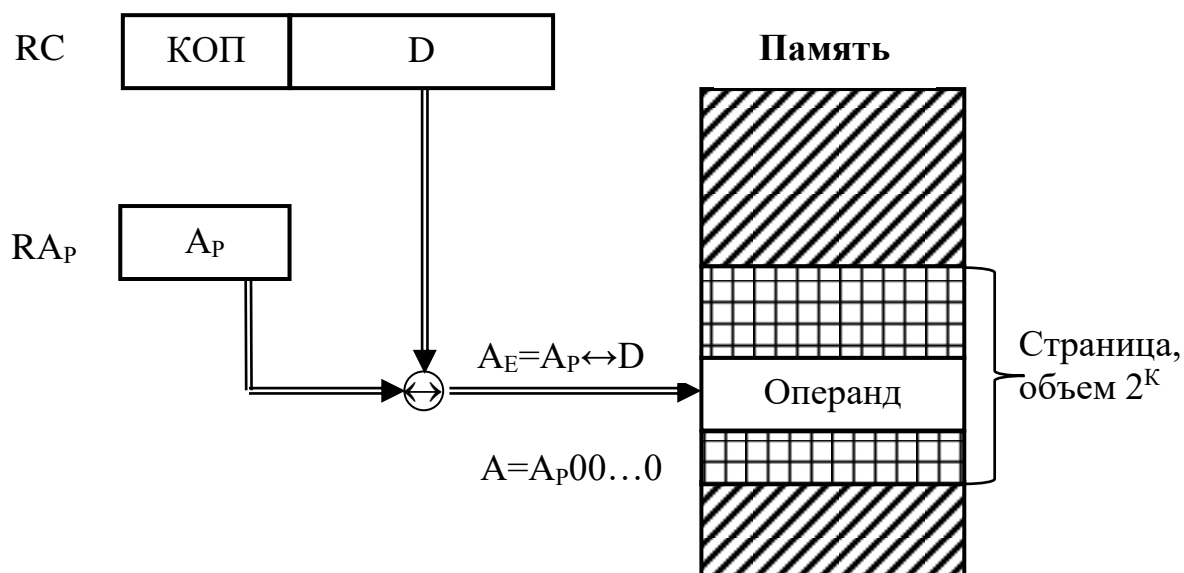


Рис. 6.7. Страничная адресация

Пример операции, выполняемой командой:  $(AC): = (AC) + M[(RA_P) \leftrightarrow D]$ .

Страничную адресацию можно рассматривать как частный случай базовой адресации. Данный вид адресации широко используется при страничной организации памяти.

*Относительная адресация.* Исполнительный адрес вычисляется путем суммирования адреса (D) (смещения) из регистра команд и адреса следующей команды ( $A_K$ ), содержащегося в программном счетчике (PC) (рис. 6.8).

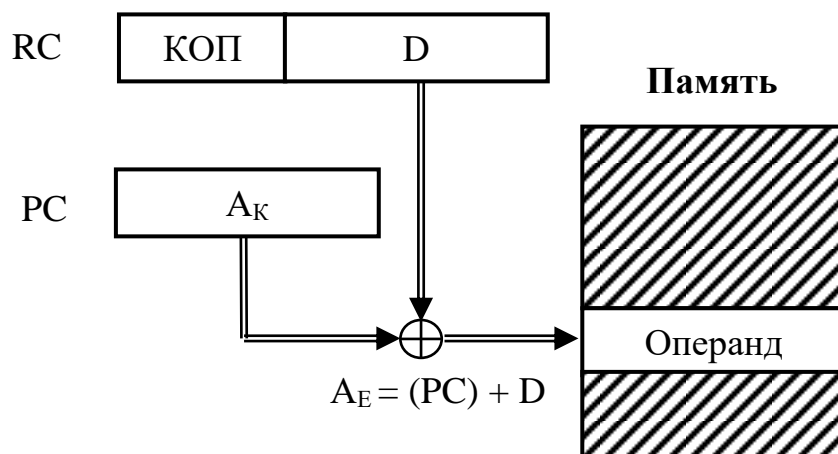


Рис. 6.8. Базовая адресация

Обычно поле D интерпретируется как двоичное число в дополнительном коде, что позволяет, задавать как положительное, так и отрицательное смещение.

*Индексная адресация.* Исполнительный адрес вычисляется путем суммирования адреса (A) из команды и индекса (I), который берется из специального регистра индекса (RI) или РОН. Индекс может быть умножен на масштабный множитель  $m = 1, 2, 4, 8$ .  $A_E = (RI) \times m + A$  или  $A_E = R[N_R] \times m + A$  (рис. 6.9).

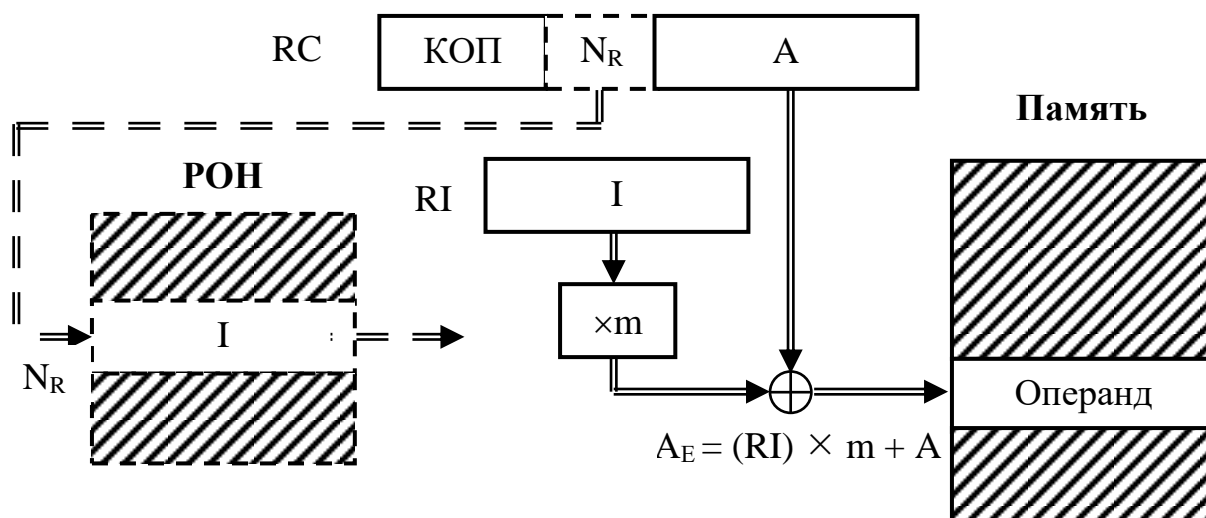


Рис. 6.9. Индексная адресация

Обычно до или после выполнения операции индекс изменяется на определенную константу (С). Уменьшение или увеличение содержимого индексного регистра до или после обращения к нему осуществляется с помощью дополнительной команды. Например, 1)  $(AC): = (AC) + M[(RI) \times 4 + A]$ , 2)  $(RI): = (RI) + C$ .

*Автоиндексация.* Очень часто изменение содержимого регистра происходит на единицу и реализуется «автоматически» микропрограммно в процессе выполнения команды, использующей индексную адресацию. Виды автоиндексации приведены в табл. 6.1.

Таблица 6.1

Виды автоиндексации

| Автоинкрементная                      |                                       | Автодекрементная                      |                                       |
|---------------------------------------|---------------------------------------|---------------------------------------|---------------------------------------|
| Постинкрементная                      | Преинкрементная                       | Постдекрементная                      | Предекрементная                       |
| $A_E = RI + A,$<br>$(RI): = (RI) + 1$ | $(RI): = (RI) + 1,$<br>$A_E = RI + A$ | $A_E = RI + A,$<br>$(RI): = (RI) - 1$ | $(RI): = (RI) - 1,$<br>$A_E = RI + A$ |

### 6.1.2. Особенности задания способов адресации

Можно выделить следующие особенности задания способов адресации.

- Для указания способа адресации в формате команды может быть специальное поле для каждого адреса. Разрядность этого поля определяется числом различных способов формирования адреса, применяемых в команде.
- Исполнительные адреса первого и второго операндов команды могут формироваться различными способами. Например, для первого операнда применяется непосредственная, а для второго – косвенная регистровая адресация.
- При формировании исполнительного адреса операнда может использоваться сочетание различных способов формирования адреса.

Основные способы адресации приведены на рис. 6.10.

|                   |
|-------------------|
| Неявная           |
| Непосредственная  |
| Прямая:           |
| к ячейке памяти;  |
| к регистру.       |
| Косвенная:        |
| к регистру;       |
| к ячейке памяти:  |
| одноступенчатая;  |
| многоступенчатая. |

|                      |
|----------------------|
| Базовая:             |
| с регистром базы;    |
| с общими регистрами. |
| Относительная        |
| Индексная:           |
| общая;               |
| автоинкрементная:    |
| постинкрементная;    |
| преинкрементная;     |
| автодекрементная:    |
| постдекрементная;    |
| предекрементная.     |

Формирование исполнительного адреса операнда путем сочетания различных способов адресации иллюстрируют следующие примеры.

*Косвенная регистровая автоинкрементная адресация.* В коде команды задается номер  $N_R$  регистра, содержащего адрес ( $A_0$ ) ячейки памяти, в которой хранится операнд. После формирования исполнительного адреса содержимое регистра с номером  $N_R$  увеличивается на единицу (рис. 6.11).

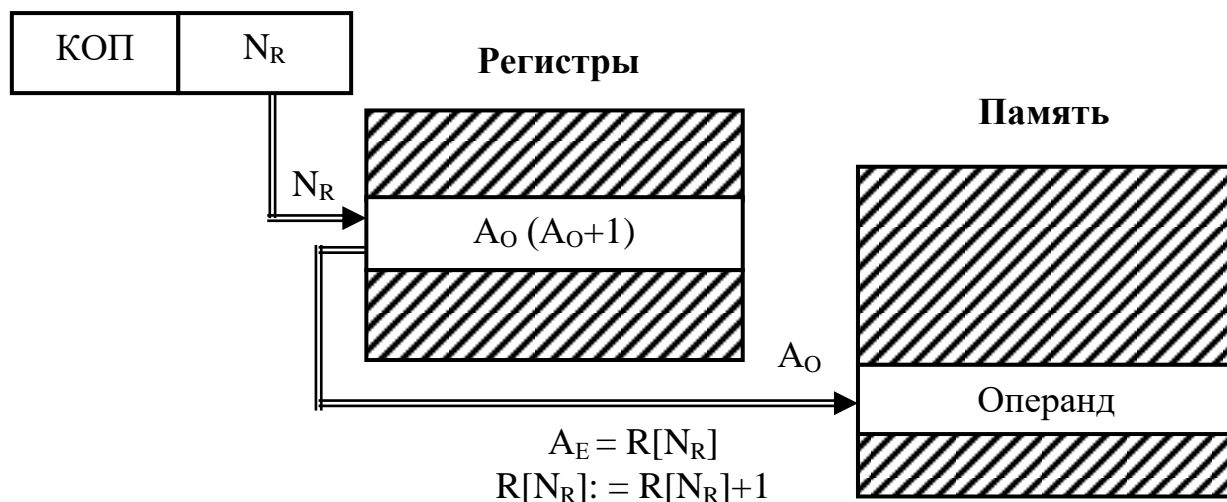


Рис. 6.11. Косвенная регистровая автоинкрементная адресация

Пример операции, выполняемой командой:  $(AC): = (AC) + M[R[N_R]]$ ;  $R[N_R] := R[N_R] + 1$ .

*Базово-индексная адресация.* Исполнительный адрес формируется следующим образом:

$$A_E = (RB) + D + (RI) \times m$$

или

$$A_E = R[N_{RB}] + R[N_{RI}] \times m + D,$$

где  $RB$  – регистр, а  $N_{RB}$  – номер регистра базы;  $RI$  – регистр, а  $N_{RI}$  – номер регистра индекса;  $D$  – смещение адреса, указанное в команде;  $m$  – масштабный множитель для умножения индекса.

*Косвенная автодекрементная адресация.* Исполнительный адрес формируется следующим образом:

$$M[A] := M[A] - 1;$$

$$A_E = M[A].$$

## **6.2. Аппаратные средства для микропрограммной реализации ЭВМ**

Система адресации ЭВМ исследуется на примере учебной ЭВМ, разрабатываемой на основе ЭВМ, реализованной при выполнении предыдущей лабораторной работы. При этом рассматриваются следующие основные вопросы: расширение архитектуры, составление программы с обращением к подпрограмме, распределение памяти программ, разработка алгоритма и микропрограммы командного цикла ЭВМ. Структура аппаратных средств, используемых при выполнении лабораторной работы, приведена на рис. 6.12.

Особенностью аппаратных средств является включение в состав ЭВМ следующих элементов: регистра команд (РК), схемы выбора адреса (СВА), буфера данных (БД). В РК по сигналу LD может быть записан 16-разрядный код с шины адреса/данных. Содержимое РК может быть выдано на шину адреса/данных через буфер данных по сигналу OE.

Схема выбора адреса позволяет задавать адреса выбираемых в ОУ регистров РЗУ на входах А и В, как из соответствующих полей микрокоманды УУ, так и непосредственно из РК. На схему СВА поступают два управляющих сигнала MS1 и MS2. Сигнал MS1 определяет источник адреса для адресного входа А, а сигнал MS2 – для адресного входа В. Если MS1 = 0 (MS2 = 0), то источником адреса по входу А (В) является регистр микрокоманд. Если MS1 = 1 (MS2 = 1), источником адреса по входу А (В) является регистр команд. В программной модели аппаратных средств предусмотрена модификация СВА. С помощью специальной настройки СВА можно выбрать любые последовательно расположенные четыре разряда РК для подключения как по входу А, так и В.



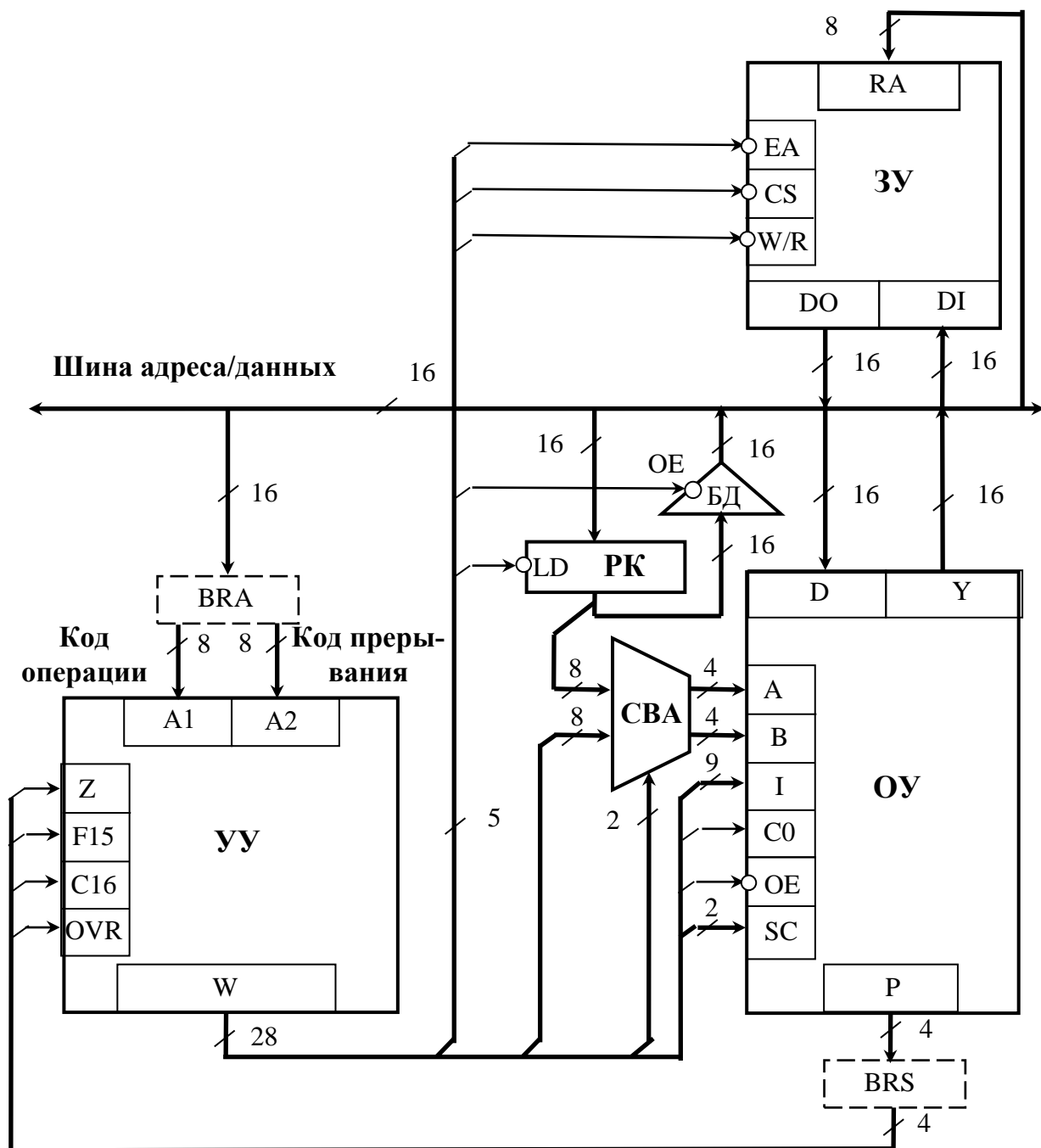


Рис. 6.12. Структурная схема аппаратных средств

### 6.3. Лабораторная работа № 5

Выполнение лабораторной работы складывается из домашней подготовки, экспериментальных исследований на лабораторной установке и оформления отчета.

Для домашней подготовки необходимо получить у преподавателя задание. Домашняя подготовка включает:

- повторение лекционного материала по теме «Системы адресации ЭВМ» и содержания предыдущей лабораторной работы;

- определение архитектуры учебной ЭВМ и разработку основной программы и подпрограммы для решения задачи, приведенной в задании на лабораторную работу;
- разработку граф-схемы алгоритма командного цикла;
- разработку микропрограммы командного цикла с использованием микроинструкций микропроцессора;

Экспериментальные исследования включают: ввод, отладку и выполнение микропрограммы командного цикла по тактам, а программы и подпрограммы – по командным циклам. Отчет по лабораторной работе должен содержать: титульный лист, задание, граф-схему решения задачи, краткое описание архитектуры ЭВМ и текст отлаженной программы, граф-схему алгоритма командного цикла, распределение внутренних регистров, микропрограмму командного цикла. Примером задания может служить задание № 5.

### **6.3.1. Задание № 5**

Определить архитектуру, разработать и отладить микропрограмму командного цикла ЭВМ, составить и выполнить программу вычисления суммы частных  $S$ :

$$S = \sum_{i=1}^N Z_i, \quad Z_i = \left\lfloor \frac{X_i}{Y_i} \right\rfloor,$$

где  $Z_i$  – частное от деления целых чисел  $X_i$  и  $Y_i$  нацело,  $X_i, Y_i, Z_i \in [0 \dots 32767]$ . Кроме результата  $S$ , необходимо формировать и записывать в ОЗУ значения признака переполнения  $Q$ , которые могут возникнуть при делении чисел. Для деления чисел использовать подпрограмму на основе программы, составленной при выполнении предыдущей лабораторной работы. Обмен данными между программой и подпрограммой должен производиться через стек.

### ***6.3.2. Определение архитектуры и программирование***

Архитектура ЭВМ может быть получена путем развития архитектуры учебной ЭВМ, разработанной при выполнении предыдущей лабораторной работы. Дополнительные операции и новые способы адресации определяются в процессе разработки и анализа алгоритма решения задачи с учетом требований, приведенных в задании. Граф-схема алгоритма суммирования частных изображена на рис. 6.13, где  $P: = P(Y)$  – определение признака переполнения при делении.

