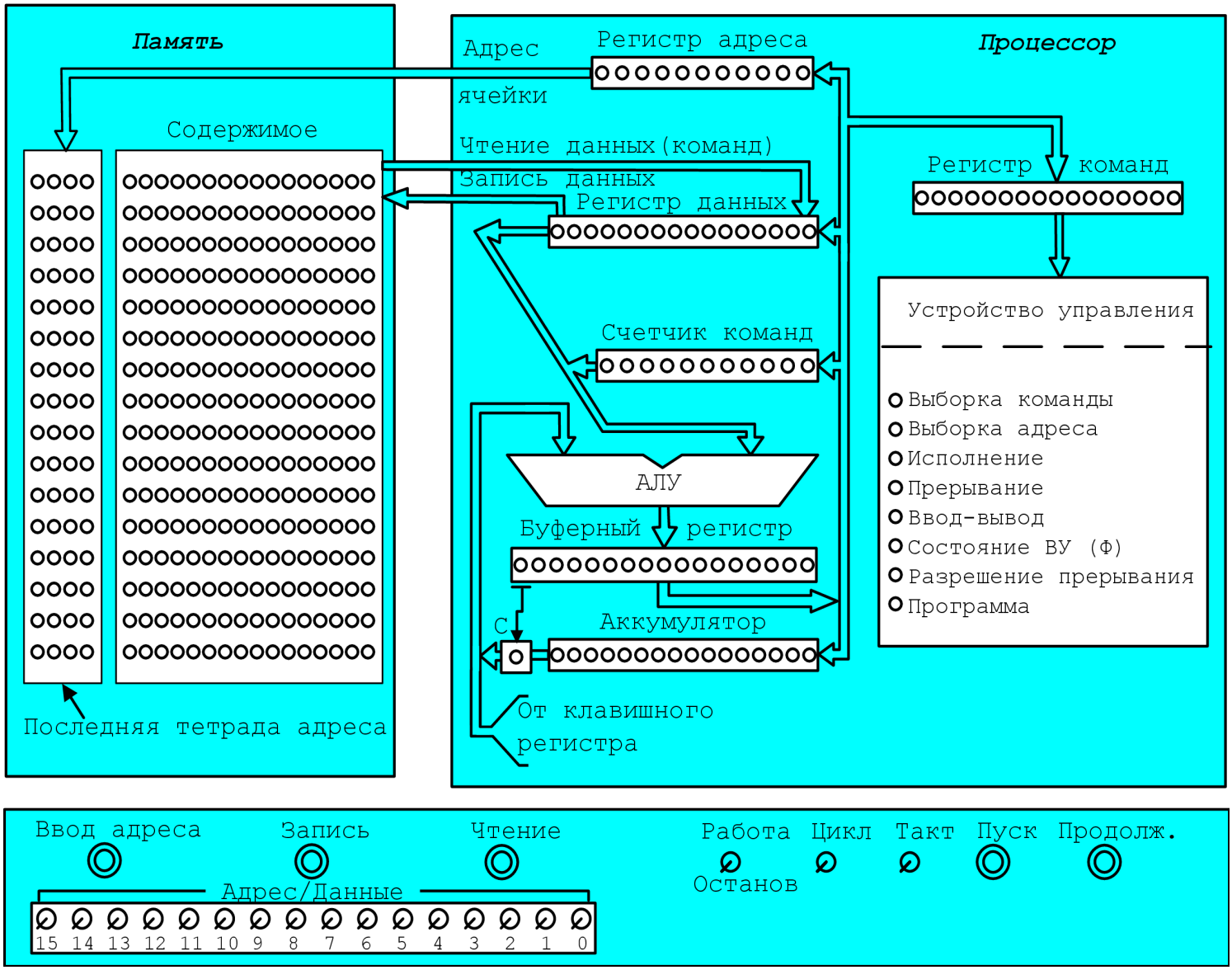
В.В. Кириллов, А.А. Приблуда, С.В. Клименков, Д.Б. Афанасьев

Методические указания к лабораторным работам по курсу "Основы вычислительной техники"



Санкт-Петербург, 2015
версия 0.74 16.03.16

Выходные данные, аннотация

Содержание

Настоящее методическое пособие предназначено для практического закрепления материала по дисциплине "Основы вычислительной техники" (*ОВТ*), преподавание которой организовано по модульному принципу и включает лекции, лабораторные работы, домашние задания и контрольные работы. Курс по основам вычислительной техники и оригинальная модель базовой ЭВМ был разработан в начале 1980-х годах Кирилловым Владимиром Васильевичем и Приблудой Анатолием Андреевичем. В учебно-методической работе, разработке методических указаний, учебников и моделей базовой ЭВМ, лабораторных работ в разное время принимало участие большое количество сотрудников кафедры вычислительной техники, среди которых особенно хотелось отметить (перечисление в алфавитном порядке) Афанасьева Дмитрия Борисовича, Блохину Елену Николаевну, Гаврилова Антона Валерьевича, Громова Геннадия Юрьевича, Громову Ирину Владимировну, Дергачева Андрея Михайловича, Клименкова Сергея Викторовича, Лемешева Алексея Сергеевича, Максимова Андрея Николаевича, Майорова Сергея Александровича, Мартьянова Николая Васильевича, Перминова Илью Валентиновича, Приблуду Андрея Анатольевича, Приблуду Константина Анатольевича, Щелокова Ивана Викторовича, а также большое количество студентов, аспирантов и выпускников кафедры ВТ.

В методическом пособии содержится информация, необходимая для успешной сдачи всех лабораторных работ, включая специально разработанную в образовательных целях учебную ЭВМ (*базовая ЭВМ*), обладающую типичными чертами многих современных ЭВМ. Лабораторные работы раздела №1 предназначены для демонстрации студентам первого курса важных предметов старших курсов: "Системное программное обеспечение" и "Системы баз данных", а также знакомства с лабораториями, где им предстоит обучаться в дальнейшем. Остальные лабораторные работы курса посвящены базовой ЭВМ. С ее помощью студенты исследуют порядок функционирования ЭВМ при выполнении программ различных типов, подходы к организации ввода-вывода информации, принципы микропрограммного управления. В приложениях приведена справочная информация, необходимая при подготовке и защите лабораторных работ.