Разработка архитектуры ЭВМ при выполнении лабораторной работы включает: определение форматов данных, выбор состава программно-доступных регистров и определение системы команд. Рассмотрим решение выделенных задач на примере задания 5.

Форматы данных. Данные в примере являются целыми числами, изменяющимися в пределах от 0 до 32767. В этом случае любое число можно представить 16-разрядным двоичным кодом, старший разряд которого определяет знак числа. Значения признака переполнения  $Q$  предлагается кодировать следующим образом: значению  $Q = 0$  (нет переполнения) соответствует код 000...0, а значению  $Q = 1$  (есть переполнение) – код 111...1.

Программно-доступные регистры. ЭВМ имеет девять программно-доступных регистров: шесть регистров общего назначения ( $r0-r5$ ), программный счетчик – PC ( $r6$ ), регистр признаков – RP ( $r7$ ), содержащий разряды двух признаков: нуля (PZ) и знака (PS), а также регистр указателя стека – rSP ( $r8$ ).

Система команд. Разработка системы команд предполагает определение и набора операций, способов адресаций, модификаций и форматов команд. Для рассматриваемого примера система команд ЭВМ приведена в табл. 6.2.

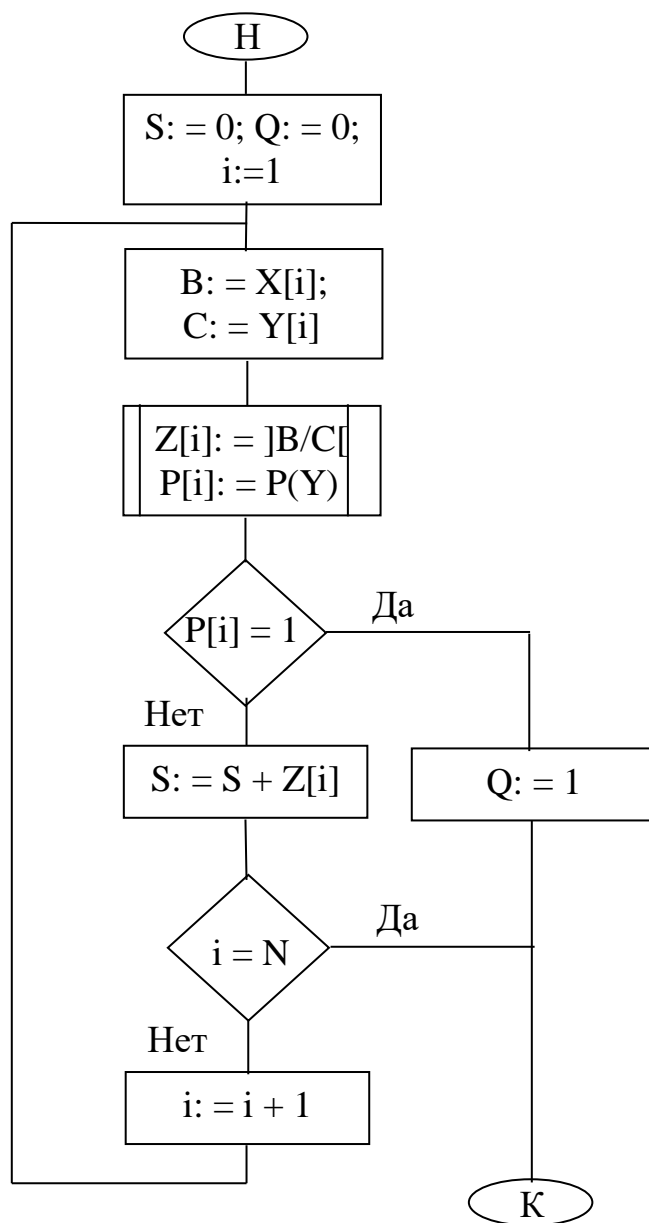


Рис. 6.13. Граф-схема алгоритма суммирования частных

Таблица 6.2

## Система команд

| Наименование                      | Мнемоника      | Описание  | Признаки |    |
|-----------------------------------|----------------|---|----------|----|
|                                   |                |   | PZ       | PS |
| СУММИРОВАНИЕ                      | ADD r r*       | $r := r + r^*, PC := PC + 1$                        | +        | +  |
| ВЫЧИТАНИЕ                         | SUB r r*       | $r := r - r^*, PC := PC + 1$                        | +        | +  |
| ДОБАВЛЕНИЕ C                      | AD r C         | $r := r + C, PC := PC + 1$                          | +        | +  |
| ВЫЧИТАНИЕ C                       | SB r C         | $r := r - C, PC := PC + 1$                          | +        | +  |
| ЧТЕНИЕ В РЕГИСТР                  | LD r A         | $r := M[A], PC := PC + 1$                           | –        | –  |
| ЗАПИСЬ РЕГИСТРА                   | MV r A         | $M[A] := r, PC := PC + 1$                           | –        | –  |
| ЧТЕНИЕ В РЕГИСТР<br>с индексацией | LDI r (r*) +   | $r := M[r^*], r^* := r^* + 1;$<br>$PC := PC + 1$    | +        | +  |
| ЗАПИСЬ В СТЕК                     | PUSH r (rSP) – | $M[rSP] := r;$<br>$rSP := rSP - 1, PC := PC + 1$    | –        | –  |
| ЧТЕНИЕ ИЗ СТЕКА                   | POP r (rSP) +  | $rSP := rSP + 1;$<br>$r := M[rSP], PC := PC + 1$    | –        | –  |
| ПЕРЕХОД                           | BR A           | $PC := A$   | –        | –  |
| ПЕРЕХОД,<br>ЕСЛИ НУЛЬ             | BEQ A          | Если PZ = 1, то $PC := A$ ,<br>иначе $PC := PC + 1$ | –        | –  |
| ПЕРЕХОД,<br>ЕСЛИ МИНУС            | BMI A          | Если PS = 1, то $PC := A$ ,<br>иначе $PC := PC + 1$ | –        | –  |
| ОБРАЩЕНИЕ<br>К ПОДПРОГРАММЕ       | CALL (rSP) – A | $M[rSP] := PC;$<br>$rSP := rSP - 1, PC := A$        | –        | –  |
| ОСТАНОВ                           | HLT A          | $PC := A$ , останов                                 | –        | –  |

В табл. 6.2 использованы следующие обозначения:  $r, r^* \in \{r0, r1, \dots, r8\}$  – программно-доступные регистры: регистр  $r^*$  является источником данных, а регистр  $r$  – приемником результата, но может также служить источником второго операнда;  $M[A]$  – ячейка памяти с адресом  $A$ ; знак «+» в описании признаков означает, что устанавливается новое значение признака по результату выполнения команды, а знак «–» свидетельствует о сохранении старого значения признака.

Система команд содержит команды суммирования ( $\text{ADD } r \ r^*$ ) и вычитания ( $\text{SUB } r \ r^*$ ) содержимого регистров, а также добавления ( $\text{AD } r \ C$ ) и вычитания ( $\text{SB } r \ C$ ) из содержимого регистра константы  $C$ , задаваемой в команде. Предусмотрена команда загрузки в регистр данного из ячейки памяти ( $\text{LD } r \ A$ ) и команда записи содержимого регистра в память ( $\text{MV } r \ A$ ) по адресу, указываемому в команде. Чтение данных в регистр возможно также из ячейки памяти, адрес которой содержится в регистре ( $\text{LDI } r \ (r^*) \ +$ ), при этом номера регистров источника и приемника задаются в команде, а адрес после выполнения операции увеличивается на единицу. В командах чтения из стека ( $\text{POP } r \ (rSP) \ +$ ) и записи в стек ( $\text{PUSH } r \ (rSP) \ -$ ) содержимого регистра в качестве накопителя стека используются область памяти, а указатель стека размещается в специальном регистре –  $rSP$ . Адрес в регистре  $rSP$  указывает на последнюю занятую ячейку стека, а стек «растет» в сторону младших адресов. В систему команд включена команда обращения к подпрограмме ( $\text{CALL } (rSP) \ - \ A$ ). Эта команда сохраняет содержимое программного счетчика в стеке и обеспечивает переход на начальный адрес подпрограммы ( $A$ ). Функция команды возврата из подпрограммы может быть выполнена командой чтения из стека в программный счетчик. Кроме того, как и в ЭВМ, рассматриваемой в предыдущей лабораторной работе, в систему команд включены команды условных переходов ( $\text{BEQ } A$ ,  $\text{BMI } A$ ), команда безусловного перехода ( $\text{BR } A$ ) и команда останова ( $\text{HLT } A$ ).

В ЭВМ используется адресация к памяти и к регистрам процессора. Команды перехода и останова являются одноадресными, остальные команды – двухадресные. Можно выделить следующие виды адресации:

- *Прямая (абсолютная)* – в адресной части команды указан адрес ячейки памяти, к которой происходит обращение при выполнении команды. Применяется в командах:  $\text{LD } r \ A$ ,  $\text{MV } r \ A$  (по второму адресу).
- *Непосредственная* – в адресной части команды содержится операнд, используемый при выполнении команды. Применяется в командах  $\text{AD } r \ C$ ,  $\text{SB } r \ C$  (по второму адресу), а также в командах  $\text{BR } A$ ,  $\text{BEQ } A$ ,  $\text{BMI } A$ ,  $\text{HLT } A$ ,

CALL (rSP)- A (по второму адресу), где в качестве операнда выступает адрес перехода A, над которым выполняется операция пересылки в регистр PC.

- *Регистровая неявная* – номер регистра в команде не указывается, а определяется кодом операции: BEQ A, BMI A, CALL (rSP) –, HLT A – регистр PC.

- *Регистровая прямая* – в адресной части команды содержится номера регистров: ADD r r\*, SUB r r\* (первый и второй адрес), AD r C, SB r C, LD r A, LDI r (r\*) +, PUSH r (rSP) –, POP r (rSP) +, MV r A (первый адрес).

- *Регистровая косвенная автоинкрементная* – в адресной части команды указан номер регистра, содержащего адрес ячейки памяти, после обращения к которой в процессе выполнения команды содержимое регистра увеличивается на единицу. Применяется в командах: LDI r (r\*) +, POP r (rSP) + (второй адрес).

- *Регистровая косвенная автодекрементная* – в адресной части команды указан номер регистра, содержащего адрес ячейки памяти, до обращения к которой в процессе выполнения команды содержимое регистра уменьшается на единицу. Команда: PUSH r (rSP)- (второй адрес), CALL (rSP) – A (первый адрес).

Программа. Программа суммирования частных составляется в соответствии с разработанным алгоритмом (рис. 6.13) с использованием системы команд ЭВМ (табл. 6.2). Программа суммирования частных приведена на рис. 6.14.

В программе приняты следующие обозначения:

AASP – адрес ячейки памяти, в которой находится адрес начала накопителя стека;

AAM – адрес ячейки памяти, в которой находится начальный адрес массива исходных данных (Y1, X1, Y2, X2, ..., YN, XN);

AD – начальный адрес подпрограммы деления чисел нацело;

AS – адрес ячейки, в которую записывается сумма S;

AP – адрес ячейки памяти, в которую помещается значение признака переполнения;

SA – начальный адрес программы суммирования частных.

|                  |   |
|------------------|---|
| LD r8 AASP       | Загрузка регистра указателя стека SP<br>(начальная установка) |
| LD r5 AAM +      | Загрузка адреса массива AM в регистр r5                       |
| LD r4 AN         | Загрузка числа повторений цикла N в регистр r4                |
| SUB r3 r3        | Очистка регистра r3 для суммы S                               |
| SUB r2 r2        | Очистка регистра r2 для признака Q                            |
| m3 LDI r1 (r5) + | Чтение делителя Y в регистр r1                                |
| BEQ m1           | Если PZ = 1 (Y = 0), то переход на метку m1                   |
| LDI r0 (r5) +    | Чтение делимого X в регистр r0                                |
| CALL AD          | Обращение к подпрограмме по адресу AD                         |
| ADD r3 r1        | Суммирование  |
| SB r4 «1»        | Вычитание единицы из числа повторений цикла                   |
| BEQ m2           | Если N = 0, то переход на метку m2                            |
| BR m3            | Переход на метку m3   |
| m1 SB r2 «1»     | Запись единиц в регистр признака Q                            |
| m2 MV r3 AS      | Запись суммы S адресу AS                                      |
| MV r2 AQ         | Запись признака Q адресу AQ                                   |
| HLT SA           | Загрузка PC и останов   |

Рис. 6.14. Программа суммирования частных

Распределение программно-доступных регистров ЭВМ показано на рис. 6.15.

| Регистры ЭВМ |              |                          |
|--------------|--------------|--------------------------|
| r0:          | X            | Делимое X                |
| r1:          | Y (Z)        | Делитель Y (частное Z)   |
| r2:          | Q            | Признак переполнения Q   |
| r3:          | S            | Сумма S                  |
| r4:          | N            | Число повторений цикла N |
| r5:          | AM           | Адрес массива AM         |
| r6:          | PC           | Программный счетчик      |
| r7:          | PS   RP   PZ | Регистр признаков        |
| r8:          | rSP          | Регистр указателя стека  |

Рис. 6.15. Распределение регистров ЭВМ



Для вычисления частных основная программа обращается к подпрограмме деления чисел нацело. Перед обращением делимое  $X$  помещается в регистр  $r0$ , а делитель  $Y$  – в регистр  $r1$ . Подпрограмма возвращает в основную программу частное  $Z$  с помощью регистра  $r1$ . Подпрограмма деления целых чисел нацело приведена на рис. 6.16.

|    |           |  |
|----|-----------|--|
|    | PUSH r2   | Сохранение в стеке PS содержимого регистра Q |
|    | SUB r2 r2 | Очистка регистра для частного Z              |
| m2 | SUB r0 r1 | Вычитание из делимого X делителя Y           |
|    | BMI m1    | Если PS = 1, то переход на метку m1          |
|    | AD r2 «1» | Увеличение на единицу частного Z             |
|    | BR m2     | Переход на метку m2                          |
| m1 | SUB r1 r1 | Очистка регистра rY                          |
|    | ADD r1 r2 | Запись частного Z в регистр rY               |
|    | POP r2    | Восстановление содержимого регистра rQ       |
|    | POP r6    | Возврат из подпрограммы                      |

Рис. 6.16. Подпрограмма деления чисел нацело

Подпрограмма использует для вычисления частного  $Z$  регистр  $r2$ , который занят в основной программе признаком переполнения Q. Поэтому в начале выполнения подпрограммы производится сохранение содержимого регистра  $r2$  в стеке. После завершения вычислений частное перемещается в регистр  $r1$ , а содержимое регистра  $r2$  восстанавливается. Возврат из подпрограммы выполняется с помощью команды чтения данных из стека в программный счетчик ( $r6$ ).

### ***6.3.3. Кодирование программы и распределение памяти программ и данных***

Команды ЭВМ имеют четыре формата и в зависимости от признака формата (Ф) и кода операции (К) делятся на четыре группы (рис. 6.17).

|    |         |        |       |       |                          |
|----|---------|--------|-------|-------|--------------------------|
| 15 | 14...12 | 11...8 | 7...4 | 3...0 |                          |
| 0  | K1      | r      | r*    |       | ADD, SUB, LDI, PUSH, POP |
|    |         |        |       |       |                          |
| 0  | K2      |        | A     |       | BR, BEQ, BMI, HLT        |
|    |         |        |       |       |                          |
| 1  | K3      | r      | C     |       | AD, SB                   |
|    |         |        |       |       |                          |
| 1  | K4      | r      | A     |       | LD, MV, CALL             |

Рис. 6.17. Форматы команд

Наименования, мнемонические обозначения и коды операций приведены в табл. 6.3.

Таблица 6.3

#### Коды операций

| Наименование                      | Мнемоника | Код операции |
|-----------------------------------|-----------|--------------|
| СУММИРОВАНИЕ                      | ADD       | 01           |
| ВЫЧИТАНИЕ                         | SUB       | 02           |
| ДОБАВЛЕНИЕ КОНСТАНТЫ              | AD        | 9            |
| ВЫЧИТАНИЕ КОНСТАНТЫ               | SB        | A            |
| ЧТЕНИЕ В РЕГИСТР                  | LD        | B            |
| ЗАПИСЬ РЕГИСТРА                   | MV        | C            |
| ЧТЕНИЕ В РЕГИСТР<br>с индексацией | LDI       | 10           |
| ЗАПИСЬ В СТЕК                     | PUSH      | 03           |
| ЧТЕНИЕ ИЗ СТЕКА                   | POP       | 04           |
| ПЕРЕХОД                           | BR        | 05           |
| ПЕРЕХОД, ЕСЛИ НУЛЬ                | BEQ       | 06           |
| ПЕРЕХОД, ЕСЛИ МИНУС               | BMI       | 07           |
| ОБРАЩЕНИЕ<br>К ПОДПРОГРАММЕ       | CALL      | 8            |
| ОСТАНОВ                           | HLT       | 00           |

Распределение памяти приведено в табл. 6.4 (основная программа и данные) и табл. 6.5 (подпрограмма и стек).

Таблица 6.4

Распределение памяти (основная программа и данные)

| Адрес                          | Код  | Мнемоника    | Комментарий                                 |
|--------------------------------|------|--------------|---|
| 00                             | 0006 | SA           | Начальный адрес программы                   |
| 01                             | 00FF | ASP          | Начальный адрес области памяти стека        |
| 02                             | 0017 | AM           | Начальный адрес массива                     |
| 03                             | 0004 | N            | Количество пар чисел в массиве              |
| 04                             |      | S            | Результат – сумма частных                   |
| 05                             |      | Q            | Признак переполнения                        |
| Программа суммирования частных |      |              |   |
| 06                             | B801 | LD r8 AASP   | Загрузка регистра указателя стека SP        |
| 07                             | B502 | LD r5 AAM    | Загрузка адреса массива AM в регистр r5     |
| 08                             | B403 | LD r4 AN     | Загрузка числа повторений цикла N           |
| 09                             | 0233 | SUB r3 r3    | Очистка регистра для суммы S                |
| 0A                             | 0222 | SUB r2 r2    | Очистка регистра для признака Q             |
| 0B                             | 1015 | LDI r1 (r5)+ | Чтение делителя Y в регистр r1              |
| 0C                             | 0713 | BEQ m1       | Если PZ = 1 (Y = 0), то переход на метку m1 |
| 0D                             | 1005 | LDI r0 (r5)+ | Чтение делимого X в регистр r0              |
| 0E                             | 8920 | CALL AD      | Обращение к подпрограмме по адресу AD       |
| 0F                             | 0131 | ADD r3 r1    | Суммирование                                |
| 10                             | A401 | SB r4 «1»    | Вычитание единицы из числа N                |
| 11                             | 0614 | BEQ m2       | Если N = 0, то переход на метку m2          |
| 12                             | 050B | BR m3        | Переход на метку m3                         |
| 13                             | A201 | SB r2 «1»    | Запись единиц в регистр признака Q          |
| 14                             | C304 | MV r3 AS     | Запись суммы S адресу AS                    |
| 15                             | C205 | MV r2 AQ     | Запись признака Q адресу AQ                 |
| 16                             | 0007 | HLT SA       | Загрузка PC и останов                       |
| Массив исходных чисел          |      |              |   |
| 17                             |      | Y1           |   |
| 18                             |      | X1           |   |
| 19                             |      | Y2           |   |
| 1A                             |      | X2           |   |
| 1B                             |      | Y3           |   |
| 1C                             |      | X3           |   |
| 1D                             |      | Y4           |   |
| 1E                             |      | X4           |   |

Распределение памяти (подпрограмма и стек)

| Адрес                             | Код  | Мнемоника | Комментарий                            |
|-----------------------------------|------|-----------|--|
| Подпрограмма деления чисел нацело |      |           |  |
| 20                                | 0320 | PUSH r2   | Сохранение содержимого регистра Q      |
| 21                                | 0222 | SUB r2 r2 | Очистка регистра для частного Z        |
| 22                                | 0201 | SUB r0 r1 | Вычитание из делимого X делителя Y     |
| 23                                | 0726 | BMI m1    | Если PS = 1, то переход на метку m1    |
| 24                                | 9201 | AD r2 «1» | Увеличение на единицу частного Z       |
| 25                                | 0522 | BR m2     | Переход на метку m2                    |
| 26                                | 0211 | SUB r1 r1 | Очистка регистра rY                    |
| 27                                | 0112 | ADD r1 r2 | Запись частного Z в регистр r1         |
| 28                                | 0420 | POP r2    | Восстановление содержимого регистра rQ |
| 29                                | 0460 | POP r6    | Возврат из подпрограммы                |
| ...                               |      |           |  |
| Область памяти стека              |      |           |  |
| ...                               |      |           |  |
| FC                                |      |           |  |
| FD                                |      |           |  |
| FE                                |      |           |  |
| FF                                |      |           |  |

#### **6.3.4. Разработка алгоритма работы и микропрограммная реализация ЭВМ**

Алгоритм работы ЭВМ представлен на рис. 6.18 в виде укрупненной граф-схемы микропрограммы командного цикла, содержащей подмикропрограмму выборки команды (ВК), анализ признака формата (Ф), подмикропрограмму распаковки команды (РК) для команд второго формата ( $\Phi = 1$ ), а также подмикропрограммы заданных операций. Подготовка к циклу включает состояние ожидания сигнала пуска (установки специального флага –  $TST = 1$ ) и загрузку начального адреса программы (SA) в программный счетчик.

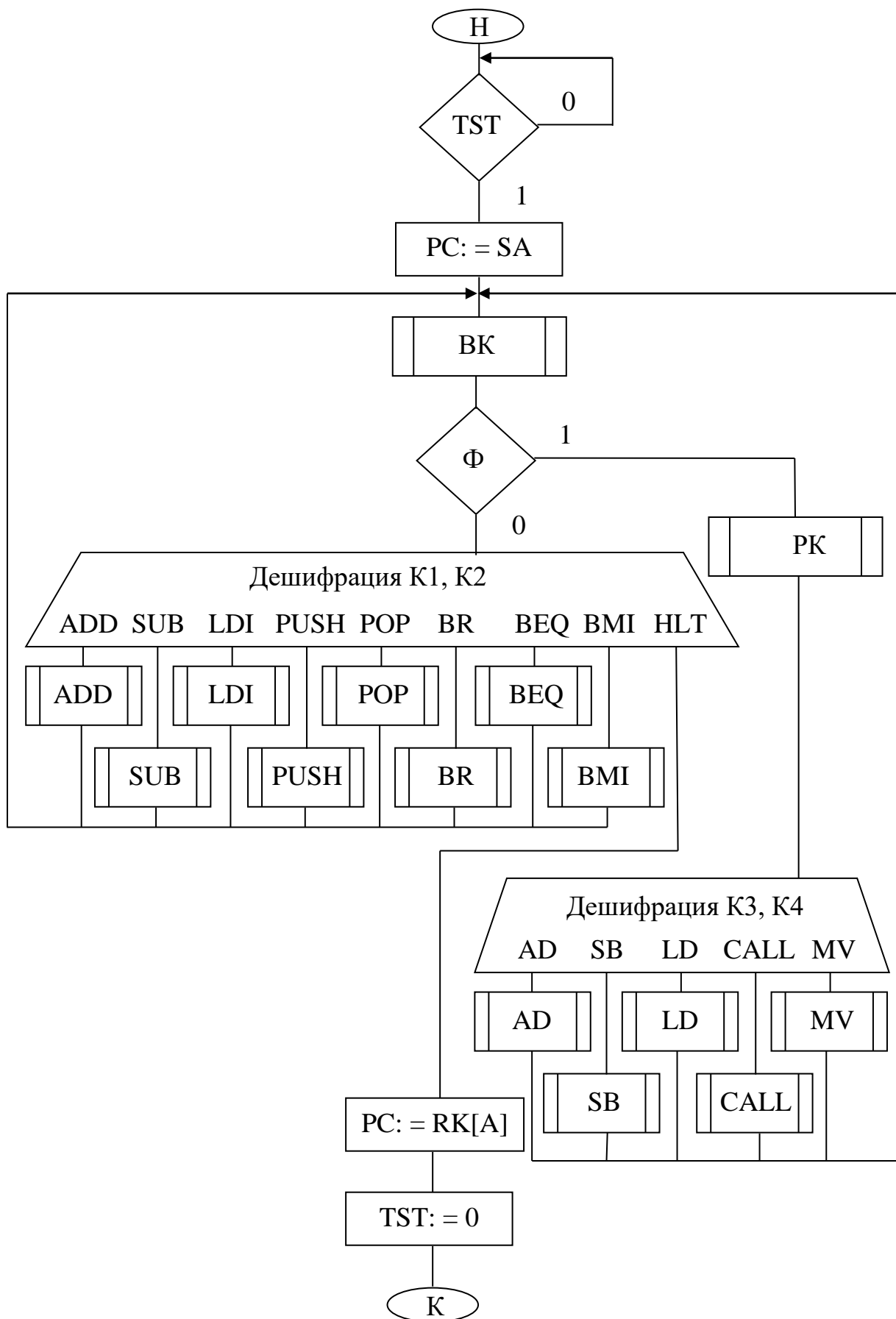


Рис. 6.18. Граф-схема микропрограммы командного цикла ЭВМ

В рассматриваемом примере командный цикл для команд первого формата ( $\Phi = 0$ ) включает три этапа: выборку команды, дешифрацию кода операции и выполнение заданной операции. Команды второго формата выполняются за четыре этапа, так как добавляется еще этап распаковки команд (подмикропрограмма РК). Выход из командного цикла производится при выполнении команды HLT. Микропрограммная реализация ЭВМ включает: распределение внутренних регистров микропроцессора, разработку и кодирование подмикропрограмм командного цикла, а также распределение памяти микропрограмм.

*Регистры ЭВМ.* Распределение внутренних регистров микропроцессора показано на рис. 6.19. Кроме девяти программно-доступных регистров r0-r8 в состав ЭВМ входят пять программно-недоступных регистров (два для команды, и по одному для константы, счетчика адреса ЗУ, операнда Y). Программно-доступные регистры отображаются на регистры (R0-R8), а программно-недоступным регистрам соответствуют регистры R13-R15 и RQ операционного устройства микропроцессора. Кроме того, дополнительными программно-недоступными регистрами являются регистр адреса ЗУ (RA) и регистр команд (RK).

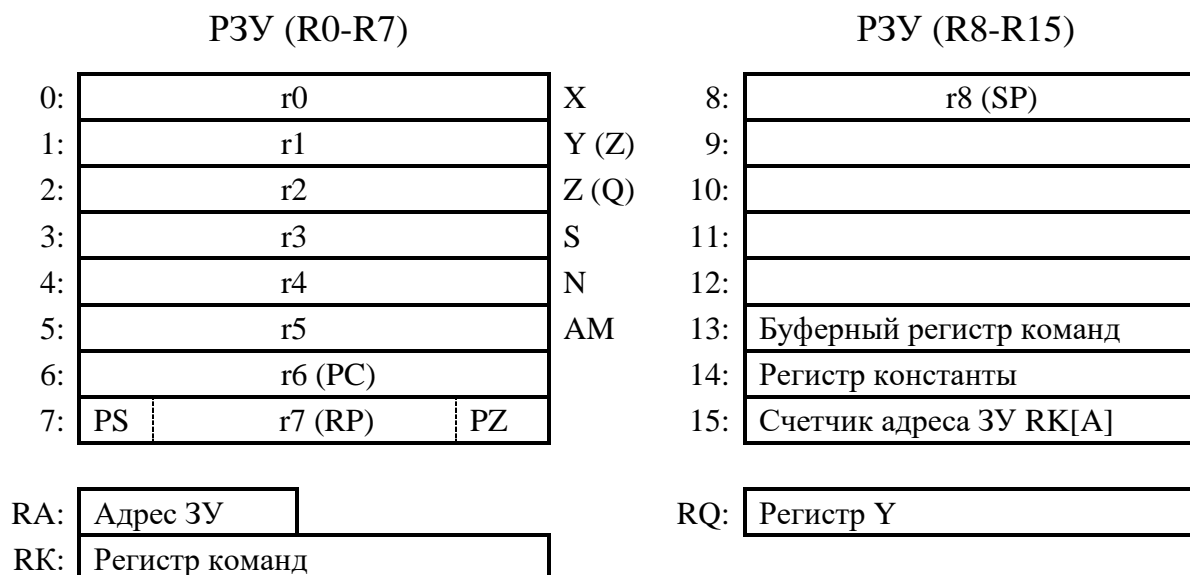


Рис. 6.19. Распределение регистров

*Дешифрация кода операции.* Код операции преобразуется в начальный адрес подмикропрограммы соответствующей операции с помощью преобразо-

вателя начального адреса (ПНА), входящего в состав устройства управления ЭВМ. Соответствие между кодами операций и начальными адресами подмикропрограмм приведены в табл. 6.6.

Таблица 6.6

Коды операций и начальные адреса подмикропрограмм

| Мнемоника | Код операции | Адрес первой МК |
|-----------|--------------|-----------------|
| ADD       | 01           | 1F              |
| SUB       | 02           | 1E              |
| AD        | 9            | 20              |
| SB        | A            | 21              |
| LD        | B            | 1A              |
| MV        | C            | 1C              |
| LDI       | 10           | 22              |
| PUSH      | 03           | 27              |
| POP       | 04           | 29              |
| BR        | 05           | 2C              |
| BEQ       | 06           | 2B              |
| BMI       | 07           | 2D              |
| CALL      | 8            | 30              |
| HLT       | 00           | 35              |

Кодирование микропрограммы командного цикла для ЭВМ производится аналогично кодированию микропрограммы учебной ЭВМ в предыдущей лабораторной работе (см. раздел 6). Особенности кодирования микропрограммы в данной лабораторной работе связаны с наличием дополнительного оборудования: регистра команд (РК) и схемы выбора адреса СВА (рис. 6.12). В рассматриваемом примере принята следующая настройка схемы выбора адреса (рис. 6.20).

| MS1<br>(управляющий<br>вход СВА) | А<br>(адресный вход<br>РЗУ) |
|----------------------------------|-----------------------------|
| 0                                | РМК[А]                      |
| 1                                | РК[3:0]                     |

| MS2<br>(управляющий<br>вход СВА) | В<br>(адресный вход РЗУ) |
|----------------------------------|--------------------------|
| 0                                | РМК[В]                   |
| 1                                | РК[7:4]                  |

Рис. 6.20. Управление коммутацией адресных входов РЗУ

Текст микропрограммы командного цикла ЭВМ, отражающий распределение памяти микропрограмм, приведен в табл. 6.7 (выборка команды, дешифрация кода операции и установка признаков).

Микропрограмма командного цикла начинается с формирования в регистре RE вспомогательной константы 00FF, используемой для выделения младшего байта команды, в котором может находиться адрес операнда или константа (микрокоманды с номерами 00 и 01).

Микрокоманды (МК) 02-04 обеспечивают загрузку начального адреса программы в программный счетчик и увеличение содержимого программного счетчика на единицу.

Таблица 6.7

Микропрограмма командного цикла  
(выборка команды и установка признаков)

| №                                   | МИ  | РЗУ |   | Упр. АЛУ |    |    | Упр. ОЗУ |   |    | Шина | МИ  | Упр. усл. |   |     | Упр. УУ |   |    | Упр. РК |   |    |
|-------------------------------------|-----|-----|---|----------|----|----|----------|---|----|------|-----|-----------|---|-----|---------|---|----|---------|---|----|
| МК                                  | I-9 | A   | B | C0       | OE | SC | CS       | W | EA | D-12 | I-4 | A         | U | CCE | C0      | R | OE | M       | L | OE |
| 00                                  | 571 | E   | E | 0        | 0  | 00 | 1        | 1 | 1  | 006  | C   | 000       | 0 | 0   | 1       | 1 | 0  | 00      | 1 | 1  |
| RE: = 0111111111111111; PA/CI: = 6  |     |     |   |          |    |    |          |   |    |      |     |           |   |     |         |   |    |         |   |    |
| 01                                  | 533 | 0   | E | 0        | 0  | 00 | 1        | 1 | 1  | 001  | 9   | 000       | 0 | 0   | 1       | 1 | 0  | 00      | 1 | 1  |
| RE – сдвиг вправо; PA/CI: = PA/CI-1 |     |     |   |          |    |    |          |   |    |      |     |           |   |     |         |   |    |         |   |    |
| 02                                  | 143 | 0   | 6 | 0        | 0  | 00 | 1        | 1 | 0  | 000  | E   | 000       | 0 | 0   | 1       | 1 | 0  | 00      | 1 | 1  |
| RA: = 0                             |     |     |   |          |    |    |          |   |    |      |     |           |   |     |         |   |    |         |   |    |
| 03                                  | 337 | 0   | 6 | 0        | 1  | 00 | 0        | 1 | 0  | 000  | E   | 000       | 0 | 0   | 1       | 1 | 0  | 00      | 1 | 1  |
| R6: = SA (PC: = SA)                 |     |     |   |          |    |    |          |   |    |      |     |           |   |     |         |   |    |         |   |    |



Окончание табл. 6.7

| №  | МИ  | РЗУ |   | Упр. АЛУ |                 |    | Упр. ОЗУ        |                |                 | Шина | МИ  | Упр. усл. |   |                  | Упр. УУ |                |                 | Упр. РК |   |                 |
|--|-----|-----|---|----------|-----------------|----|-----------------|----------------|-----------------|------|-----|-----------|---|------------------|---------|----------------|-----------------|---------|---|-----------------|
| МК   | I-9 | A   | B | C0       | $\overline{OE}$ | SC | $\overline{CS}$ | $\overline{W}$ | $\overline{EA}$ | D-12 | I-4 | A         | U | $\overline{CCE}$ | C0      | $\overline{R}$ | $\overline{OE}$ | M       | L | $\overline{OE}$ |
| 04   | 203 | 6   | 6 | 1        | 0               | 00 | 1               | 1              | 0               | 000  | E   | 000       | 0 | 0                | 1       | 1              | 0               | 00      | 1 | 1               |
| RA: = R6; R6: = R6 + 1 (RA: = PC; PC: = PC + 1)  |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                |                 |         |   |                 |
| 05   | 337 | 0   | C | 0        | 1               | 00 | 0               | 1              | 1               | 007  | 3   | 001       | 1 | 0                | 1       | 1              | 0               | 00      | 0 | 1               |
| RK: = K; RC: = K                                 |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                |                 |         |   |                 |
| 06   | 345 | E   | F | 0        | 1               | 00 | 1               | 1              | 1               | 000  | 2   | 000       | 0 | 0                | 1       | 1              | 0               | 00      | 1 | 0               |
| RF: = K[A]      Переход по КОП                   |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                |                 |         |   |                 |
| 07   | 345 | E   | F | 0        | 1               | 00 | 1               | 1              | 1               | 003  | C   | 000       | 0 | 0                | 1       | 1              | 0               | 00      | 1 | 0               |
| RF: = K[A]      PA/CI: = 3                       |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                |                 |         |   |                 |
| 08   | 533 | 0   | C | 0        | 0               | 00 | 1               | 1              | 1               | 008  | 9   | 000       | 0 | 0                | 1       | 1              | 0               | 00      | 1 | 1               |
| RC – сдвиг вправо; PA/CI: = PA/CI-1              |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                |                 |         |   |                 |
| 09   | 131 | C   | C | 0        | 0               | 00 | 1               | 1              | 1               | 000  | 2   | 000       | 0 | 0                | 1       | 1              | 0               | 00      | 0 | 1               |
| RK: = RC;      Переход по КОП                    |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                |                 |         |   |                 |
| 0A   | 343 | 0   | 7 | 0        | 1               | 00 | 1               | 1              | 1               | 000  | E   | 000       | 0 | 0                | 1       | 1              | 0               | 00      | 1 | 1               |
| R7: = 0 (RP: = 0) Формирование признаков         |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                |                 |         |   |                 |
| 0B   | 133 | 0   | 0 | 0        | 1               | 00 | 1               | 1              | 1               | 00E  | 3   | 000       | 1 | 0                | 1       | 1              | 0               | 01      | 1 | 1               |
| Переход, если r = 0                              |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                |                 |         |   |                 |
| 0C   | 133 | 0   | 0 | 0        | 1               | 00 | 1               | 1              | 1               | 004  | 3   | 001       | 0 | 0                | 1       | 1              | 0               | 01      | 1 | 1               |
| Переход, если r[15] = 0                          |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                |                 |         |   |                 |
| 0D   | 503 | 0   | 7 | 1        | 1               | 01 | 1               | 1              | 1               | 004  | 3   | 000       | 0 | 1                | 1       | 1              | 0               | 00      | 1 | 1               |
| R7: = R7 + 1; циклический сдвиг вправо (PS: = 1) |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                |                 |         |   |                 |
| 0E   | 303 | 0   | 7 | 1        | 1               | 00 | 1               | 1              | 1               | 004  | 3   | 000       | 0 | 1                | 1       | 1              | 0               | 00      | 1 | 1               |
| R7: = R7 + 1 (PZ: = 1)                           |     |     |   |          |                 |    |                 |                |                 |      |     |           |   |                  |         |                |                 |         |   |                 |
| 0F   | ... | .   | . | .        | .               | .. | .               | .              | .               | ...  | .   | ...       | . | .                | .       | .              | .               | ..      | . | .               |

Чтение команды в регистр команд (RK) выполняет МК05, которая также анализирует признак формата (Φ). При Φ = 0 выполняется следующая по порядку микрокоманда (МК06), иначе МК07. Микрокоманда МК06 обеспечивает дешифрацию кода операции, для команд первого формата, а также выделяет адресную часть команды и заносит ее в регистр RF. Подмикропрограмма рас-

паковки команд второго формата ( $\Phi = 1$ ) включает МК07, 08. Эта подмикропрограмма выделяет младший байт команды и производит сдвиг команды в регистре RC на четыре разряда в сторону младших разрядов. С помощью МК09 осуществляется занесение преобразованной команды в регистр команд RK и дешифрация кода операции для команд второго формата.

Подмикропрограмма PZS формирования значений признаков нуля (PZ) и знака (PS) состоит из пяти микрокоманд (МК0А-МК0Е).

Коды подмикропрограмм операций приведены в табл. 6.8. Начальные адреса подмикропрограмм соответствуют распределению памяти преобразователя начального адреса (табл. 6.6). В виду простоты операций большая часть из них требует для реализации 1–2 микрокоманд.