Раздел 1. Знакомство с кафедрой вычислительной техники

Кафедра вычислительной техники

Кафедра вычислительной техники (*ВТ*) ведет подготовку бакалавров по направлениям подготовки 09.03.01 - Информатика и Вычислительная техника (профиль подготовки "Вычислительные машины, комплексы, системы и сети") и 09.03.04 - Программная инженерия (профиль подготовки "Разработка программно-информационных систем"). В рамках курса ОВТ ведущие преподаватели направлений подготовки прочитают вводные лекции по своим дисциплинам.

Лабораторные работы курса ОВТ

В рамках курсах предусмотрено 8 лабораторных работ. Результатом выполнения работы является выполнение всех требований к работе и отчет, который должен включать ряд обязательных составляющих. К ним относятся:

- титульный лист: название университета, кафедры, дисциплины, название и номер лабораторной работы, номер группы и варианта, Ф.И.О. студента, год;
- задание к работе, включая вариант задания;
- порядок выполнения лабораторной работы, дополнительные требования и указания, которые находятся в описании к каждой работе;
- выводы, которые отвечают на вопросы "Что было изучено при выполнении лабораторной работы? Что нового вы узнали? Как можно использовать изученный материал?";
- скрепу, скобу или люверс. Листы должны быть скреплены между собой!

Лабораторная работа №0.
Основные команды ОС семейства UNIX

В лабораторных аудиториях кафедры установлено разнообразное вычислительное оборудование под управлением различных операционных систем (*ОС*). Важное место занимают ОС семейства UNIX, включая различные версии Linux, BSD, Solaris и AIX. Основным способом взаимодействия пользователей и администраторов с такими операционными системами является командный интерфейс с использованием интерпретатора shell.

Цель работы. Знакомство с основным способом взаимодействия с ОС UNIX, командным интерфейсом, а также базовой функциональностью интерпретатора shell. Получение основных сведений о файловой системе и правах доступа к файлам.

Задание.

1. Создать приведенное в варианте дерево каталогов и файлов с содержимым. В качестве корня дерева использовать каталог `lab0` своего домашнего каталога. Для создания и навигации по дереву использовать команды: `mkdir`, `echo`, `cat`, `touch`, `ls`, `pwd`, `cd`, `more`, `cp`, `rm`, `rmdir`, `mv`.
2. Установить согласно заданию права на файлы и каталоги при помощи команды `chmod`, используя различные способы указания прав.
3. Скопировать часть дерева и создать ссылки внутри дерева согласно заданию при помощи команд `cp` и `ln`, а также команды `cat` и перенаправления ввода-вывода.
4. Используя команды `find`, `ls`, `head`, `tail`, `echo`, `cat`, `wc` выполнить в соответствии с вариантом задания поиск и фильтрацию файлов, каталогов и содержащихся в них данных.
5. Выполнить удаление файлов и каталогов при помощи команд `rm` и `rmdir` согласно варианту задания.

Подготовка к выполнению работы. Изучить справочные страницы по указанным командам. Разобраться с основными принципами организации ввода-вывода с использованием стандартных потоков ввода-вывода (`stdin`, `stdout`, `stderr`, включая перенаправление данных потоков на другие команды-фильтры и в файлы), типами файлов, правами пользователей на доступ к файлу для операций чтения, записи и исполнения для владельца файла, группы владельца и остальных пользователей системы.

Порядок выполнения работы. Создать указанное в п. 1 задания дерево файлов и каталогов. Обратить внимание на точное соответствие всех атрибутов полученного дерева заданию. Последовательность команд, необходимых для создания дерева, записать в файл для возможности автоматического повтора п.1, по одной команде на строке. Изменить права на файлы, согласно п.2 задания, при этом последовательность команд для изменения прав добавить в файл создания дерева. Выполнить п.3 задания, включив в файл команды. Выполнить запросы поиска по дереву (п.4), сохранить и включить в отчет вывод исполненных команд. Показать полученное дерево преподавателю. Выполнить команды удаления п.5. Включить в отчет последовательность команд удаления с результатом их выполнения.

Содержание отчета по работе. В дополнение к общим обязательным требованиям, отчет должен содержать:

- Иерархию файлов и каталогов, полученную при помощи команд `find . -ls` из директории `lab0`, после выполнения п.3 задания.
- Задания, команды и результаты их выполнения для п.4 и п.5.
- Файл с последовательностью команд по всей лабораторной работе.

Контрольные вопросы:

1. Расскажите про команду {*имя команды*}. Какие она принимает аргументы (их количество, формат, способ задания)? Является ли данная команда фильтром?

2. Стандартные потоки ввода-вывода, назначение. Способы управления стандартными потоками в shell.
3. Стандартные права доступа к файлам и каталогам. Способы задания прав при помощи команды `chmod`. Интерпретация вывода команды `ls -l`.
4. Файлы в ОС UNIX. Типы файлов. Символические и жесткие ссылки.

Лабораторная работа №1.
Введение в базы данных

Базы данных получили широкое распространение в современном обществе, и используются для хранения больших объемов структурированных данных. Подробные курсы по базам данных будут проходить в последующих семестрах.

Цель работы. Знакомство с основными современными понятиями, используемы в теории баз данных, табличным способом представления данных, моделью "сущность-связь", основами языка запросов к БД SQL.

Задание. По варианту, выданному преподавателем, составить и выполнить запросы к базе данных "Учебный процесс".

Подготовка к выполнению работы. Изучить основные понятия теории базы данных реляционной алгебры. Изучить синтаксис и возможности оператора SQL `SELECT` для запросов по одной таблице, включая сортировку, группировку, встроенные функции, выборку уникальных строк и синтаксис фразы `WHERE`.

Порядок выполнения работы. Пункты задания необходимо выполнять строго по порядку, т. к. они сформированы от простых запросов к сложным. Прочитайте внимательно условия для запроса. Найдите подходящую таблицу в БД "Учебный процесс" (Приложение 2), сформируйте и выполните запрос к таблице. Проверьте корректность выдаваемых результатов.