Таблица 6.8

Микропрограмма командного цикла (выполнение операций)

| №                       | МИ   | РЗУ |   | Упр. АЛУ |    |    | Упр. ОЗУ |   |    | Шина  | МИ   | Упр. усл. |   |     | Упр. УУ |   |    | Упр. РК |   |    |
|-------------------------|------|-----|---|----------|----|----|----------|---|----|-------|------|-----------|---|-----|---------|---|----|---------|---|----|
| МК                      | I8-0 | A   | B | C0       | OE | SC | CS       | W | EA | D11-0 | I3-0 | A         | U | CCE | C0      | R | OE | M       | L | OE |
| 1A                      | 133  | 0   | F | 0        | 0  | 00 | 1        | 1 | 0  | 000   | E    | 000       | 0 | 0   | 1       | 1 | 0  | 00      | 1 | 1  |
| RA: = RF[A] (LD r A)    |      |     |   |          |    |    |          |   |    |       |      |           |   |     |         |   |    |         |   |    |
| 1B                      | 337  | 0   | 0 | 0        | 1  | 00 | 0        | 1 | 1  | 00A   | 3    | 000       | 1 | 1   | 1       | 1 | 0  | 01      | 1 | 1  |
| r: = M[A]               |      |     |   |          |    |    |          |   |    |       |      |           |   |     |         |   |    |         |   |    |
| 1C                      | 133  | 0   | F | 0        | 0  | 00 | 1        | 1 | 0  | 000   | E    | 000       | 0 | 0   | 1       | 1 | 0  | 00      | 1 | 1  |
| RA: = RF[A] (MV r A)    |      |     |   |          |    |    |          |   |    |       |      |           |   |     |         |   |    |         |   |    |
| 1D                      | 133  | 0   | 0 | 0        | 0  | 00 | 0        | 0 | 1  | 00A   | 3    | 000       | 0 | 1   | 1       | 1 | 0  | 01      | 1 | 1  |
| M[A]: = r               |      |     |   |          |    |    |          |   |    |       |      |           |   |     |         |   |    |         |   |    |
| 1E                      | 311  | 0   | 0 | 1        | 0  | 00 | 1        | 1 | 1  | 00A   | 3    | 000       | 0 | 1   | 1       | 1 | 0  | 11      | 1 | 1  |
| r: = r - r* (SUB r r*)  |      |     |   |          |    |    |          |   |    |       |      |           |   |     |         |   |    |         |   |    |
| 1F                      | 301  | 0   | 0 | 0        | 0  | 00 | 1        | 1 | 1  | 00A   | 3    | 000       | 0 | 1   | 1       | 1 | 0  | 11      | 1 | 1  |
| r: = r - r* (ADD r r*)  |      |     |   |          |    |    |          |   |    |       |      |           |   |     |         |   |    |         |   |    |
| 20                      | 301  | F   | 0 | 0        | 0  | 00 | 1        | 1 | 1  | 00A   | 3    | 000       | 0 | 1   | 1       | 1 | 0  | 01      | 1 | 1  |
| r: = r + C (AD r C)     |      |     |   |          |    |    |          |   |    |       |      |           |   |     |         |   |    |         |   |    |
| 21                      | 311  | F   | 0 | 1        | 0  | 00 | 1        | 1 | 1  | 00A   | 3    | 000       | 0 | 1   | 1       | 1 | 0  | 01      | 1 | 1  |
| r: = r - C (SB r C)     |      |     |   |          |    |    |          |   |    |       |      |           |   |     |         |   |    |         |   |    |
| 22                      | 104  | 0   | 0 | 0        | 0  | 00 | 1        | 1 | 0  | 000   | E    | 000       | 0 | 0   | 1       | 1 | 0  | 11      | 1 | 1  |
| RA: = r*; (LDI r (r*)+) |      |     |   |          |    |    |          |   |    |       |      |           |   |     |         |   |    |         |   |    |

| №                                     | МИ       | РЗУ |   | Упр. АЛУ |                 |    | Упр. ОЗУ        |                |                 | Шина      | МИ       | Упр. усл. |   |                  | Упр. УУ |                |                 | Упр. РК |   |                 |
|---------------------------------------|----------|-----|---|----------|-----------------|----|-----------------|----------------|-----------------|-----------|----------|-----------|---|------------------|---------|----------------|-----------------|---------|---|-----------------|
| МК                                    | I8-<br>0 | A   | B | C0       | $\overline{OE}$ | SC | $\overline{CS}$ | $\overline{W}$ | $\overline{EA}$ | D11-<br>0 | I3-<br>0 | A         | U | $\overline{CCE}$ | C0      | $\overline{R}$ | $\overline{OE}$ | M       | L | $\overline{OE}$ |
| 23                                    | 337      | 0   | 0 | 0        | 1               | 00 | 0               | 1              | 1               | 004       | C        | 000       | 0 | 1                | 1       | 1              | 0               | 11      | 1 | 1               |
| r: = M[r*] PA/CI: = 4                 |          |     |   |          |                 |    |                 |                |                 |           |          |           |   |                  |         |                |                 |         |   |                 |
| 24                                    | 733      | C   | C | 0        | 0               | 00 | 1               | 1              | 1               | 024       | 9        | 000       | 0 | 0                | 1       | 1              | 0               | 00      | 0 | 1               |
| RC – сдвиг влево PA/CI: = PA/CI – 1   |          |     |   |          |                 |    |                 |                |                 |           |          |           |   |                  |         |                |                 |         |   |                 |
| 25                                    | 303      | 0   | 0 | 1        | 1               | 00 | 1               | 1              | 1               | 000       | E        | 000       | 0 | 0                | 1       | 1              | 0               | 11      | 1 | 1               |
| r* = r* + 1                           |          |     |   |          |                 |    |                 |                |                 |           |          |           |   |                  |         |                |                 |         |   |                 |
| 26                                    | 133      | 0   | F | 0        | 0               | 00 | 1               | 1              | 1               | 00A       | 3        | 000       | 0 | 1                | 1       | 1              | 0               | 00      | 0 | 1               |
| RK: = RF (для формирования признаков) |          |     |   |          |                 |    |                 |                |                 |           |          |           |   |                  |         |                |                 |         |   |                 |
| 27                                    | 213      | 8   | 8 | 0        | 0               | 00 | 1               | 1              | 0               | 000       | E        | 000       | 0 | 1                | 1       | 1              | 0               | 00      | 1 | 1               |
| RA: = SP; SP: = SP – 1 (PUSH r)       |          |     |   |          |                 |    |                 |                |                 |           |          |           |   |                  |         |                |                 |         |   |                 |
| 28                                    | 133      | 0   | 0 | 0        | 0               | 00 | 0               | 0              | 1               | 004       | 3        | 000       | 0 | 1                | 1       | 1              | 0               | 11      | 1 | 1               |
| M[SP]: = r                            |          |     |   |          |                 |    |                 |                |                 |           |          |           |   |                  |         |                |                 |         |   |                 |
| 29                                    | 303      | 0   | 8 | 1        | 0               | 00 | 1               | 1              | 0               | 000       | E        | 000       | 0 | 1                | 1       | 1              | 0               | 00      | 1 | 1               |
| SP: = SP + 1; RA: = SP (POP r)        |          |     |   |          |                 |    |                 |                |                 |           |          |           |   |                  |         |                |                 |         |   |                 |
| 2A                                    | 337      | 0   | 0 | 0        | 1               | 00 | 0               | 1              | 1               | 004       | 3        | 000       | 0 | 1                | 1       | 1              | 0               | 11      | 1 | 1               |
| r: = M[SP]                            |          |     |   |          |                 |    |                 |                |                 |           |          |           |   |                  |         |                |                 |         |   |                 |
| 2B                                    | 113      | 0   | 7 | 0        | 1               | 00 | 1               | 1              | 1               | 004       | 3        | 000       | 0 | 0                | 1       | 1              | 0               | 00      | 1 | 1               |
| Переход, если R7[0] = 0 (BEQ A)       |          |     |   |          |                 |    |                 |                |                 |           |          |           |   |                  |         |                |                 |         |   |                 |
| 2C                                    | 334      | F   | 6 | 0        | 1               | 00 | 1               | 1              | 1               | 004       | 3        | 000       | 0 | 1                | 1       | 1              | 0               | 00      | 1 | 1               |
| R6: = RF (PC: = RK[A]) (BR A)         |          |     |   |          |                 |    |                 |                |                 |           |          |           |   |                  |         |                |                 |         |   |                 |
| 2D                                    | 113      | 0   | 7 | 0        | 1               | 00 | 1               | 1              | 1               | 004       | 3        | 001       | 1 | 0                | 1       | 1              | 0               | 00      | 1 | 1               |
| Переход, если R7[15] = 0 (BMI A)      |          |     |   |          |                 |    |                 |                |                 |           |          |           |   |                  |         |                |                 |         |   |                 |
| 2E                                    | 334      | F   | 6 | 0        | 1               | 00 | 1               | 1              | 1               | 004       | 3        | 000       | 0 | 1                | 1       | 1              | 0               | 00      | 1 | 1               |
| R6: = RF (PC: = RK[A])                |          |     |   |          |                 |    |                 |                |                 |           |          |           |   |                  |         |                |                 |         |   |                 |
| 2F                                    | 334      | F   | 6 | 0        | 1               | 00 | 1               | 1              | 1               | 004       | 3        | 000       | 0 | 1                | 1       | 1              | 1               | 00      | 1 | 1               |
| R6: = RF (PC: = RK[A]) (HLT A)        |          |     |   |          |                 |    |                 |                |                 |           |          |           |   |                  |         |                |                 |         |   |                 |
| 30                                    | 213      | 8   | 8 | 0        | 0               | 00 | 1               | 1              | 0               | 000       | E        | 000       | 0 | 1                | 1       | 1              | 0               | 00      | 1 | 1               |
| RA: = SP, SP: = SP – 1 (CALL A)       |          |     |   |          |                 |    |                 |                |                 |           |          |           |   |                  |         |                |                 |         |   |                 |
| 31                                    | 133      | 0   | 6 | 0        | 0               | 00 | 0               | 0              | 1               | 000       | E        | 000       | 0 | 1                | 1       | 1              | 0               | 00      | 1 | 1               |
| M[SP]: = r                            |          |     |   |          |                 |    |                 |                |                 |           |          |           |   |                  |         |                |                 |         |   |                 |
| 32                                    | 334      | F   | 6 | 0        | 1               | 00 | 1               | 1              | 1               | 004       | 3        | 000       | 0 | 1                | 1       | 1              | 0               | 00      | 1 | 1               |
| R6: = RF (PC: = RK[A])                |          |     |   |          |                 |    |                 |                |                 |           |          |           |   |                  |         |                |                 |         |   |                 |
| 33                                    | ...      | .   | . | .        | .               | .. | .               | .              | .               | ...       | .        | ...       | . | .                | .       | .              | .               | ..      | . | .               |

### 6.3.5. Ввод и отладка микропрограммы командного цикла и программы решения задачи

В лабораторной работе используется программа «Имитатор микропрограммируемой микроЭВМ». В программе создаются файлы с расширением «.mvm», содержащие двоичные коды микропрограмм, программ и данных, записанных в ЗУ, ПНА и ПА, а также настройки параметров имитатора. Интерфейс пользователя этой программы имеет много общего с интерфейсом программы «Имитатор микропрограммируемого микропроцессора», используемой в предыдущей лабораторной работе. После загрузки и запуска программы для лабораторных исследований она переводится в режим ввода микропрограмм, в котором производится ввод разработанной микропрограммы командного цикла. Отладка микропрограммы осуществляется в режиме выполнения микрокоманд (рис. 6.21).

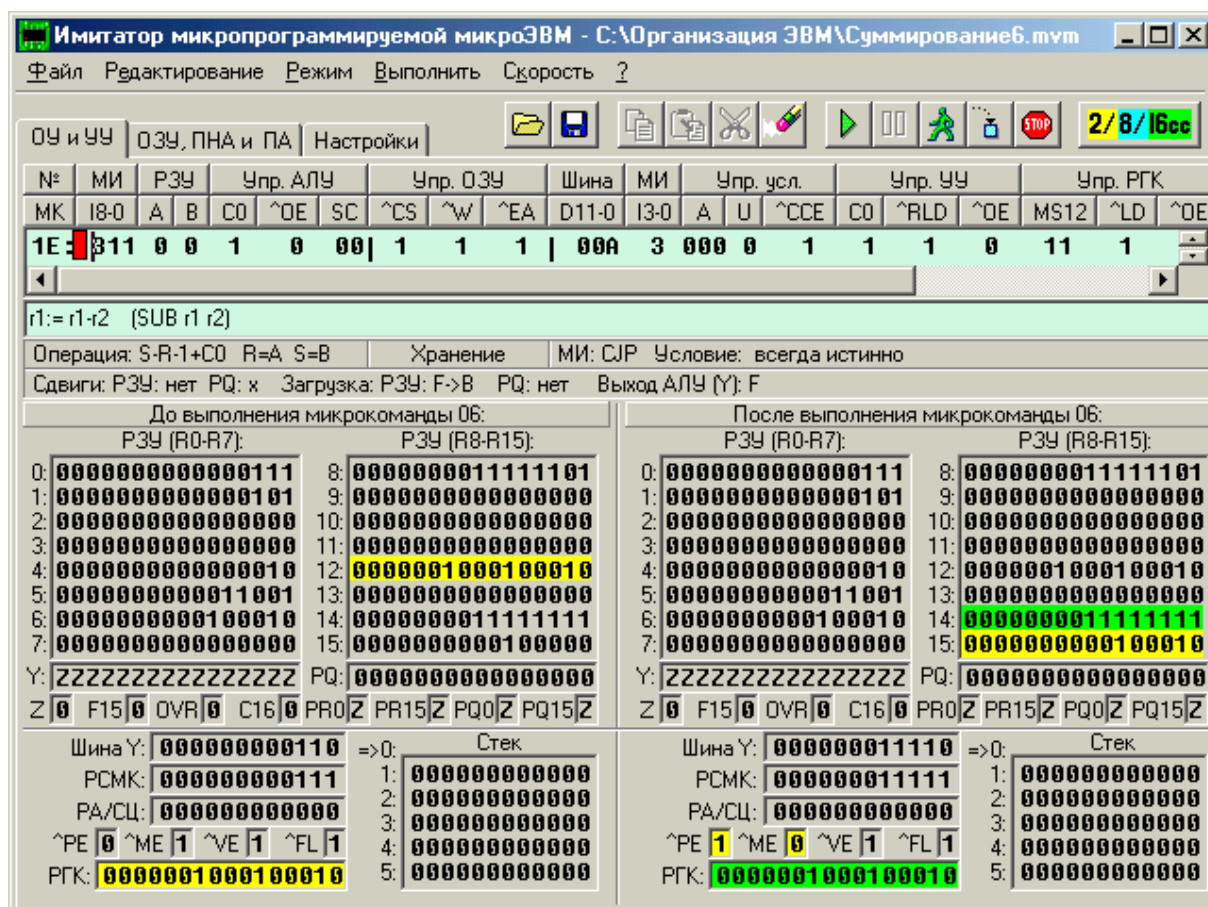


Рис. 6.21. Окно выполнения микрокоманд

Ввод и редактирование данных в ЗУ, а также запись в преобразователь начального адреса начальных адресов микропрограмм операций осуществляется с помощью вкладки «ОЗУ, ПНА и ПА». Настройка схемы выбора адреса производится во вкладке «Настройки» путем перемещения движков с помощью указателя мыши (рис. 6.22). Заданный пользователем вариант настройки схемы выбора адреса сохраняется вместе с файлом, содержащим микропрограмму командного цикла и программу ЭВМ. При необходимости, используя кнопку «По умолчанию» на вкладке «Настройки», можно восстановить принятый в программе вариант коммутации мультиплексоров адреса. Отладку программы удобнее выполнять, используя вкладку «ОЗУ, ПНА и ПА», в которой специальным маркером отмечается выполняемая команда и отображается состояние регистров РЗУ (рис. 6.23).

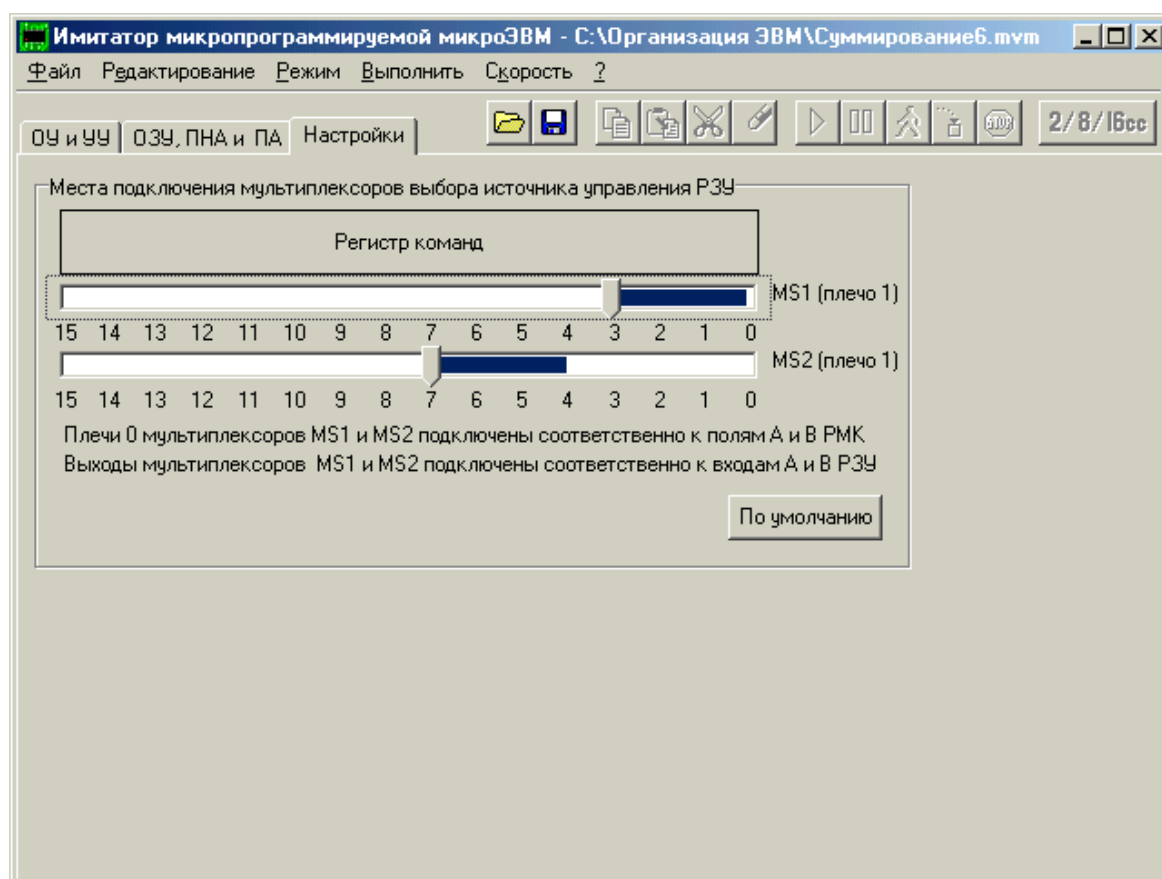


Рис. 6.22. Настройка схемы выбора адреса

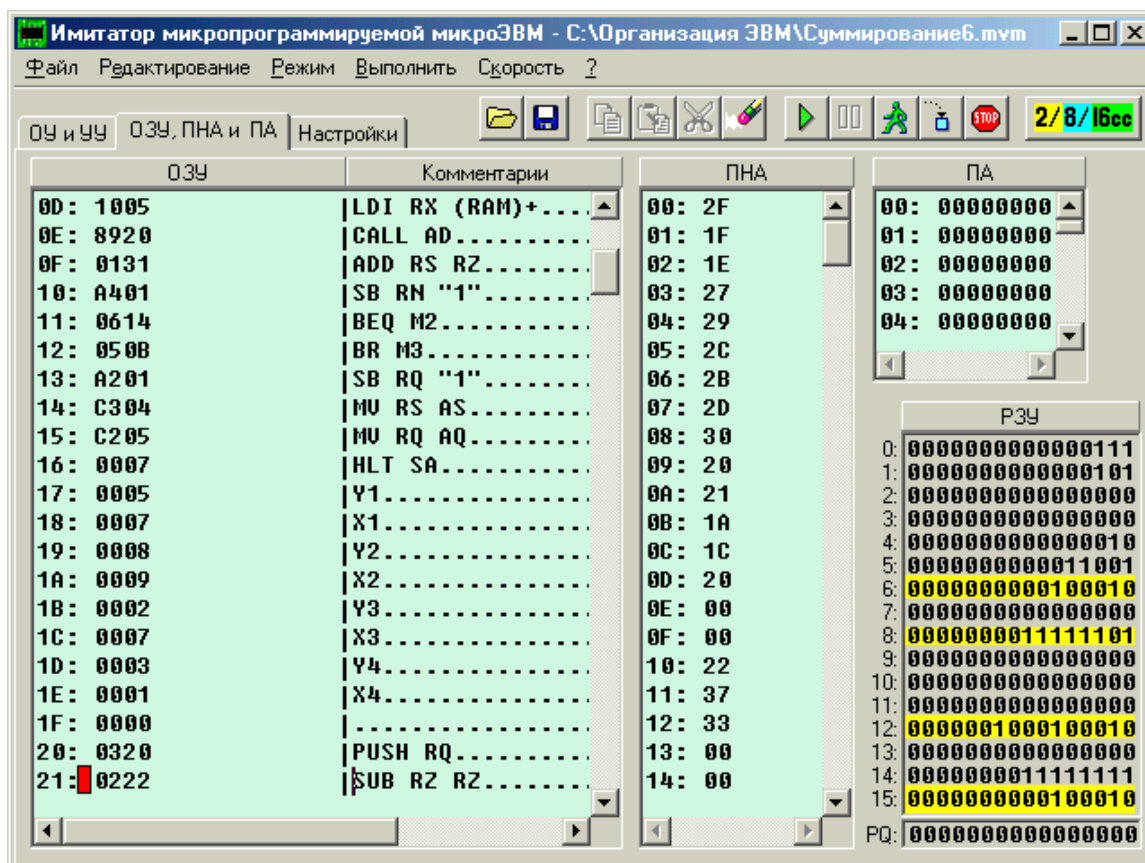


Рис. 6.23. Форма для отладки программы

Кроме того, для наблюдения за выполняемыми командами, может быть открыто дополнительное окно обращений к ОЗУ (рис. 6.24), информация в котором автоматически обновляется после выполнения каждой команды.