Содержание отчета по работе. В дополнение к общим обязательным требованиям, отчет должен содержать:

- Текст задания для запроса.
- Выполняющий задание запрос `SELECT` и первые 5 строк результата запроса.

Контрольные вопросы:

1. Назначение БД, реляционное представление данных.
2. Таблицы и основные операции над ними — селекция и проекция.
3. Оператор `SELECT` синтаксис и использование.
4. Арифметические и логические операторы (`+, -, *, /, =, >, <, <>, ||, AND, BETWEEN, IN, LIKE, NOT, OR`), функции SQL (`ABS, ROUND, SING, TRUNC, CONCAT, INTCAP, LENGHT, SUBSTR, TRIM, LOWER, UPPER, MONTH_BETWEEN, GREATEST, LEAST`) и агрегатные функции (`AVG, COUNT, MAX, MIN, SUM`).
5. Функции `CASE` и `DECODE`
6. Сортировка таблиц с помощью фразы `ORDER BY`, порядок сортировки.
7. Группировка таблиц с помощью фразы `GROUP BY`.

Для подготовки необходимо использовать главы 1-5 [2].

Рубежный контроль №1

Рубежный контроль проводится в виде теоретически-практического опроса по изученному материалу. При ответе на его вопросы необходимо показать знания и умения в рамках курса лекций, а также практических занятий по ОС Unix и базам данных. Уровень сложности вопросов аналогичен или проще соответствующих контрольных вопросов для подготовки к лабораторным работам.

Раздел 2. Введение в базовую ЭВМ

В рамках этого раздела студент должен изучить состав, структуру и принцип функционирования БЭВМ на уровне машинных команд.

Лабораторная работа №2.
Исследование работы БЭВМ

Цель работы - изучение приемов работы на базовой ЭВМ и исследование порядка выполнения арифметических команд и команд пересылки.

Задание. По выданному преподавателем варианту определить функцию, вычисляемую программой, область представления и область допустимых значений исходных данных и результата, выполнить трассировку программы, предложить вариант с меньшим числом команд. При выполнении работы представлять результат и все операнды арифметических операций знаковыми числами, а логических операций беззнаковым набором из шестнадцати логических значений.

Подготовка к выполнению работы. Познакомиться с устройством и системой команд базовой ЭВМ (см. приложение В, п.п. 1.1 — 1.6), порядком выполнения машинных команд (п.1.9), инструкцией по работе с моделью базовой ЭВМ (см. приложение Г). Изучить представления в БЭВМ числовых и логических значений.

Порядок выполнения работы. Восстановить текст заданного варианта программы, написать назначение программы и реализуемые ею функции (формулы).

Получить у преподавателя исходные данные, занести в память базовой ЭВМ заданный вариант программы и, выполняя ее по командам, заполнить таблицу трассировки выполненной программы. Таблицу трассировки подписать у преподавателя!

Содержание отчета по работе. В дополнение к общим обязательным требованиям, отчет должен содержать:

1. Текст исходной программы по следующей форме:

Оформление текста программы для л/р №2-5. *Таблица 2.1*

Адрес	**Код команды**	**Мнемоника**	**Комментарии**
021	4015	ADD 15	Добавить содержимое ячейки памяти 15 к аккумулятору

2. Описание программы:
   - назначение программы и реализуемые ею функции (формулы);
   - область представления и область допустимых значений исходных данных и результата;
   - расположение в памяти ЭВМ программы, исходных данных и результатов;
   - адреса первой и последней выполняемой команд программы.
3. Таблица трассировки должна быть представлена в соответствии с форматом:

Форма таблицы трассировки выполнения команд. *Таблица 2.2*

Выполняемая команда		**Содержимое регистров процессора после выполнения команды.**						**Ячейка, содержимое которой изменилось после выполнения команды**	
Адрес	Код	СК	РА	РК	РД	А	С	Адрес	Новый код
xxx	xxxx	xxx	xxx	xxxx	xxxx	xxxx	x	xxx	xxxx

4. Вариант программы с меньшим числом команд.

Контрольные вопросы:

1. Форматы представления в БЭВМ целых чисел со знаком и логических значений.
2. Представление чисел в разрядной сетке в прямом, обратном и дополнительном кодах.
3. Описание команды находящейся, по указанному адресу: наименование, назначение, тип команды и вид адресации. Количество и название машинных

циклов, потактовое выполнение команды.

4. Какую формулу реализует программа? Как можно упростить программу?
5. Где находятся аргументы программы? Где находится результат? Как они представлены? Какие дополнительные ячейки использует программа? Для чего?
6. В каком случае можно расширить область допустимых значений исходных данных?
7. Какое количество обращений к ячейкам памяти при выполнении безадресной команды? На каких циклах оно выполняется?

Домашнее задание раздела 2

Цель работы - Закрепление навыков разработки линейных программ для БЭВМ с использованием целочисленной знаковой арифметики.

Задание. По выданному варианту разработать программу, реализующую заданную функцию. Для использования операций умножения и деления использовать команды сдвига и сложения с сохранением промежуточных результатов. Проверить функционирование программы на выданных исходных данных. Заполнить таблицу трассировки для одного выбранного набора исходных данных.

Содержание отчета по работе. Содержание отчета должно соответствовать общим требованиям, а также содержать п.п.1-3 из требований к отчету для лабораторной работы №2.

Дополнительные требования. При разработке программы необходимо учитывать следующее:

- исходные данные и результат являются знаковыми числами;
- значение X является неотрицательным;
- умножение на 48 реализовывать сорока восемью последовательными сложениями запрещено!

Рубежный контроль №2

В рубежном контроле второго раздела необходимо заполнить таблицу трассировки правильными значениями и восстановить формулу, вычисляемую программой. Задание аналогично лабораторной работе №2.