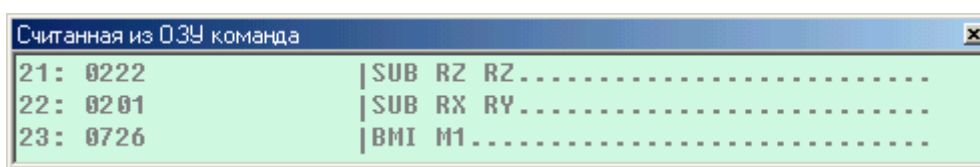


Рис. 6.24. Окно обращений к ОЗУ

Особенности управления программой «Имитатор микропрограммируемой микроЭВМ» проявляются в составе групп команд «Режим» и «Выполнить» управляющего меню программы (рис. 6.25).

| <b><u>Режим</u></b>  |                 |
|--|-----------------|
| <b>Ввода микрокоманд</b>   | <b>Alt+1</b>    |
| <b>Выполнения микрокоманд</b>  | <b>Alt+2</b>    |
| <b>Выполнения команд</b>   | <b>Alt+3</b>    |
| <b>Настройки параметров имитатора</b>  | <b>Alt+4</b>    |
| <b>Переключить</b>   | <b>Ctrl+Tab</b> |
| <b>Конвейер микрокоманд<br/>Аппаратура прерываний<br/>Окно обращений к ОЗУ</b> |                 |

а)





| <b><u>Выполнить</u></b>               |                |
|---------------------------------------|----------------|
| <b>Команду на уровне МК</b>           | <b>F8</b>      |
| <b>Команду</b>                        | <b>Ctrl+F9</b> |
| <b>Программу</b>                      | <b>Ctrl+F2</b> |
| <b>Точка останова</b>                 | <b>Ctrl+F8</b> |
| <b>Сброс</b>                          | <b>Ctrl+R</b>  |
| <b>Переключение системы счисления</b> |                |

б)

Рис. 6.25. Группы команд управляющего меню программы «Имитатор микропрограммируемой микроЭВМ»: «Режим» (а), «Выполнить» (б)

В режиме выполнения команд происходит автоматическое выполнение разработанной микропрограммы командного цикла, обеспечивающее выполнение одной (текущей) команды программы. При этом во вкладке «ОЗУ, ПНА и ПА» отображается новое состояние регистров РЗУ. Перевод программы «Имитатора микропрограммируемой микроЭВМ» в режим выполнения команд меняет назначение отдельных кнопок панели инструментов (табл. 6.9).

Назначение кнопок панели инструментов в режиме выполнения команд

|   |   |
|---|---|
|  | – Выполнить программу                     |
|  | – Остановить выполнение программы         |
|  | – Выполнить команду                       |
|  | – Выполнить команду на уровне микрокоманд |

Команда «Аппаратура прерываний» из группы «Режим» рассматривается и используется в следующей лабораторной работе.

## 6.4. Контрольные вопросы

### *Основные положения*

1. Вид адресации, при которой в команде отсутствует указатель на источник операнда и/или приемник результата называется:

- неявной;
- непосредственной;
- относительной;
- индексной.

2. Использование в команде чтения регистровой косвенной автодекрементной адресации означает выборку операнда из:

- ячейки памяти, адрес которой на единицу меньше адреса, указанного в регистре, номер которого задан в команде;
- регистра, номер которого задан в команде;
- ячейки памяти, адрес которой указан в регистре, номер которого задан в команде;
- ячейки памяти, адрес которой задан в команде.



3. Команда обращения к подпрограмме CALL (rSP) – A (здесь CALL – код операции, rSP – регистр указателя стека, (r) – содержимое регистра, (r)– – уменьшение содержимого регистра на единицу, A – адрес подпрограммы) для сохранения адреса возврата в стеке использует следующие виды адресации:

- регистровая неявная – регистровая косвенная автодекрементная;
- регистровая неявная – регистровая косвенная автоинкрементная;
- регистровая косвенная автодекрементная – базовая;
- прямая – косвенная регистровая.

4. После выполнения двухадресной команды суммирования целых двоичных чисел с записью результата в аккумулятор, в первом адресном поле которой используется непосредственная адресация и указан код 0111, а во втором – косвенная регистровая адресация и указан код 1000, при условии, что в регистре с номером 8 находится код 1001, а в ячейке памяти с адресом 8 находится код 0010, а с адресом 9 – 0011 в аккумулятор будет записан двоичный код \_\_\_\_\_.

5. После выполнения одноадресной команды конъюнктивного умножения содержимого аккумулятора, где первоначально записан двоичный код 1001, и операнда, определяемого адресной частью команды, в которой используется косвенная регистровая автодекрементная адресация и указан код 1000, при условии, что в регистре с номером 8 находится код 1001, в ячейке памяти с адресом 8 находится код 1011, с адресом 9–1010 в аккумулятор будет записан двоичный код \_\_\_\_\_.

#### *Исследуемое устройство*

6. Каким образом программно производится установка в нулевое состояние регистров процессора?

7. Число двухадресных команд в программе, приведенной на рис. 6.16, равно \_\_\_\_ .

8. Какие способы адресации используются в команде POP r (rSP) +?

9. Какие способы адресации используются в команде PUSH r (rSP) – ?

10. Какие способы адресации используются в команде SB r C?

11. Какие команды устанавливают значение признака нуля PZ?

12. Какие команды сохраняют значение признака знака PS?

## 7. СИСТЕМА ПРЕРЫВАНИЯ ЭВМ

### 7.1. Основные положения

#### 7.1.1. Характеристики систем прерывания

Прерывание программ можно определить, как свойство процессора временно прекращать выполнение текущей программы или выходить из состояния ожидания при возникновении определенных событий внутри или вне ЭВМ и переходить к выполнению программы, специально предназначенной для реакции на данное событие.

События проявляют себя в виде специальных сигналов – запросов на прерывание, в случае аппаратных прерываний, или при выполнении специальных команд, когда инициируются программные прерывания.

*Временные характеристики систем прерывания (СП).* К основным временным характеристикам СП относятся: время реакции системы на прерывание и время обработки запроса на прерывание.

Время реакции системы на прерывание  $t_p$  определяется с момента поступления запроса на прерывание до момента прекращения выполнения текущей программы (рис. 7.1, где ТП – текущая программа, ЗП – запрос на прерывание, ПП – прерывающая программа,  $t_p$  – время реакции системы на прерывание,  $t_{оп}$  – время обработки запроса на прерывание).

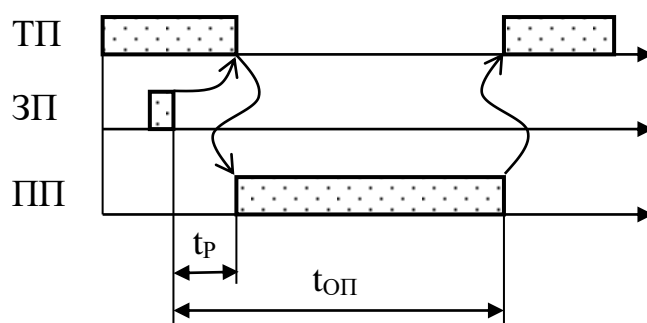


Рис. 7.1 Время реакции системы на прерывание ( $t_p$ ) и время обработки запроса на прерывание ( $t_{оп}$ )

Время обработки запроса на прерывание определяется с момента поступления запроса на прерывание до момента завершения соответствующей прерывающей программы (рис. 7.1).

*Глубина прерываний.* Глубина прерываний – это максимально допустимое число программ, которые последовательно могут прервать друг друга (рис. 7.2, где П0 – текущая программа, П1, П2, П3 – прерывающие программы).

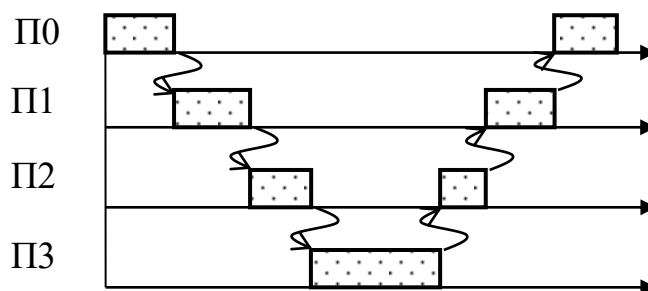


Рис. 7.2. Глубина прерывания

*Насыщение системы прерывания.* Система прерываний входит в состояние насыщения, когда до завершения выполнения прерывающей программы поступает запрос на прерывание по той же самой причине (рис. 7.3, где ТП – текущая программа, ЗП – запрос на прерывание, ПП – прерывающая программа).

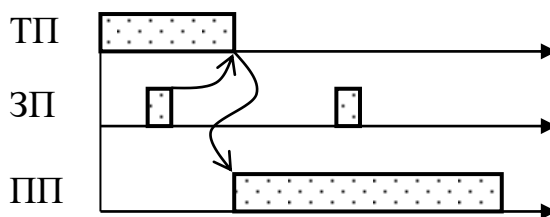


Рис. 7.3. Насыщение системы прерывания

*Допустимый момент прерывания.* В зависимости от допустимого момента прерывания можно выделить следующие основные виды прерываний:

- прерывание допускается после завершения выполнения текущей команды;
- прерывание разрешается после завершения выполнения команд определенных типов;

- прерывание разрешается после завершения выполнения любой микрокоманды;
- прерывание разрешается после завершения микрокоманд определенных типов.

Время реакции системы на прерывание минимально, если прерывания разрешены после выполнения любой микрокоманды. Однако объем сохраняемой информации в этом случае значительно возрастает по сравнению с прерываниями, разрешаемыми после завершения выполнения команды.

### ***7.1.2. Основные фазы процесса прерывания***

Процесс прерывания программы можно разделить на следующие основные фазы.

1. Фиксация запросов на прерывание в регистре запросов на прерывание. Определение наличия не замаскированных запросов. При их наличии выделение запроса с наивысшим приоритетом и формирование адреса соответствующей прерывающей программы. В некоторых системах прерывания принимаемый к обработке запрос на прерывание запоминается в специальном регистре, соответствующий ему разряд в регистре запросов на прерывание сбрасывается уже на первой фазе.

2. Прекращение выполнения текущей программы и сохранение состояния процессора, соответствующего данной программе в момент ее прекращения.

3. Переключение процессора на выполнение прерывающей программы.

4. Выполнение прерывающей программы.

5. Возврат из прерывающей программы и восстановление состояния процессора, соответствующего прерванной программе. В некоторых системах прерывания разряд в регистре запросов на прерывание сбрасывается после завершения выполнения обслуживающей его прерывающей программы.

## 7.2. Аппаратные средства системы прерывания

Система прерывания программ исследуется на примере учебной ЭВМ, разрабатываемой на основе ЭВМ, реализованной при выполнении предыдущей лабораторной работы. При этом в состав ЭВМ включается аппаратура прерывания (АП). Структура аппаратных средств, используемых при выполнении лабораторной работы, приведена на рис. 7.4.

Аппаратура прерывания может принимать восемь запросов на прерывание программ:  $Z_7, Z_8, \dots, Z_0$ . Запросы фиксируются в регистре запросов  $RZ$ . Причем запрос с меньшим номером имеет более высокий приоритет. Запросы на прерывание могут быть замаскированы с помощью 8-разрядной маски  $M$ , записываемой в регистр маски  $RM$ . Если в разряде маски записана единица, то соответствующий запрос при обработке прерываний игнорируется. Аппаратура прерывания формирует сигнал запроса на прерывание  $IN$  и 3-разрядный вектор прерывания  $V$ . Вектор прерывания фиксируется в регистре  $RV$  и соответствует номеру запроса на прерывание, имеющего наивысший приоритет. АП имеет следующие внешние управляющие сигналы:

$CZ$  – очистка в регистре запросов разряда, соответствующего принятому к обработке запросу на прерывание;

$LM$  – загрузка маски с шины данных/адреса в регистр маски;

$OEM$  – выдача маски из регистра маски на шину данных/адреса;

$OEV$  – выдача вектора прерываний из регистра прерываний на шину данных/адреса.

Структура АП показана на рис. 7.5. Аппаратура прерывания работает следующим образом. При наличии хотя бы одного незамаскированного запроса на прерывание вырабатывается сигнал  $IN$ , а с помощью схемы приоритетов  $SP$  формируется унитарный двоичный код, содержащий только одну единицу, соответствующую запросу с наивысшим приоритетом. Унитарный двоичный код преобразуется шифратором  $DC$  в номер запроса, имеющего наивысший приоритет. Этот номер фиксируется в регистре вектора запроса на прерывание  $RV$ . Содержимое регистра  $RV$  может быть выдано на шину данных/адреса через буфер данных  $BD$ , а также преобразовано в унитарный двоичный код с помощью дешифратора  $DC$  для очистки разряда регистра запросов на прерывание при приеме запроса к обработке.

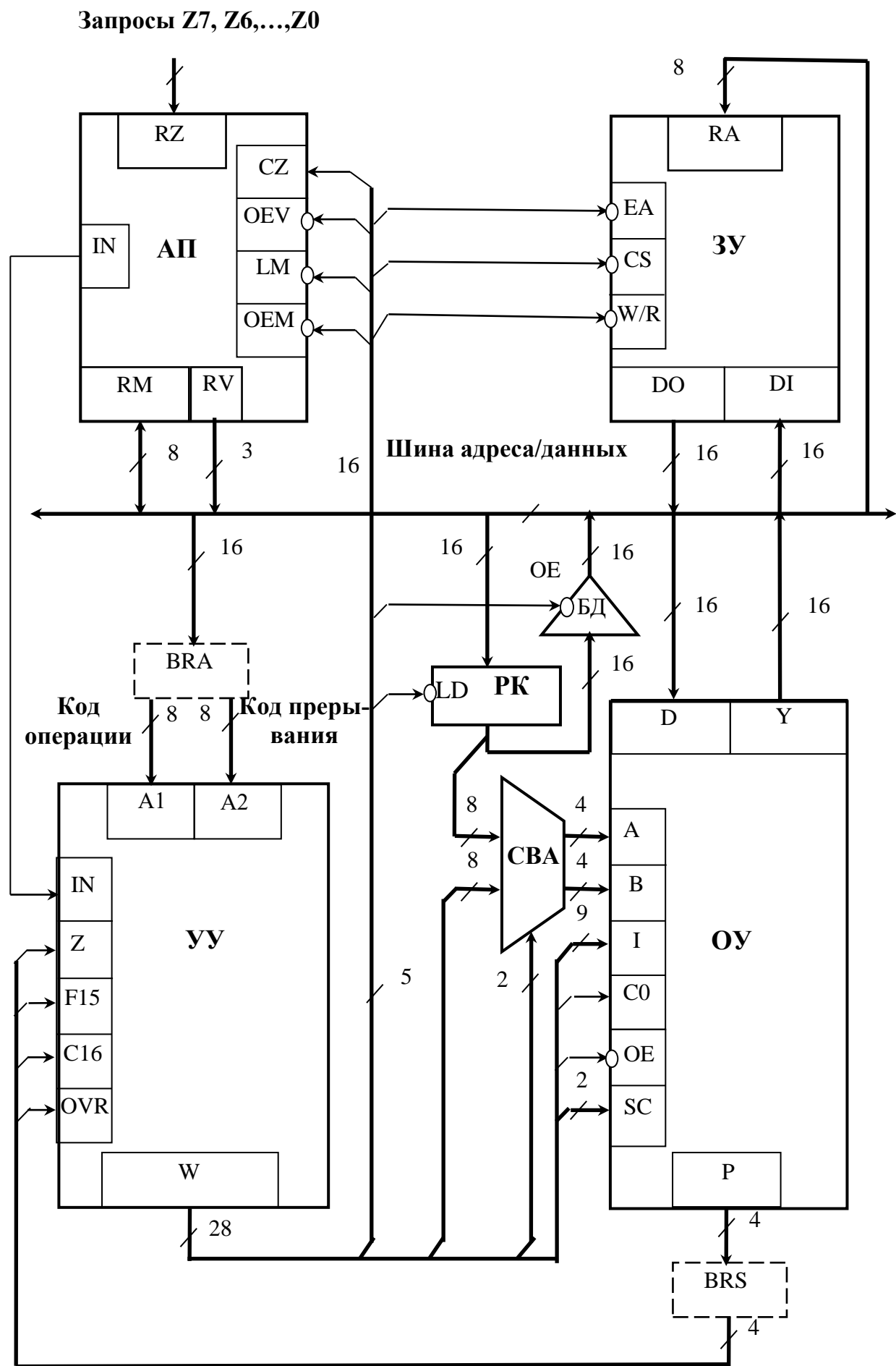


Рис. 7.4. Структурная схема аппаратных средств

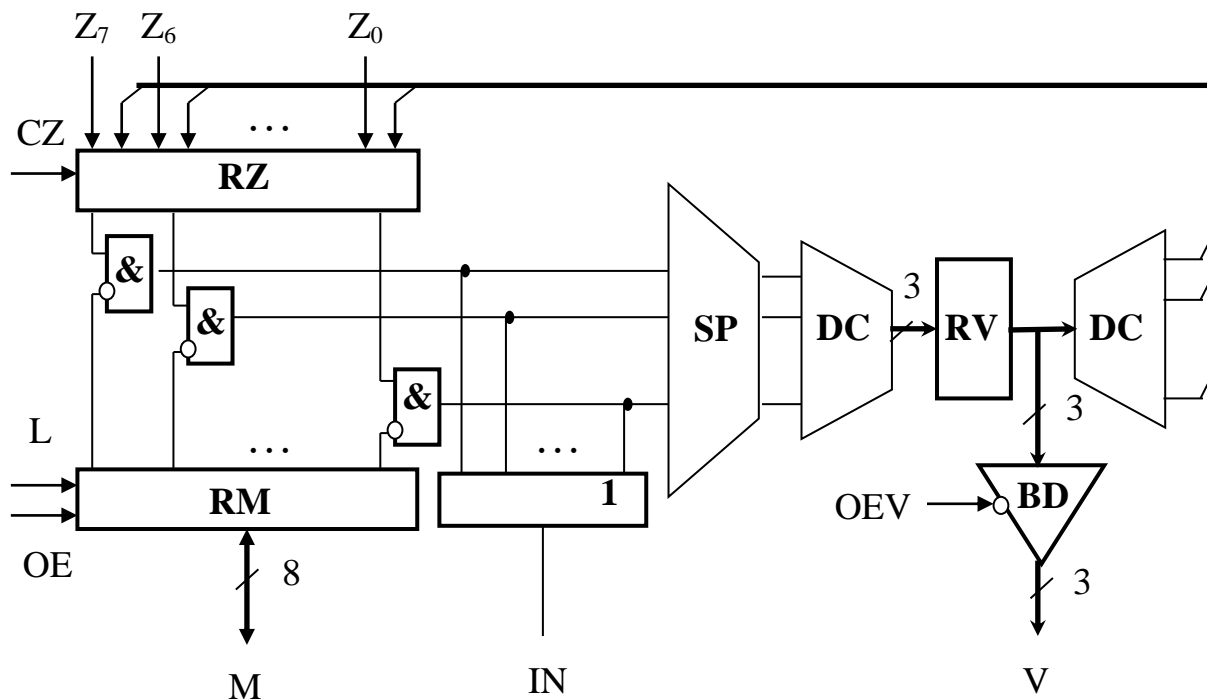


Рис. 7.5. Структура аппаратуры прерывания

### 7.3. Лабораторная работа № 6

Выполнение лабораторной работы складывается из домашней подготовки, экспериментальных исследований на лабораторной установке и оформления отчета.

Для домашней подготовки необходимо получить у преподавателя задание. Домашняя подготовка включает:

- повторение лекционного материала по теме «Системы прерывания программ» и содержания предыдущей лабораторной работы;
- определение архитектуры учебной ЭВМ, разработку программы для решения основной задачи и программ обработки прерываний в соответствии с заданием на лабораторную работу;
- разработку граф-схемы алгоритма командного цикла с подпрограммами перехода на прерывающую программу и возврата к прерванной программе;
- разработку микропрограммы командного цикла с использованием микроинструкций микропроцессора.

Экспериментальные исследования включают: ввод, отладку и выполнение микропрограммы командного цикла по тактам, а основной программы и программ обработки прерываний – по командным циклам. Кроме того, при выполнении лабораторной работы преподаватель задает начальное состояние системы прерывания (поступившие запросы на прерывания, маски программ и т. п.), для которого путем анализа функционирования ЭВМ необходимо получить данные для построения диаграммы, отражающей последовательность переключения ЭВМ с одной программы на другую. При этом обычно предполагается, что ЭВМ начинает работу с основной программы.

Отчет по лабораторной работе должен содержать: титульный лист, задание, граф-схемы решения задач, краткое описание архитектуры ЭВМ и тексты отлаженных программ с распределением памяти программ и данных, граф-схему алгоритма и микропрограмму командного цикла с распределением внутренних регистров и памяти микропрограмм. Кроме того, в отчете необходимо привести диаграмму переключения ЭВМ с одной программы на другую для заданного начального состояния системы прерывания.

Более подробно содержание лабораторной работы рассмотрим на примере задания № 6.

### **7.3.1. Задание № 6**

Определить архитектуру, разработать и отладить микропрограмму командного цикла ЭВМ, составить и выполнить программы для следующих исходных данных:

1. Основная программа производит вычисление суммы  $S$  элементов  $T_i$  массива размерности  $N$ :

$$S = \sum_{i=1}^N T_i ,$$

где  $T \geq 0$  и  $S < 32768$ .

2. В ЭВМ одновременно может поступать до восьми запросов на прерывание программ:  $Z_7, Z_8, \dots Z_0$ . При этом запрос с меньшим номером имеет более



высокий приоритет. Запросы на прерывание могут быть замаскированы с помощью 8-разрядной маски  $M$ , записываемой в регистр маски  $RM$ . Запрос маскируется единичным значением одноименного разряда маски.

3. Каждая программа имеет маску, с помощью которой она может быть защищена от прерываний по тем или иным запросам. Основная программа ( $P$ ) имеет маску, разрешающую все прерывания. Служебная программа ( $PS$ ), выполняющая необходимые начальные установки, характеризуется маской, запрещающей все прерывания.

4. Программа  $P_0$  обработки прерывания нулевого уровня, вызываемая по запросу  $Z_0$ , вычисляет частное от деления  $X$  на  $Y$  нацело ( $X, Y > 0$ , целые числа). Остальные программы обработки прерываний ( $P_1$ - $P_2$ ) являются «пустыми» и после установки маски возвращают управление.

5. При запуске ЭВМ в начале выполняется служебная программа, которая затем передает управление основной программе. Необходимо построить диаграмму прерываний, исходя из анализа работы ЭВМ для заданного начального состояния системы прерывания. Начальное состояние системы прерывания характеризуется набором масок программ и множеством запросов, поступивших в систему (табл. 7.1).

Таблица 7.1

Начальное состояние системы прерывания

| Объект                | Условное обозначение | Значение |
|-----------------------|----------------------|----------|
| Маска программы $P$   | $M$                  | 00000000 |
| Маска программы $P_0$ | $M_0$                | 00000001 |
| Маска программы $P_1$ | $M_1$                | 01010111 |
| Маска программы $P_2$ | $M_2$                | 01001101 |
| Маска программы $P_3$ | $M_3$                | 10111100 |
| Маска программы $P_4$ | $M_4$                | 00110110 |
| Маска программы $P_5$ | $M_5$                | 01110011 |
| Маска программы $P_6$ | $M_6$                | 01110010 |
| Маска программы $P_7$ | $M_7$                | 11100110 |
| Маска программы $PS$  | $MS$                 | 11111111 |
| Множество запросов    | $Z$                  | 11110111 |

### 7.3.2. Определение архитектуры и программирование

Архитектура ЭВМ может быть получена путем развития архитектуры, разработанной при выполнении предыдущей лабораторной работы. Дополнительные команды определяются в процессе разработки алгоритмов основной и прерывающих программ и анализа функций системы прерывания ЭВМ с учетом требований, приведенных в задании. Граф-схема алгоритма основной программы изображена на рис. 7.6, а программы обработки прерывания P0 – на рис. 7.7, где SAVE – подпрограмма сохранения, RESTORE – подпрограмма восстановления содержимого регистров ЭВМ.

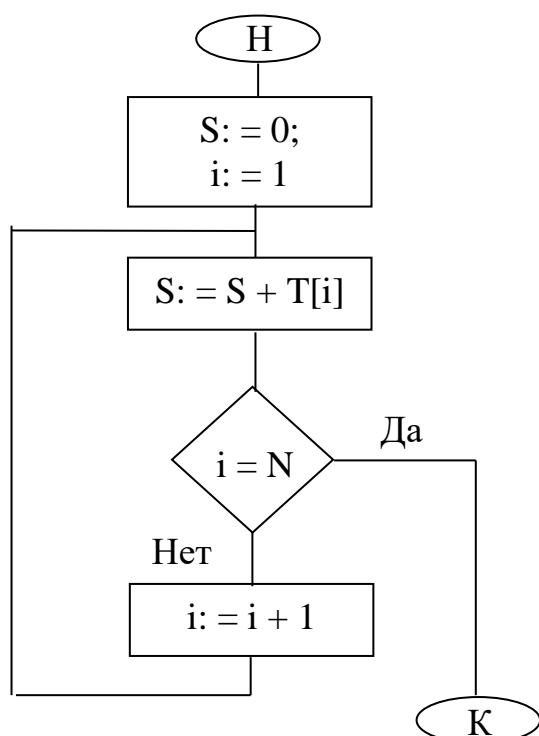


Рис. 7.6. Граф-схема алгоритма основной программы

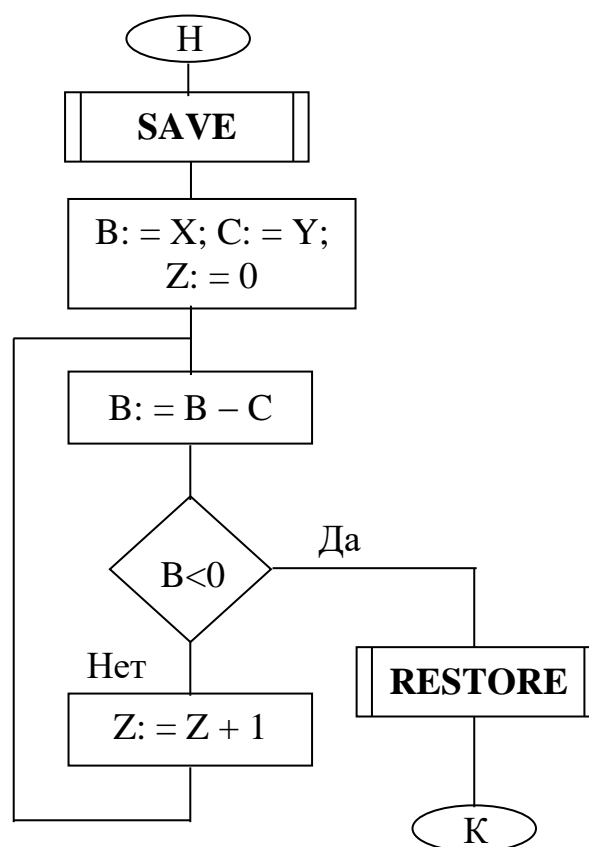


Рис. 7.7. Граф-схема алгоритма прерывающей программы P0

Разработка архитектуры ЭВМ при выполнении лабораторной работы включает: определение форматов данных, выбор состава программно-доступных регистров и определение системы команд. Рассмотрим решение выделенных задач на примере задания № 6.

*Форматы данных.* Данные в примере являются целыми числами, изменяющимися в пределах от 0 до 32767. В этом случае любое число можно представить 16-разрядным двоичным кодом, старший разряд которого определяет знак числа.

*Программно-доступные регистры.* ЭВМ имеет 10 16-разрядных программно-доступных регистров: шесть регистров общего назначения (r0–r5), программный счетчик – PC (r6), регистр признаков – RP (r7), содержащий разряды двух признаков: нуля (PZ) и знака (PS), регистр указателя стека – rSP (r8), регистр адреса таблицы прерываний – rATI, а также 8-разрядный регистр маски RM. Распределение программно-доступных регистров ЭВМ показано на рис. 7.8.

| Регистры ЭВМ |                    |                                   |
|--------------|--------------------|-----------------------------------|
| r0:          | X                  | Делимое X                         |
| r1:          | Y (T)              | Делитель Y (суммируемые числа T)  |
| r2:          | Z                  | Частное Z                         |
| r3:          | S                  | Сумма S                           |
| r4:          | N                  | Число повторений цикла N          |
| r5:          | AM                 | Адрес массива AM                  |
| r6:          | PC                 | Программный счетчик               |
| r7:          | PS      RP      PZ | Регистр признаков                 |
| r8:          | rSP                | Регистр указателя стека           |
| r9:          | rATI               | Регистр адреса таблицы прерываний |
| RM:          | M                  | Регистр маски программы           |

Рис. 7.8. Распределение регистров ЭВМ

*Система команд.* Разработка системы команд предполагает определение и набора операций, способов адресаций, модификаций и форматов команд.

В рассматриваемом примере в систему команд ЭВМ, разработанную при выполнении предыдущей лабораторной работы, дополнительно включаются следующие две команды.

LM A – загрузка маски. Данная команда обеспечивает загрузку в регистр маски младшего байта ячейки памяти, адрес A которой указан в команде.

IRET – возврат из прерывающей программы. При выполнении команды из стека восстанавливаются значения регистра маски и программного счетчика, соответствующие последней прерванной программе, что обеспечивает возврат на команду, перед которой произошло прерывание. При этом предполагается, что сохранение содержимого программного счетчика и регистра маски в стеке осуществляется с помощью специальной подмикропрограммы командного цикла ЭВМ при переключении процессора на прерывающую программу.

Система команд ЭВМ приведена в табл. 7.2.

Таблица 7.2

Система команд

| Наименование                      | Мнемоника      | Описание   | Признаки |    |
|-----------------------------------|----------------|--|----------|----|
|                                   |                |  | PZ       | PS |
| СУММИРОВАНИЕ                      | ADD r r*       | $r := r + r^*, PC := PC + 1$                     | +        | +  |
| ВЫЧИТАНИЕ                         | SUB r r*       | $r := r - r^*, PC := PC + 1$                     | +        | +  |
| ДОБАВЛЕНИЕ C                      | AD r C         | $r := r + C, PC := PC + 1$                       | +        | +  |
| ВЫЧИТАНИЕ C                       | SB r C         | $r := r - C, PC := PC + 1$                       | +        | +  |
| ЧТЕНИЕ В РЕГИСТР                  | LD r A         | $r := M[A], PC := PC + 1$                        | –        | –  |
| ЗАПИСЬ РЕГИСТРА                   | MV r A         | $M[A] := r, PC := PC + 1$                        | –        | –  |
| ЧТЕНИЕ В РЕГИСТР<br>с индексацией | LDI r (r*) +   | $r := M[r^*], r^* := r^* + 1;$<br>$PC := PC + 1$ | +        | +  |
| ЗАПИСЬ В СТЕК                     | PUSH r (rSP) – | $M[rSP] := r;$<br>$rSP := rSP - 1, PC := PC + 1$ | –        | –  |
| ЧТЕНИЕ ИЗ СТЕКА                   | POP r (rSP) +  | $rSP := rSP + 1;$<br>$r := M[rSP], PC := PC + 1$ | –        | –  |
| ПЕРЕХОД                           | BR A           | $PC := A$  | –        | –  |

| Наименование                            | Мнемоника     | Описание  | Признаки |    |
|---|---------------|---|----------|----|
|   |               |   | PZ       | PS |
| ПЕРЕХОД,<br>ЕСЛИ НУЛЬ                   | BEQ A         | Если PZ = 1, то PC: = A,<br>иначе PC: = PC + 1            | –        | –  |
| ПЕРЕХОД,<br>ЕСЛИ МИНУС                  | BMI A         | Если PS = 1, то PC: = A,<br>иначе PC: = PC + 1            | –        | –  |
| ОБРАЩЕНИЕ<br>К ПОДПРОГРАММЕ             | CALL (rSP)- A | M[rSP]: = PC;<br>rSP: = rSP – 1, PC: = A                  | –        | –  |
| ОСТАНОВ                                 | HLT A         | PC: = A, останов  | –        | –  |
| ЗАГРУЗКА МАСКИ                          | LM A          | RM: = M[A], PC: = PC + 1                                  | –        | –  |
| ВОЗВРАТ<br>ИЗ ПЕРЕРЫВАЮЩЕЙ<br>ПРОГРАММЫ | IRET          | SP: = SP + 1, RM: = M[rSP];<br>SP: = SP + 1, PC: = M[rSP] | –        | –  |

В таблице использованы следующие обозначения:  $r, r^* \in \{r0, r1, \dots, r8\}$  – программно-доступные регистры: регистр  $r^*$  является источником данных, а регистр  $r$  – приемником результата, но может также служить источником второго операнда; RM – регистр маски, M[A] – ячейка памяти с адресом A; знак «+» в описании признаков означает, что устанавливается новое значение признака по результату выполнения команды, а знак «–» свидетельствует о сохранении старого значения признака.

*Программирование.* Программирование ЭВМ с аппаратурой прерывания включает разработку основной и прерывающих программ и распределение программ и данных с учетом выбранного способа переключения ЭВМ на прерывающие программы. Кроме того, для выполнения начальных установок вводится специальная служебная программа. Служебная программа (PS) выполняет следующие действия (рис. 7.9): загружает маску, запрещающую все прерывания, инициализирует регистр указателя стека и регистр адреса таблицы прерываний, а затем передает управление основной программе.

Основная программа (рис. 7.10) и программа прерывания P0 (рис. 7.11) составляются в соответствии с алгоритмами, изображенными на рис. 7.6, рис. 7.7 с использованием системы команд ЭВМ (табл. 7.2).

|             |  |
|-------------|--|
| LM AMS      | Загрузка маски служебной программы     |
| LD rSP ASP  | Загрузка указателя стека SP            |
| LD rATI ATI | Загрузка адреса таблицы прерываний ATI |
| LD PC AP    | Загрузка начального адреса программы P |

Рис. 7.9. Служебная программа начальных установок

|    |               |  |
|----|---------------|--|
|    | LM AMP        | Загрузка маски программы P                     |
|    | LD r5 AAM     | Загрузка адреса массива AM в регистр r5        |
|    | LD r4 AN      | Загрузка числа повторений цикла N в регистр r4 |
|    | SUB r3 r3     | Очистка регистра r3 для суммы S                |
| m2 | LDI r1 (r5) + | Чтение числа T[i] в регистр r1                 |
|    | ADD r3 r1     | Суммирование                                   |
|    | SB r4 «1»     | Вычитание единицы из числа повторений цикла    |
|    | BEQ m1        | Если PZ = 1 (r4 = 0), то переход на метку m1   |
|    | BR m2         | Переход на метку m2                            |
| m1 | MV r3 AS      | Запись в память суммы S по адресу AS           |
|    | HLT SA        | Загрузка PC и останов                          |

Рис. 7.10. Основная программа

|    |           |  |
|----|-----------|--|
|    | LM AM0    | Загрузка маски программы P0                |
|    | PUSH RP   | Сохранение в стеке содержимого регистра RP |
|    | PUSH r0   | Сохранение в стеке содержимого регистра r0 |
|    | PUSH r1   | Сохранение в стеке содержимого регистра r1 |
|    | PUSH r2   | Сохранение в стеке содержимого регистра r2 |
|    | SUB r2 r2 | Очистка регистра для частного Z            |
|    | LD r0 AX  | Загрузка X в регистр r0                    |
|    | LD r1 AY  | Загрузка Y в регистр r1                    |
| m2 | SUB r0 r1 | Вычитание из делимого X делителя Y         |
|    | BMI m1    | Если PS = 1, то переход на метку m1        |
|    | AD r2 «1» | Увеличение на единицу частного Z           |
|    | BR m2     | Переход на метку m2                        |
| m1 | MV r2 AZ  | Запись частного Z                          |
|    | POP r2    | Чтение из стека содержимого регистра r2    |
|    | POP r1    | Чтение из стека содержимого регистра r1    |
|    | POP r0    | Чтение из стека содержимого регистра r0    |
|    | POP RP    | Чтение из стека содержимого регистра RP    |
|    | IRET      | Возврат из программы P0                    |

Рис. 7.11. Прерывающая программа P0

Программы обработки прерываний P1, P2, ..., P7 являются по заданию «пустыми» и состоят из двух команд: загрузки маски и возврата из прерывания. Пример такой программы приведен на рис. 7.12.

|  |          |                               |
|--|----------|-------------------------------|
|  | LM AM[K] | Загрузка маски программы P[K] |
|  | IERT     | Возврат из программы P[K]     |

Рис. 7.12. Прерывающая программа P[K] (K = 1, 2, 3, ..., 7)

### 7.3.3. Кодирование программы и распределение памяти программ и данных

Дополнительные команды загрузки маски LM A и возврата из прерывающей программы IRET имеют такой же формат, как и команды переходов (см. раздел 7.2.3), при этом адресная часть в команде возврата из прерывающей программы игнорируется. Наименования, мнемонические обозначения и коды операций приведены в табл. 7.3.