Пример задания и правильного ответа рубежного контроля №2. *Таблица 2.3*

Исходные данные			**Пример правильного ответа. Формула: C=2A+B**							
Адрес	**Команда/данные**		**Адрес**	**Команда/данные**	**СК**	**РА**	**РК**	**РД**	**A**	**C**
00A	38BA		00A	38BA	–	–	–	–	–	–
00B	02EF		00B	02EF	–	–	–	–	–	–
00C	0000		00C	0000	–	–	–	–	–	–
00D	+CLA		00D	+CLA	00E	00D	F200	F200	0000	0
00E	ADD 00A		00E	ADD 00A	00F	00A	400A	38BA	38BA	0
00F	CLC		00F	CLC	010	00F	F300	F300	38BA	0
010	ROL		010	ROL	011	010	F600	F600	7174	0
011	ADD 00B		011	ADD 00B	012	00B	400B	02EF	7463	0
012	MOV 00C		012	MOV 00C	013	00C	300C	7463	7463	0
013	HLT		013	HLT	014	013	F000	F000	7463	0

Раздел 3. Исследование возможностей базовой ЭВМ

В рамках третьего раздела студент должен более подробно познакомится с системой команд БЭВМ, детальной последовательностью исполнения команд с прямой и косвенной адресациями, подпрограммами, основными подходами, применяемыми для низкоуровневой обработки данных.

Лабораторная работа №3.
В*ыполнение циклических программ*

Цель работы - изучение способов организации циклических программ и исследование порядка функционирования БЭВМ при выполнении циклических программ.

Задание. По выданному преподавателем варианту восстановить текст заданного варианта программы, определить предназначение и составить описание программы, определить область представления и область допустимых значений исходных данных и результата, выполнить трассировку программы.

Подготовка к выполнению работы.

Получить у преподавателя номер варианта и исходные данные к лабораторной работе. Изучить способы и средства организации циклических программ с использованием системы команд базовой ЭВМ (приложение В, п.1.7, примеры 1 и 2). Восстановить текст заданного варианта программы. Составить описание программы.

Порядок выполнения работы. Получить допуск к лабораторной работе, предъявив преподавателю подготовленные материалы. Занести в память базовой ЭВМ заданный вариант программы и заполнить таблицу трассировки, выполняя эту программу по командам. Таблицу трассировки подписать у преподавателя!

Содержание отчета по работе. Отчет по работе должен быть составлен аналогично лабораторной работе №2, за исключением п. 4 (разработка программы с сокращенным числом команд). Необходимо привести диапазон *всех* ячеек памяти, где может размещаться массив исходных данных.

Контрольные вопросы:

1. Сравнение значений в БЭВМ. Команды условных и безусловного переходов.
2. Организация циклических вычислений. Команда `ISZ`.
3. Прямая и косвенная адресация, индексные ячейки.
4. Описание команд `ADD`, `AND`, `BR`, `BEQ`, `BMI`, `BPL`, `BCS`, `ISZ` с прямой и косвенной адресациями: наименование, назначение, тип команды и вид адресации. Количество и название машинных циклов, потактовое выполнение команд.
5. Количество обращений к памяти команд БЭВМ с прямой и косвенной адресациями.
6. Где находятся аргументы программы? Где находится результат? Как они представлены?

Лабораторная работа №4.
В*ыполнение комплекса программ*

Цель работы - изучение способов связи между программными модулями, команды обращения к подпрограмме и исследование порядка функционирования БЭВМ при выполнении комплекса взаимосвязанных программ.

Задание. По выданному преподавателем варианту восстановить текст заданного варианта программы и подпрограммы (программного комплекса), определить предназначение и составить его описание, определить область представления и область допустимых значений исходных данных и результата, выполнить трассировку программного комплекса.

Подготовка к выполнению работы.

Получить у преподавателя номер варианта и исходные данные к

лабораторной работе. Изучить способы связи между программными модулями и команды обращения к подпрограмме в базовой ЭВМ (приложение В, п. 1.8). Восстановить текст заданного варианта программного комплекса, составить его описание.

Порядок выполнения работы. Получить допуск к лабораторной работе, предъявив преподавателю подготовленные материалы. Занести в память базовой ЭВМ заданный вариант программного комплекса и заполнить таблицу трассировки, выполняя этот комплекс по командам. Подписать таблицу трассировки!

Содержание отчета по работе. Отчет по работе должен быть составлен аналогично лабораторной работе №2, за исключением п. 4 (разработка программы с сокращенным числом команд). Необходимо привести диапазон *всех* ячеек памяти, где может размещаться массив исходных данных.

Контрольные вопросы:

1. Организация подпрограмм в БЭВМ. Команды вызова подпрограммы и возврата. Недостатки существующей реализации.
2. Аргументы и возвращаемые значения подпрограммы. Способы организации передачи аргументов и возвращаемых значений.
3. Рекурсивный вызов подпрограмм. Организация стека.
4. Описание команды `JSR` с прямой и косвенной адресациями: наименование, назначение, тип команды и вид адресации. Количество и название машинных циклов, потактовое выполнение команды, количество обращений к памяти.

Рубежный контроль раздела 3

Рубежный контроль раздела 3 предназначен для выполнения трассировки по микрокомандам заданной в варианте команды с косвенной адресацией. При выполнении рубежного контроля можно использовать таблицу интерпретатора базовой ЭВМ (Приложение В, табл В.10).

Задание. Запишите последовательность микрокоманд для выполнения команды. Заполните таблицу значениями после выполнения каждой микрокоманды.

Пример задания и правильного ответа рубежного контроля раздела 3. Таблица 3.1

Исходные данные: команда 00E SUB (000), код команды: 6800												
СчМК до выборки МК	**Содержимое памяти и регистров процессора после выборки и исполнения микрокоманды**											
	яч. 000	**РМК**	**СК**	**РА**	**РК**	**РД**	**А**	**С**	**БР**	**N**	**Z**	**СчМК**
	F00E	0000	00E	000	0000	E3BB	85F5	1	00000	1	0	
Пример правильного ответа												
01	F00E	0300	00E	000	0000	E3BB	85F5	1	0000E	1	0	02
02	F00E	4001	00E	00E	0000	E3BB	85F5	1	0000E	1	0	03
03	F00E	0311	00E	00E	0000	6800	85F5	1	0000F	1	0	04
04	F00E	4004	00F	00E	0000	6800	85F5	1	0000F	1	0	05
05	F00E	0100	00F	00E	0000	6800	85F5	1	06800	1	0	06
06	F00E	4003	00F	00E	6800	6800	85F5	1	06800	1	0	07
07	F00E	AF0C	00F	00E	6800	6800	85F5	1	06800	1	0	0C
0C	F00E	AB1D	00F	00E	6800	6800	85F5	1	06800	1	0	0D
0D	F00E	0100	00F	00E	6800	6800	85F5	1	06800	1	0	0E
0E	F00E	4001	00F	000	6800	6800	85F5	1	06800	1	0	0F
0F	F00E	0001	00F	000	6800	F00E	85F5	1	00000	1	0	10
10	F00E	A31D	00F	000	6800	F00E	85F5	1	06800	1	0	1D
1D	F00E	EF2D	00F	000	6800	F00E	85F5	1	06800	1	0	1E
1E	F00E	0100	00F	000	6800	F00E	85F5	1	0F00E	1	0	1F
1F	F00E	4001	00F	00E	6800	F00E	85F5	1	0F00E	1	0	20
20	F00E	EE27	00F	00E	6800	F00E	85F5	1	06800	1	0	27
27	F00E	0001	00F	00E	6800	6800	85F5	1	00000	1	0	28
28	F00E	AD2B	00F	00E	6800	6800	85F5	1	06800	1	0	29
29	F00E	AC43	00F	00E	6800	6800	85F5	1	06800	1	0	43
43	F00E	1190	00F	00E	6800	6800	85F5	1	11DF5	1	0	44
44	F00E	4075	00F	00E	6800	6800	1DF5	1	11DF5	0	0	45
45	F00E	8390	00F	00E	6800	6800	1DF5	1	00081	0	0	90

Раздел 4. Организация ввода-вывода информации в БЭВМ

Основным назначением вычислительных устройства является обработка внешней по отношению к ЭВМ информации. Для того, что бы получать ее извне, обрабатывать и передавать результаты обработки используются внешние (по отношению к ЭВМ) устройства, такие как клавиатура, мышь, дисплей, принтер и т.д. Вывод-вывод информации на эти устройства должен быть специально организован. Данный раздел предназначен для изучения организации ввода-вывода БЭВМ.

Лабораторная работа №5.
Асинхронный обмен данными с ВУ

Цель работы - изучение организации системы ввода-вывода базовой ЭВМ, команд ввода-вывода и исследование процесса функционирования ЭВМ при обмене данными по сигналам готовности внешних устройств (ВУ).

Задание. По выданному преподавателем варианту разработать программу асинхронного обмена данными с внешним устройством. При помощи программы осуществить ввод или вывод информации, используя в качестве подтверждения данных сигнал (кнопку) готовности ВУ.

Подготовка к выполнению работы.

Изучить организацию системы ввода-вывода и команды ввода-вывода базовой ЭВМ, организацию асинхронного программно-управляемого обмена данными (Приложение В, п.п.2.1-2.3, пример 3). Разработать заданную программу и составить ее описание. Команды программы, используемые переменные и коды символов необходимо разместить в указанных ячейках. Закодировать строку в заданной кодировке, а также в кодировках UTF-8 и UTF-16.

Порядок выполнения работы

1. Получить допуск к лабораторной работе, предъявив преподавателю подготовленные материалы.
2. Разработать и занести в БЭВМ программу, при необходимости ввести данные.
3. Предъявить преподавателю заданную работающую программу, выполняющую в автоматическом режиме ввод-вывод символов.
4. В режиме покомандного выполнения программы ввести (вывести) два первых символов заданного слова, заполняя таблицу трассировки.
5. Перевести ЭВМ в режим автоматического выполнения программы и ввести (вывести) остальные символы заданного слова. Таблицу трассировки подписать у преподавателя!

Содержание отчета по работе. Отчет по работе должен быть составлен аналогично лабораторной работе №2, за исключением п. 4 (разработка программы с сокращенным числом команд). Кроме того, отчет должен содержать заданное слово и коды его символов.

Контрольные вопросы:

1. Синхронный и асинхронный режимы передачи данных.
2. Программно-управляемый и управляемый прерываниями ввод-вывод, прямой доступ к памяти. Преимущества и недостатки.
3. Способы и формат представления символьных и строковых данных в БЭВМ. Кодировки ASCII, КОИ-8, Windows-1251, ISO-8859-5, UTF-8, UTF-16.
4. Порядок байтов в памяти от младшего к старшему (little-endian) и от старшего к младшему (big-endian).
5. Система команд ввода-вывода БЭВМ. Команды `IN`, `OUT`, `CLF`, `TSF` - название, назначение и тип команды. Количество и название машинных циклов, потактовое выполнение команды, с перечислением всех шин, участвующих в обмене.
6. Какие режимы передачи данных и управления вводом-выводом реализуемы в БЭВМ? Почему не возможно реализовать другие?
7. Может ли ВУ определить в каком режиме с ним работают?

8. Назначение флага готовности ВУ, регистра данных ВУ (*РДВУ*), флага состояния ВУ (*ФГВУ*)?
9. Какие элементы БЭВМ участвуют в обмене с ВУ? Укажите направление передачи данных между элементами при операциях ввода и вывода.
10. Опрашивает ли ВУ состояние флага готовности ВУ, если да то при каких условиях?

Лабораторная работа №6.
Обмен данными с ВУ по прерыванию

Цель работы - изучение организации процесса прерывания программы и исследования порядка функционирования ЭВМ при обмене данными в режиме прерывания программы.

Задание. По выданному преподавателем варианту разработать и исследовать работу комплекса программ обмена данными в режиме прерывания программы. Основная программа должна наращивать на N содержимое заданной ячейки памяти (X), которое должно быть представлено как **8-ми битовое знаковое число**. В цикле наращивания необходимо предусмотреть ограничение для такого представления. Программа обработки прерывания должна выводить на ВУ-3 модифицированное значение X в соответствии с заданием. При реализации деления обязательно учитывать возможность отрицательного делимого. Вывод результатов осуществляется с дополнительным использованием программно-управляемого ввода-вывода по готовности ВУ-3.