Таблица 7.3

Коды операций

| Наименование                     | Мнемоника | Код операции |
|----------------------------------|-----------|--------------|
| СУММИРОВАНИЕ                     | ADD       | 01           |
| ВЫЧИТАНИЕ                        | SUB       | 02           |
| ДОБАВЛЕНИЕ КОНСТАНТЫ             | AD        | 9            |
| ВЫЧИТАНИЕ КОНСТАНТЫ              | SB        | A            |
| ЧТЕНИЕ В РЕГИСТР                 | LD        | B            |
| ЗАПИСЬ РЕГИСТРА                  | MV        | C            |
| ЧТЕНИЕ В РЕГИСТР с индексацией   | LDI       | 10           |
| ЗАПИСЬ В СТЕК                    | PUSH      | 03           |
| ЧТЕНИЕ ИЗ СТЕКА                  | POP       | 04           |
| ПЕРЕХОД                          | BR        | 05           |
| ПЕРЕХОД, ЕСЛИ НУЛЬ               | BEQ       | 06           |
| ПЕРЕХОД, ЕСЛИ МИНУС              | BMI       | 07           |
| ОБРАЩЕНИЕ К ПОДПРОГРАММЕ         | CALL      | 8            |
| ОСТАНОВ                          | HLT       | 00           |
| ЗАГРУЗКА МАСКИ                   | LM        | 11           |
| ВОЗВРАТ ИЗ ПРЕРЫВАЮЩЕЙ ПРОГРАММЫ | IRET      | 12           |

Распределение памяти программ и данных приведено в табл. 7.4 (служебная программа, таблица прерываний и маски программ), табл. 7.5 (основная программа и прерывающая программа P0) и табл. 7.6 (прерывающие программы P1–P7 и стек).



Таблица 7.4

**Распределение памяти**  
(служебная программа, таблица прерываний и маски программ)

| Адрес               | Код  | Мнемоника   | Комментарий                            |
|---------------------|------|-------------|--|
| Служебная программа |      |             |  |
| 00                  | 0005 | SA          | Начальный адрес служебной программы    |
| 01                  | 00FF | MS          | Маска служебной программы              |
| 02                  | FFFF | ASP         | Начальный адрес области памяти стека   |
| 03                  | 000A | ATI         | Начальный адрес таблицы прерываний     |
| 04                  | 0013 | AP          | Начальный адрес основной программы P   |
| 05                  | 1101 | LM AMS      | Загрузка маски служебной программы     |
| 06                  | B802 | LD rSP ASP  | Загрузка указателя стека SP            |
| 07                  | B903 | LD rATI ATI | Загрузка адреса таблицы прерываний     |
| 08                  | B604 | LD PC AP    | Загрузка начального адреса программы P |
| 09                  |      |             | Свободная ячейка памяти                |
| Таблица прерываний  |      |             |  |
| 0A                  | 0043 | AP0         | Адрес прерывающей программы P0         |
| 0B                  | 0060 | AP1         | Адрес прерывающей программы P1         |
| 0C                  | 0062 | AP2         | Адрес прерывающей программы P2         |
| 0D                  | 0064 | AP3         | Адрес прерывающей программы P3         |
| 0E                  | 0066 | AP4         | Адрес прерывающей программы P4         |
| 0F                  | 0068 | AP5         | Адрес прерывающей программы P5         |
| 10                  | 006A | AP6         | Адрес прерывающей программы P6         |
| 11                  | 006C | AP7         | Адрес прерывающей программы P7         |
| Маски программ      |      |             |  |
| 12                  | 0000 | M           | Маска основной программы P             |
| 13                  | 00FF | M0          | Маска прерывающей программы P0         |
| 14                  | 0000 | M1          | Маска прерывающей программы P1         |
| 15                  | 0000 | M2          | Маска прерывающей программы P2         |
| 16                  | 0000 | M3          | Маска прерывающей программы P3         |
| 17                  | 0000 | M4          | Маска прерывающей программы P4         |
| 18                  | 0000 | M5          | Маска прерывающей программы P5         |
| 19                  | 0000 | M6          | Маска прерывающей программы P6         |
| 1A                  | 0000 | M7          | Маска прерывающей программы P7         |

Таблица 7.5

## Распределение памяти (основная программа и прерывающая программа P0)

| Адрес                    | Код  | Мнемоника    | Комментарий                                 |
|--------------------------|------|--------------|---|
| Основная программа       |      |              |   |
| 20                       | 002E | AM           | Начальный адрес массива                     |
| 21                       | 0003 | N            | Количество чисел в массиве                  |
| 22                       |      | S            | Результат – сумма чисел                     |
| 23                       | 1112 | LM AMP       | Загрузка маски программы P                  |
| 24                       | B520 | LD r5 AAM    | Загрузка адреса массива AM в регистр r5     |
| 25                       | B421 | LD r4 AN     | Загрузка числа повторений цикла N           |
| 26                       | 0233 | SUB r3 r3    | Очистка регистра r3 для суммы S             |
| 27                       | 1015 | LDI r1 (r5)+ | Чтение числа T[i] в регистр r1              |
| 28                       | 0131 | ADD r3 r1    | Суммирование                                |
| 29                       | A401 | SB r4 «1»    | Вычитание единицы из числа N                |
| 2A                       | 062C | BEQ m1       | Если PZ = 1 (N = 0), то переход на метку m1 |
| 2B                       | 0527 | BR m2        | Переход на метку m2                         |
| 2C                       | C322 | MV r3 AS     | Запись суммы S адресу AS                    |
| 2D                       | 0005 | HLT SA       | Загрузка PC и останов                       |
| 2E                       |      | T1           |   |
| 2F                       |      | T2           |   |
| 30                       |      | T3           |   |
| Прерывающая программа P0 |      |              |   |
| 40                       |      | X            | Делимое X                                   |
| 41                       |      | Y            | Делитель Y                                  |
| 42                       |      | Z            | Частное Z                                   |
| 43                       | 1113 | LM AM0       | Загрузка маски программы P0                 |
| 44                       | 0370 | PUSH RP      | Сохранение содержимого регистра RP          |
| 45                       | 0300 | PUSH r0      | Сохранение содержимого регистра r0          |
| 46                       | 0310 | PUSH r1      | Сохранение содержимого регистра r1          |
| 47                       | 0320 | PUSH r2      | Сохранение содержимого регистра r2          |

|       |      |           |   |
|-------|------|-----------|---|
| 48    | 0222 | SUB r2 r2 | Очистка регистра для частного Z         |
| 49    | B040 | LD r0 AX  | Загрузка X в регистр r0                 |
| 4A    | B141 | LD r1 AY  | Загрузка Y в регистр r1                 |
| 4B    | 0201 | SUB r0 r1 | Вычитание из делимого X делителя Y      |
| Адрес | Код  | Мнемоника | Комментарий                             |
| 4C    | 074F | BMI m1    | Если PS = 1, то переход на метку m1     |
| 4D    | 9201 | AD r2 «1» | Увеличение на единицу частного Z        |
| 4E    | 054B | BR m2     | Переход на метку m2                     |
| 4F    | C242 | MV r2 AZ  | Запись частного Z                       |
| 50    | 0420 | POP r2    | Чтение из стека содержимого регистра r2 |
| 51    | 0410 | POP r1    | Чтение из стека содержимого регистра r1 |
| 52    | 0400 | POP r0    | Чтение из стека содержимого регистра r0 |
| 53    | 0470 | POP RP    | Чтение из стека содержимого регистра RP |
| 54    | 1200 | IRET      | Возврат из программы P0                 |

Таблица 7.6

## Распределение памяти (прерывающие программы P1–P7 и стек)

| Адрес                    | Код  | Мнемоника | Комментарий                 |
|--------------------------|------|-----------|-----------------------------|
| Прерывающая программа P1 |      |           |                             |
| 60                       | 1114 | LM AM1    | Загрузка маски программы P1 |
| 61                       | 1200 | IERT      | Возврат из программы P1     |
| Прерывающая программа P2 |      |           |                             |
| 62                       | 1115 | LM AM2    | Загрузка маски программы P2 |
| 63                       | 1200 | IERT      | Возврат из программы P2     |
| Прерывающая программа P3 |      |           |                             |
| 64                       | 1116 | LM AM3    | Загрузка маски программы P3 |
| 65                       | 1200 | IERT      | Возврат из программы P3     |
| Прерывающая программа P4 |      |           |                             |
| 66                       | 1117 | LM AM4    | Загрузка маски программы P4 |
| 67                       | 1200 | IERT      | Возврат из программы P4     |

| Адрес                    | Код  | Мнемоника | Комментарий                 |
|--------------------------|------|-----------|-----------------------------|
| Прерывающая программа P5 |      |           |                             |
| 68                       | 1118 | LM AM5    | Загрузка маски программы P5 |
| 69                       | 1200 | IERT      | Возврат из программы P5     |
| Прерывающая программа P6 |      |           |                             |
| 6A                       | 1119 | LM AM6    | Загрузка маски программы P6 |
| 6B                       | 1200 | IERT      | Возврат из программы P6     |
| Прерывающая программа P7 |      |           |                             |
| 6C                       | 111A | LM AM7    | Загрузка маски программы P7 |
| 6D                       | 1200 | IERT      | Возврат из программы P7     |
| Область памяти стека     |      |           |                             |
| ...                      |      |           |                             |
| FD                       |      |           |                             |
| FE                       |      |           |                             |
| FF                       |      |           |                             |

#### ***7.3.4. Разработка алгоритма работы и микропрограммная реализация ЭВМ***

Основу микропрограммы командного цикла разрабатываемой ЭВМ может составить микропрограмма, составленная при выполнении предыдущей лабораторной работы, в которую добавляются дополнительные подмикропрограммы. Алгоритм работы ЭВМ представлена на рис. 7.13 в виде укрупненной граф-схемы микропрограммы командного цикла, дополнительно содержащей подмикропрограммы загрузки маски LM A и возврата из прерывающей программы IRET, а также подмикропрограмму прерывания INT.

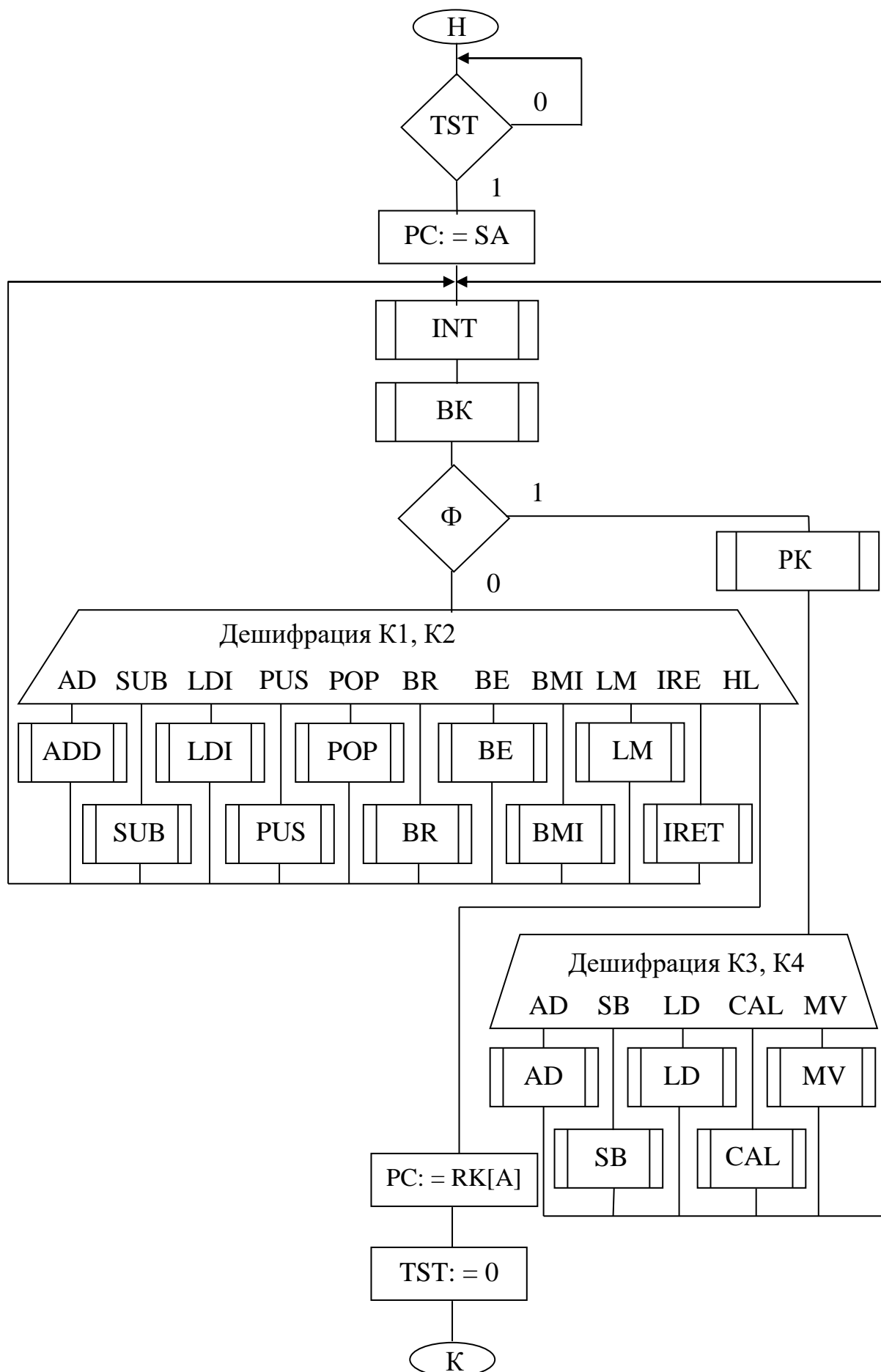


Рис. 7.13. Граф-схема микропрограммы командного цикла ЭВМ

Прерывания в ЭВМ допускаются перед считыванием очередной команды. Подмикропрограмма прерывания INT (рис. 7.14, а) опрашивает выход IN АП и при наличии незамаскированных запросов на прерывания осуществляет переключение ЭВМ на программу обработки прерывания, номер которого сформирован АП.

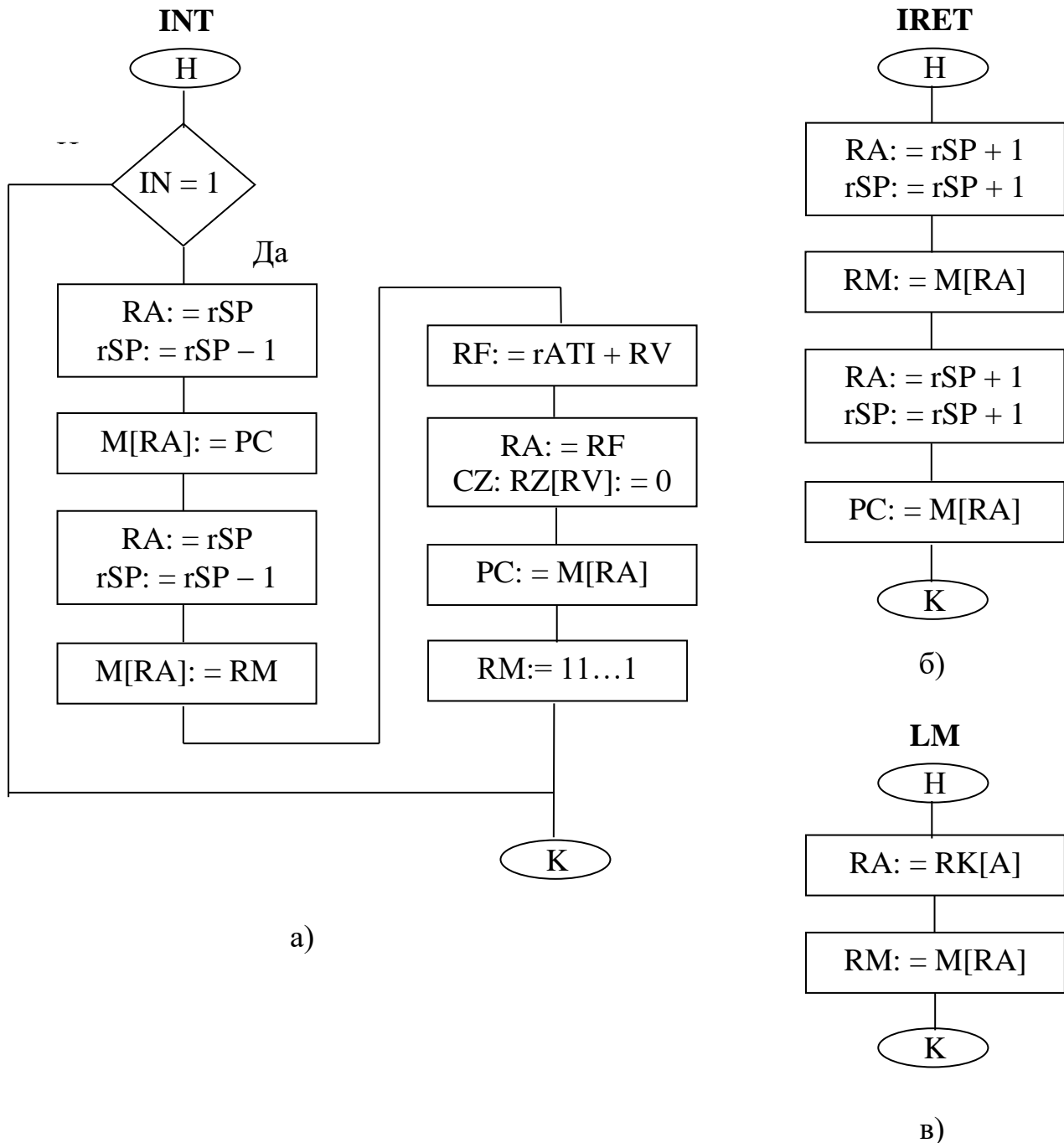


Рис. 7.14. Граф-схемы подмикропрограмм: перехода на прерывающую программу (а), операции возврата из прерывающей программы (б) и операции загрузки маски программы (в)

Для переключения ЭВМ на прерывающую программу подмикропрограмма INT выполняет следующие действия:

- записывает в стек адрес следующей команды прерываемой программы (из PC) и содержимое регистра маски RM (считывая его из AP);
- используя адрес таблицы прерываний и номер запроса на прерывание, загружает в программный счетчик адрес первой команды прерывающей программы;
- выдает сигнал CZ, управляющий сбросом соответствующего обрабатываемому прерыванию разряда регистра запросов RZ;
- заносит в регистр маски код 11111111, что означает запрещение прерываний.

Граф-схемы подмикропрограмм дополнительных операций приведены на рис. 7.14, а и б. При выполнении команды возврата из прерывающей программы IRET из стека восстанавливаются ранее сохраненные значения маски и программного счетчика.

Микропрограммная реализация ЭВМ включает: распределение внутренних регистров микропроцессора, разработку и кодирование подмикропрограмм командного цикла, а также распределение памяти микропрограмм.

*Регистры ЭВМ.* Распределение внутренних регистров ЭВМ показано на рис. 7.15. ЭВМ имеет 10 16-разрядных программно-доступных регистров: шести регистров общего назначения (r0-r5), программный счетчик – PC (r6), регистр признаков – RP (r7), содержащий разряды двух признаков: нуля (PZ) и знака (PS), регистр указателя стека – rSP (r8), регистр адреса таблицы прерываний – rATI, а также 8-разрядный регистр маски RM. Все 16-разрядные программно-доступные регистры отображаются на регистры R0-R9 операционного устройства. Регистр маски RM входит в состав аппаратуры прерывания. Кроме одиннадцати программно-доступных регистров в состав ЭВМ входят восемь программно-недоступных регистров. Четыре таких регистра (два для команды, и по одному для константы, счетчика адреса ЗУ, операнда Y) отображаются на регистры R13–R15 и RQ операционного устройства.