Подготовка к выполнению работы

Изучить организацию в базовой ЭВМ программно-управляемого обмена данными в режиме прерывания программы (Приложение В, п.2.4, пример 4). Разработать комплекс программ, указанный в задании. Составить методику проверки правильности выполнения разработанного комплекса программ на БЭВМ, т.е. написать последовательность действий оператора (пользователя) БЭВМ, которые необходимо выполнить, чтобы проверить все возможные режимы работы комплекса программ (при появлении запроса прерывания от любого ВУ) и получить заданное количество результатов. Пример методики см. в разделе содержание отчета.

Порядок выполнения работы.

1. Получить допуск к лабораторной работе, предъявив преподавателю подготовленные материалы.
2. Занести разработанный комплекс программ в память БЭВМ.
3. В присутствии преподавателя провести тестирование работающего комплекса программ, используя разработанную методику проверки.
4. Используя методику проверки разработанного комплекса программ, получить три пары результатов, указывая для каждого выведенного значения величину X.
5. Результаты работы программного комплекса представить в виде таблицы результатов работы комплекса.

Содержание отчета по работе. В дополнение к общим обязательным требованиям, отчет должен содержать:

1. Текст исходной программы на языке Ассемблера БЭВМ (синтаксис и особенности приведены в Приложении Д).
2. Полностью разработанную и проверенную на БЭВМ методику проверки.

Пример. Начальный фрагмент методики проверки.

1. Загрузить комплекс программ в память базовой ЭВМ.
2. Запустить основную программу в автоматическом режиме с адреса XXX.
3. Установить "Готовность ВУ-3".
4. Дождаться сброса готовности ВУ-3.
5. Перевести БЭВМ в режим "Останов".
6. …

Контрольные вопросы:

1. Особенности организации программ обмена данными с использованием прерываний. Сохранение и восстановление значений регистров.
2. Команды работы разрешения/запрещения прерывания БЭВМ. Название, назначение и тип команды. Количество и название машинных циклов, потактовое выполнение команды.
3. Вектора прерываний. Преимущество использования векторов прерываний.
4. Потактовое выполнение цикла прерывания после команды `BR (0)`.
5. Когда выполняется цикл обработки прерывания? После каких команд он не выполняется? Почему?
6. Обрабатываются ли прерывания в пошаговом режиме (режиме останов) работы программы? Почему?
7. Что происходит при одновременном поступлении сигнала готовности нескольких внешних устройств? В какой последовательности они будут обработаны?
8. Почему не будет обработано прерывание сразу после выполнения команды EI? Для чего сделано так, что после цикла исполнения команды EI цикл прерывания не вызывается и прерывания не обрабатываются?
9. За что отвечают биты 4, 5, 6 и 7 регистра состояния? Когда изменяется их значение?

Рубежный контроль раздела 4

Рубежный контроль проводится в виде теоретически-практического опроса по изученному материалу. При ответе на его вопросы необходимо показать знания и умения в рамках лекционных и практических занятий по общим вопросам организации и структуры ЭВМ, таких как общая организация, организация памяти, подсистемы ввода-вывода, контроллеров ввода-вывода и принципов обмена ЭВМ и внешних устройств. Для подготовки необходимо использовать базовый учебник [1] и конспект лекций.

Раздел 5. Организация микропрограммного устройства БЭВМ

Микропрограммное устройство предназначено для исполнения команд БЭВМ. В данном разделе изучается его состав, структура и принцип работы.

Лабораторная работа №7. *Синтез команд БЭВМ*

Цель работы - практическое освоение принципов микропрограммирования и разработки адресных и безадресных команд.

Задание. Синтезировать цикл исполнения для выданных преподавателем команд. Предложить мнемоническое обозначение команды, объяснить его. Разработать тестовые программы, которые проверяют каждую из синтезированных команд. Загрузить в микропрограммную память БЭВМ циклы исполнения синтезированных команд, загрузить в основную память БЭВМ тестовые программы. Проверить и отладить разработанные тестовые программы и микропрограммы.

Подготовка к выполнению работы.

Получить у преподавателя вариант задания. Изучить организацию микропрограммного устройства базовой ЭВМ, (Приложение В раздел 3).

1. Синтезировать завершающие вертикальные микрокоманды цикла исполнение следующих команд:

- команда 7xxx — команда, предназначенная для выполнения арифметических и других операций с памятью;
- команда Dxxx — команда, осуществляющая переход по заданному условию;
- Безадресная команда с кодом FCXX, FDXX, FEXX или FFXX— осуществляет

операцию с аккумулятором.

2. Написать тестовые программы для проверки правильности исполнения синтезированных команд базовой ЭВМ. Тестовые программы должны отвечать следующим требованиям:

- Тестовая программа должна состоять из отдельных тестовых блоков (частей программы или подпрограмм), которые проверяют различные результаты выполнения команды. Количество таких тестовых блоков необходимо согласовать с преподавателем.
- Каждый тестовый блок должен в случае корректной работы микропрограммы записывать 1 в выбранную ячейку памяти. Если микропрограмма работает некорректно, тест должен обнулять выбранную ячейку.
- Тестовая программа должна проверить что все тестовые блоки завершились корректно и записать 1 в выбранную ячейку памяти.
- Для синтезированных арифметических и безадресных команд результат их выполнения должен быть зафиксирован в выбранной ячейке памяти БЭВМ.
- Если проверяемая арифметическая или безадресная команда устанавливает признаки результата (биты C,Z,N), необходимо проверить правильную установку только одного из них, используя соответствующую команду перехода.
- Для синтезированных команд переходов необходимо проверить команду как при выполнении условия перехода, так и при его невыполнении.

Таким образом, после выполнения правильно разработанной тестовой программы в автоматическом режиме в памяти базовой ЭВМ будет размещена информация, позволяющая однозначно подтвердить правильность выполнения синтезированной команды.

3. При разработке микропрограмм заданных команд следует иметь в виду:

- В процессе дешифрации команды 7xxx в РА записывается адрес операнда (может использоваться для команд пересылки), а в РД - сам операнд (может использоваться для команд загрузки и сравнения). Затем осуществляется переход к ячейке памяти микрокоманд B0, где надо разместить первую синтезируемую микрокоманду команды 7xxx.
- После выборки команды перехода Dxxx в РД сохраняется адрес перехода (адресная часть команды), который может быть переписан в СК при выполнении условия перехода. Последняя микрокоманда дешифрации команды Dxxx передает управление в ячейку с адресом D0, где надо разместить первую синтезируемую микрокоманду команды Dxxx.
- Когда в процессе дешифрации безадресных команд выясняется, что в 10-м и 11-м разрядах РК содержатся единицы(т.е. выбрана одна из команд: FCXX, FDXX, FEXX или FFXX), управление передается в ячейку с адресом E0. Здесь должны начинаться микрокоманды дополнительной дешифрации, выделяющие заданную команду путем анализа 9-го и 8-го разрядов РК и передающие управление в свободную область памяти микрокоманд(от Ex до FF), где следует разместить микрокоманды реализации безадресной команды.
- Все микропрограммы реализуемых команд должны заканчиваться микрокомандой 8390 (GOTO ПРЕ(90)), осуществляющей переход к микрокомандам, завершающим исполнение команды БЭВМ.

Пример. FF00 - инверсия содержимого аккумулятора и очистка регистра С.

Реализация цикла исполнения для команды "COMACLC". *Таблица 5.1*

Адрес МП	Микро-команды	Действие : Комментарии
E0	A990	IF BIT(9,PK)=0 THEN ПРЕ(90): Если к окончанию цикла выборки
E1	A890	IF BIT(8,PK)=0 THEN ПРЕ(90): дешифрируемая команда не FF00
E2	1040	COM(A) → БР : Инверсия А
E3	40B5	БР → A,N,Z; 0 → C : Пересылка результата в А
		: и признаки результата
E4	8390	GOTO ПРЕ(90) : Переход на цикл прерывания

Порядок выполнения работы

1. Получить допуск к лабораторной работе, предъявив преподавателю подготовленные материалы.

2. Занести разработанные микропрограммы циклов исполнения заданных команд в микропрограммную память базовой ЭВМ и разработанные тестовые программы в память базовой ЭВМ.

3. Выполнить в пошаговом режиме тестовые программы, проверив работоспособность синтезированных команд. Заполнить таблицу трассировки цикла исполнения для разработанных микрокоманд по форме таблицы 3.1 для одного варианта выполнения каждой микрокоманды.

Содержание отчета по работе. В дополнение к общим обязательным требованиям, отчет должен содержать:

1. Текст синтезированных микропрограмм по форме таблицы 5.1
2. Текст тестовых программ на языке Ассемблера БЭВМ (см. Приложение Д).
3. Таблицу трассировки циклов исполнения разработанных микрокоманд по форме таблицы 3.1

Контрольные вопросы:

1. Микропрограммное устройство ЭВМ, назначение, состав, принцип работы.
2. Формат горизонтальных и вертикальных микрокоманд. Для чего существуют два формата команд?
3. Исполнение горизонтальных (ОМК, УМК) и вертикальных (ОМК0,ОМК1, УМК) микрокоманд на примере заданной микрокоманды.
4. Структура и принципы функционирования АЛУ.
5. Выполнение операций суммирования и логического умножения, схема сумматора. Инверторы входов, схема инверторов входов.
6. Выполнение операций сдвигов с использованием БР.
7. Принципы построения РС, значение отдельных битов.
8. Почему возможно кодирование микрокоманд в вертикальные?
9. Как организована и выполняется микрокоманда безусловного перехода?
10. По какому принципу в вертикальных МК УС объединены в поля?
11. В каком цикле и с какой целью используется значение $(111)_2$ в младших битах ОМК1?
12. Какая операция происходит в АЛУ если биты 4,5 ОМК0 установлены в 1?
13. В какой момент происходит увеличение СчМК?
14. Переведите заданную вертикальную команду в горизонтальную и наоборот.
15. Что будет если на вентили В9 и В10 одновременно подать единицы?