Кроме того, программно-недоступными регистрами являются регистр адреса ЗУ (RA) и регистр команд (RK), регистр запросов на прерывание (RZ) и регистр номера (вектора) запроса на прерывание (RV).

| PЗУ (R0–R7) |                   |       | PЗУ (R8–R15) |                         |  |
|-------------|-------------------|-------|--------------|-------------------------|--|
| 0:          | r0                | X     | 8:           | r8 (rSP)                |  |
| 1:          | r1                | Y (T) | 9:           | r9 (rATI)               |  |
| 2:          | r2                | Z     | 10:          |                         |  |
| 3:          | r3                | S     | 11:          |                         |  |
| 4:          | r4                | N     | 12:          |                         |  |
| 5:          | r5                | AM    | 13:          | Буферный регистр команд |  |
| 6:          | r6 (PC)           |       | 14:          | Регистр константы       |  |
| 7:          | PS   r7 (RP)   PZ |       | 15:          | Счетчик адреса ЗУ RK[A] |  |

|     |          |     |                  |
|-----|----------|-----|------------------|
| RM: | Маска    | RQ: | Регистр операнда |
| RA: | Адрес ЗУ |     |                  |
| RZ: | Запросы  | RK: | Регистр команды  |
| RV: | Вектор   |     |                  |

Рис. 7.15. Распределение регистров ЭВМ

*Дешифрация кода операции.* Код операции преобразуется с помощью преобразователя начального адреса (ПНА), входящего в состав устройства управления ЭВМ, в начальный адрес подмикропрограммы соответствующей операции. Набор операций разрабатываемой ЭВМ является расширением набора операций ЭВМ, полученной в предыдущей лабораторной работе. Соответствие между кодами дополнительных операций и начальными адресами подмикропрограмм дополнительных операций приведены в табл. 7.7.



Таблица 7.7

## Коды операций и начальные адреса подмикропрограмм

| Мнемоника | Код операции | Адрес первой МК |
|-----------|--------------|-----------------|
| LM        | 11           | 37              |
| IRET      | 12           | 33              |

*Кодирование микропрограммы.* Кодирование микропрограммы командного цикла для ЭВМ производится аналогично кодированию микропрограммы, разработанной в предыдущей лабораторной работе (см. раздел 6) и отличается дополнительными микропрограммами (перехода на прерывающую программу, возврата из прерывающей программы и загрузки маски), а также дополнительным полем микрокоманды «Управление АП», содержащим сигналы (CZ, LM, OEM, OEV) для управления АП. В данной лабораторной работе сохранена ранее принятая настройка схемы выбора адреса (рис. 7.16).

| MS1<br>(управляющий<br>вход СВА) | A<br>(адресный вход<br>РЗУ) | MS2<br>(управляющий вход<br>СВА) | B<br>(адресный вход РЗУ) |
|----------------------------------|-----------------------------|----------------------------------|--------------------------|
| 0                                | PMK[A]                      | 0                                | PMK[B]                   |
| 1                                | PK[3:0]                     | 1                                | PK[7:4]                  |

Рис. 7.16. Управление коммутацией адресных входов РЗУ

### 7.3.5. Ввод и отладка микропрограммы командного цикла и программ решения задач

В лабораторной работе используется программа «Имитатор микропрограммируемой микроЭВМ», возможности которой достаточно подробно рассмотрены в разделе 6. После загрузки и запуска программы для лабораторных исследований она переводится в режим ввода микропрограмм, в котором производится ввод разработанной микропрограммы командного цикла.

Отладка микропрограммы осуществляется в режиме выполнения микрокоманд. Ввод и редактирование данных в ЗУ, а также запись начальных адресов микропрограмм операций в преобразователь начального адреса осуществляется с помощью вкладки «ОЗУ, ПНА и ПА».

Отладку программ удобнее выполнять, используя вкладку «ОЗУ, ПНА и ПА», в которой специальным маркером отмечается выполняемая команда и отображается состояние регистров РЗУ. Кроме того, для отладки системы прерывания программ предусмотрено специальное окно (рис. 7.17). Это окно открывается при выполнении команды «Аппаратура прерывания» из группы команд «Режим».

Окно отражает состояние аппаратуры прерывания, после выполнения каждой микрокоманды или команды в зависимости от заданного режим выполнения. На входах аппаратуры прерывания можно задавать и изменять множество запросов на прерывания. Изменение состояния АП происходит под действием управляющих сигналов, в соответствии с микропрограммой разработанной пользователем.

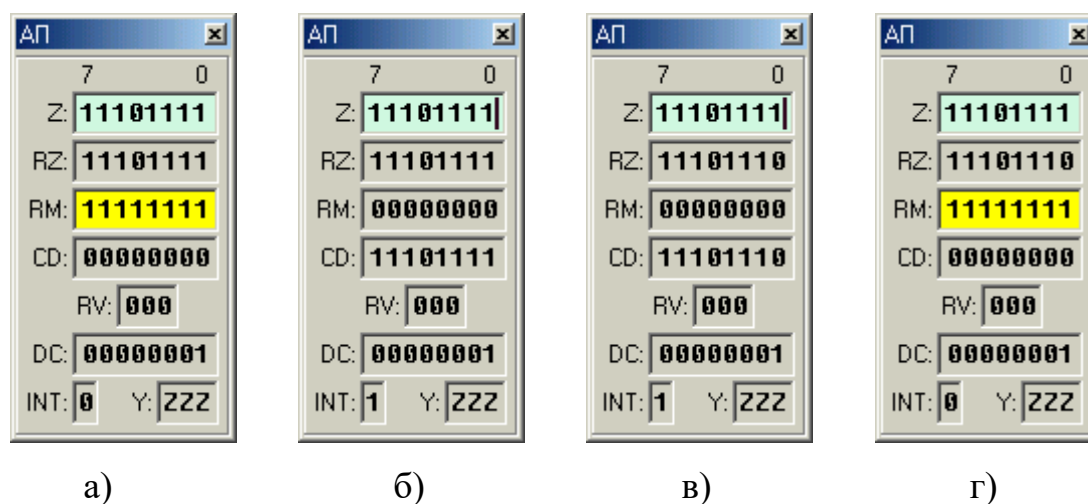


Рис. 7.17. Окно аппаратуры прерывания: прерывания запрещены (а), определение запроса с наивысшим приоритетом (б), сброс запроса, принятого к обработке (в); прерывания снова запрещены (г)

### 7.3.6. Построение диаграммы прерываний

В соответствии с заданием на лабораторную работу, начальное состояние системы прерывания характеризуется набором масок программ и множеством запросов, поступивших в систему (табл. 7.1). Проследим последовательность переключений ЭВМ с одной программы на другую, необходимых для реализации на поступившие в систему запросы на прерывания. Для этого необходимо вывести на экран окно аппаратуры прерывания и воспользоваться покомандным выполнением программ.

ЭВМ начинает реагировать на запросы при выполнении основной программы Р. В процессе выполнения очередной команды этой программы запускается подмикропрограмма INT. Подмикропрограмма INT анализирует наличие незамаскированных запросов на прерывание и считывает из аппаратуры прерывания номер запроса с наивысшим приоритетом. Затем по этому номеру запроса с помощью таблицы прерываний определяется начальный адрес соответствующей прерывающей программы. В рассматриваемом примере такой программой будет программа Р0. Соответствующий выделенному запросу разряд регистра запросов обнуляется, в программный счетчик заносится начальный адрес программы Р0, поэтому следующей выполняемой командой будет первая команда прерывающей программы Р0.

После завершения команды установки маски в процессе выполнения очередной команды программы Р0 вновь производится анализ возможности реакции системы на запрос прерывания. В результате управление получает прерывающая программа Р1. Продолжая процесс выполнения программ можно получить данные, необходимые для построения диаграммы прерываний. Диаграмма прерываний для рассматриваемого примера приведена на рис. 7.18.

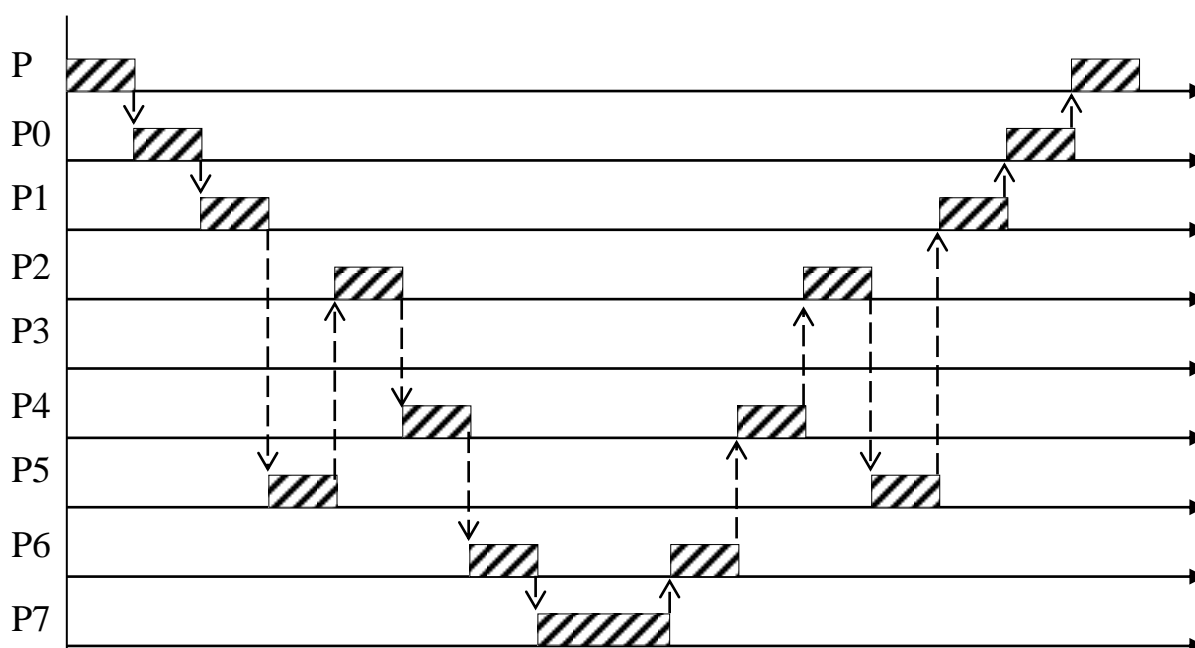


Рис. 7.18. Диаграмма прерываний: Р – основная программа, Р0– Р7 – прерывающие программы

## 7.4. Контрольные вопросы

### *Основные положения*

1. Время реакции системы на прерывание определяется:
  - с момента поступления запроса на прерывание до момента прекращения выполнения текущей программы;
  - с момента поступления запроса на прерывание до момента завершения прерывающей программы;
  - с момента прекращения выполнения текущей программы до момента завершения прерывающей программы;
  - с момента начала до момента завершения прерывающей программы.
2. Время реакции системы на прерывание минимально, если прерывания разрешены после выполнения:
  - любой микрокоманды;
  - определенных типов микрокоманд;
  - любой команды;
  - определенных типов команд.
3. Невозможно при переключении процессора на прерывающую программу в процессе аппаратных прерываний программно сохранить состояние:
  - программного счетчика;
  - аккумулятора
  - регистра базы;
  - регистра признаков.
4. Глубина прерываний в ЭВМ с системой прерываний, использующей для сохранения состояния процессора стек, ограничена:
  - объемом стека;
  - числом программ, последовательно прерывающих друг друга;
  - числом регистров указателя стека;
  - числом сохраняемых регистров процессора.
5. Сократить время переключения ЭВМ на прерывающую программу позволяет использование в процессоре:

- регистрового файла, в котором каждой программе выделен свой набор регистров.

- регистра указателя стека для работы со стеком, размещаемым в памяти ЭВМ;

- программного сохранения состояния регистров при прерывании;

- программного опроса регистра запросов на прерывания.

6. Последовательность из пяти цифр, каждая из которых соответствует одной из фаз процесса прерывания программ: 1 – выполнение прерывающей программы, 2 – выделение запроса на прерывание с наивысшим приоритетом, 3 – переключение процессора на выполнение прерывающей программы, 4 – прекращение выполнения текущей программы и сохранение состояния процессора, 5 – возврат из прерывающей программы и восстановление состояния процессора, причем цифры располагаются в порядке выполнения фаз, имеет следующий вид: \_\_\_\_\_ .

7. Система прерываний входит в особое состояние, когда до завершения выполнения прерывающей программы поступает запрос на прерывание по той же самой причине. Это состояние имеет название \_\_\_\_\_ .

8. Программа может прервать сама себя (да/нет) \_\_\_\_\_ .

### *Исследуемое устройство*

9. Аппаратура прерывания формирует номер прерывающей программы в соответствии с номером:

- разряда регистра запросов, содержащего незамаскированный запрос с наивысшим приоритетом;

- разряда регистра запросов, содержащего замаскированный запрос с наивысшим приоритетом;

- разряда регистра маски, имеющего наивысший приоритет;

- разряда регистра запросов, содержащего незамаскированный запрос с низшим приоритетом.

10. Аппаратура прерывания выдает в процессор запрос на прерывание если:

- есть хотя бы один незамаскированный запрос на прерывание;
- все запросы на прерывание замаскированы;
- есть запрос, принятый на обработку, но выполнение соответствующей

ему прерывающей программы еще не завершено.

- обработка запроса на прерывание завершена.

11. Номер незамаскированного запроса с наивысшим приоритетом при условии, что регистр запросов содержит двоичный код запросов 11110111, регистр маски – двоичный код маски 01110011 (единица в разряде маски маскирует одноименный разряд регистра запросов) и младший – нулевой разряд имеет наивысший приоритет, равен \_\_\_\_\_ .

12. Время обслуживания запроса на прерывание в ЭВМ, когда время реакции системы на прерывание составляет 1 мкс, а время выполнения прерывающей программы – 10 мкс, равно \_\_\_\_\_ мкс.

## 8. ПРИМЕРЫ ЗАДАНИЙ

Многолетний опыт проведения лабораторных работ показал, что использование сложных задач в лабораторном практикуме по организации ЭВМ не оправдано, поскольку увеличивает время на разработку алгоритмов и микропрограмм, отвлекая от основных целей практикума, связанных с разработкой и исследованием устройств. Постепенное упрощение заданий привело к простым задачам, требующим, тем не менее, разработки циклических алгоритмов. Примеры таких заданий приведены табл. 8.1.

Таблица 8.1

Варианты заданий на лабораторные работы

| Вариант<br>(№) | Задание   |
|----------------|---|
| 1              | Определить остаток от деления числа $X$ на число $Y$ ( $X$ и $Y$ – целые числа от 0 до 255)   |
| 2              | Вычислить: $Y = \sum_{i=1}^N i$ , где $N$ может принимать значения от 1 до 22   |
| 3              | Определить номер $N$ разряда двоичного кода числа $X$ , в котором находится первая, начиная с младшего (нулевого) разряда, цифра «1». Число $X$ представлено 8-разрядным двоичным кодом |
| 4              | Выполнить умножение: $Z=XY$ ( $X$ и $Y$ – целые числа от 0 до 255) путем $Y$ -кратного суммирования множимого $X$   |
| 5              | Подсчитать число $N$ комбинаций цифр «01» в 8-разрядном двоичном коде, начиная со старших разрядов  |
| 6              | Выполнить преобразование двоично-десятичного кода (8421) целого числа в диапазоне от 0 до 99 в двоичный код   |
| 7              | Подсчитать число $N$ единиц в 8-разрядном двоичном коде $X$   |
| 8              | Выполнить сдвиг 8-разрядного двоичного кода $X$ на $N$ разрядов в сторону младших разрядов (сдвиг логический)   |
| 9              | Выполнить умножение положительных чисел с фиксированной запятой методом «младшими разрядами вперед». Произведение представлено 16-разрядным, а множители 8-разрядными двоичными числами |
| 10             | Подсчитать число одинаковых цифр, расположенных в одноименных разрядах $x_i$ и $y_i$ 8-разрядных двоичных чисел $X$ и $Y$   |
| 11             | Выполнить сдвиг 8-разрядного двоичного кода $X$ на $N$ разрядов в сторону старших разрядов (сдвиг арифметический)   |
| 12             | Определить номер $N$ разряда двоичного кода числа $X$ , в котором находится первая, начиная со старшего (седьмого) разряда, цифра «0»   |

| Вариант<br>(№) | Задание   |
|----------------|---|
| 13             | Подсчитать число $N$ комбинаций цифр «11» в 8-разрядном двоичном коде, начиная с младших разрядов   |
| 14             | Выполнить сдвиг 8-разрядного двоичного кода $X$ на $N$ разрядов в сторону старших разрядов (сдвиг логический)   |
| 15             | Выполнить умножение положительных чисел с фиксированной запятой методом «старшими разрядами вперед». Произведение представлено 16-разрядным, а множители – 8-разрядными двоичными числами |

Решение задач предполагает анализ особых ситуаций (равенство нулю делителя, отсутствие подсчитываемых комбинаций цифр в двоичном коде и т. п.) и формирование соответствующих признаков. Приветствуются наиболее простые методы и алгоритмы решения задач. Простота методов и алгоритмов не является обязательным требованием при выполнении лабораторных работ, но позволяет ускорить отладку микропрограмм.



## БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Орлов, С. А. Организация ЭВМ и систем : учеб. для вузов / С. А Орлов . – 4-е изд., доп. и перераб. – Санкт-Петербург : Питер, 2018. – 668 с. – ISBN 978-5-4461-0811-4. – Текст : непосредственный.
2. Новожилов, О. П. Архитектура ЭВМ и систем : учеб. пособие для академ. бакалавриата / О. П. Новожилов. – Москва : Юрайт, 2018. – 527 с. – ISBN 978-5-9916-1658-4. – Текст : непосредственный.
3. Страбыкин, Д. А. Организация ЭВМ : лаб. практикум на компьютерах : учеб. пособие / Д. А. Страбыкин. – 3-е изд., перераб. и доп. – Киров : ВятГУ, 2013. – 162 с. – Текст : непосредственный.
4. Паттерсон, Д. Архитектура компьютера и проектирование компьютерных систем / Д. Паттерсон, Дж. Хеннесси. – 4-е изд. – Санкт-Петербург : Питер, 2012. – 784 с. – (Классика Computers Science). – ISBN 978-5-459-00291-1. – Текст : непосредственный.
5. Жмакин, А. П. Архитектура ЭВМ : учеб. пособие / А. П. Жмакин. – 2-е изд., перераб. и доп. – Санкт-Петербург : БХВ-Петербург, 2010. – 352 с. – ISBN 978-5-9775-0550-5. – Текст : непосредственный.

Учебное издание

Страбыкин Дмитрий Алексеевич

ОРГАНИЗАЦИЯ ЭВМ:  
ЛАБОРАТОРНЫЙ ПРАКТИКУМ  
НА КОМПЬЮТЕРАХ

Учебное пособие

4-е издание, переработанное и дополненное

Авторская редакция  
Тех. редактор Нилова А. В.

Подписано в печать 30.07.2020. Печать цифровая. Бумага для офисной техники.

Усл. печ. л. 11,04. Тираж 5 экз. Заказ № 6301

Федеральное государственное бюджетное образовательное учреждение  
высшего образования «Вятский государственный университет»

610000, г. Киров, ул. Московская, 36, тел.: (8332) 64-23-56, <http://vyatsu.ru>