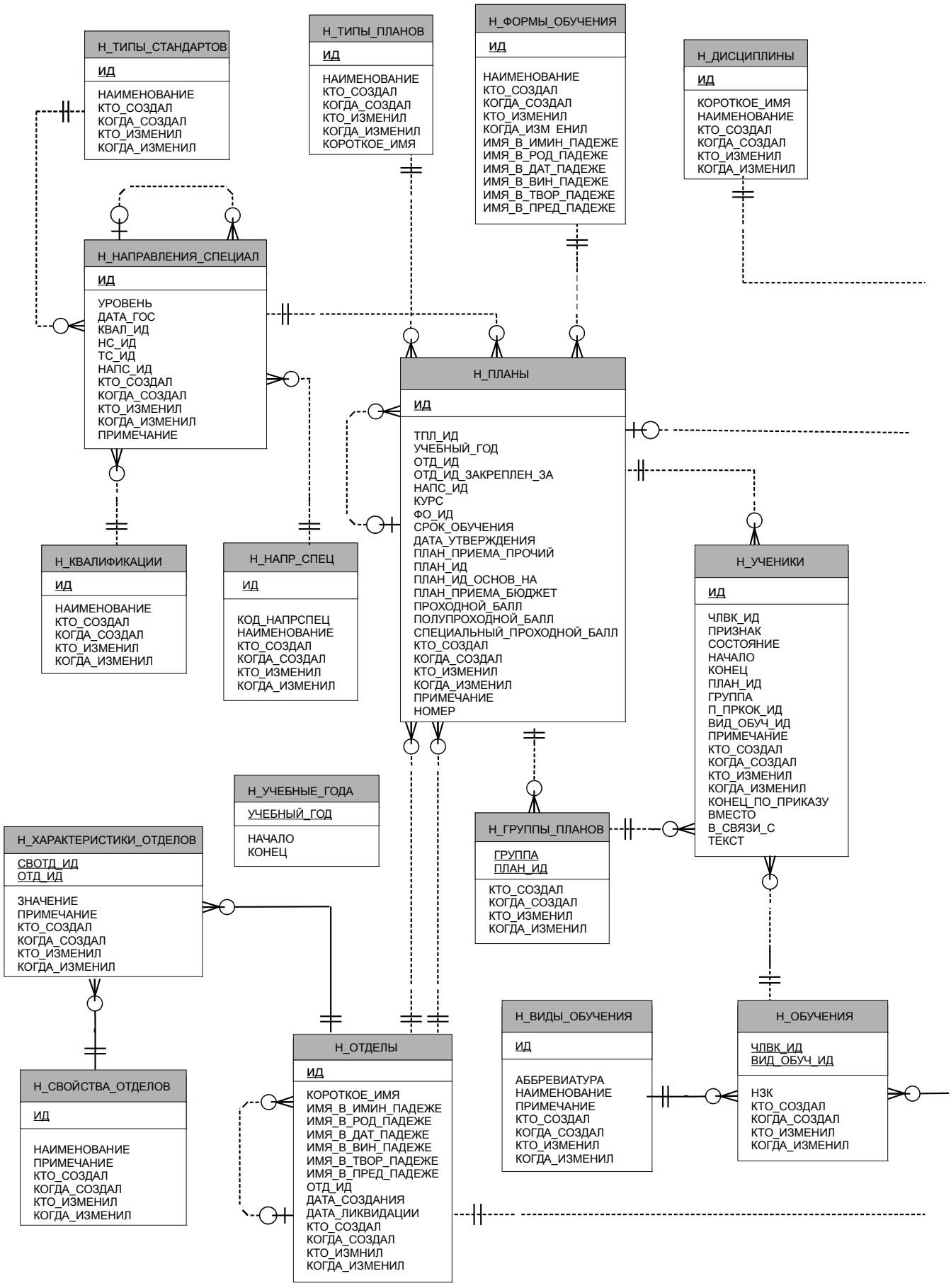
Литература

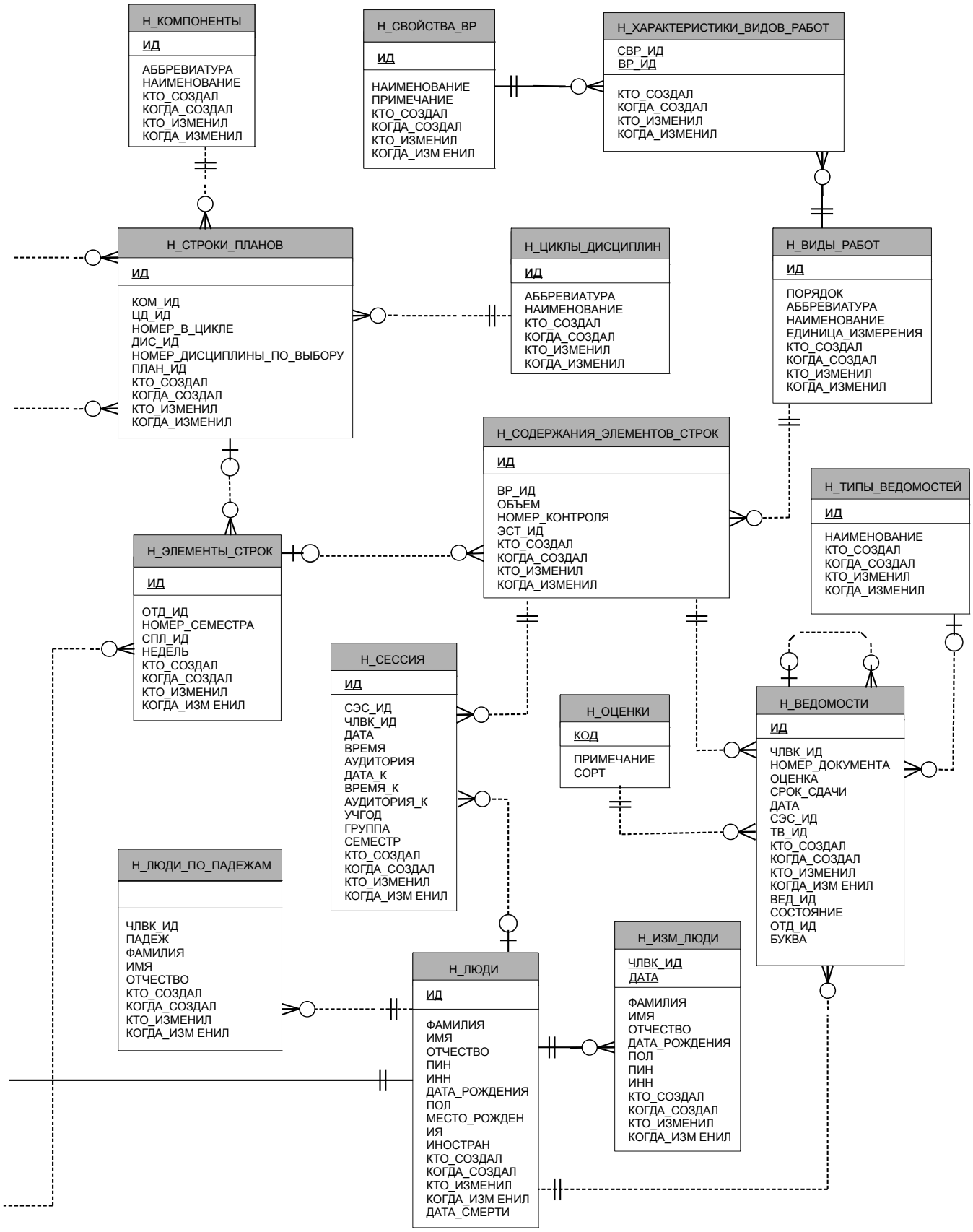
1. Введение в микроЭВМ / С. А. Майоров, В. В. Кириллов, А. А. Приблуда. — Л. Машиностроение, 1988. — 304 с. ISBN 5-217-00180-1
2. Введение в реляционные базы данных / В. В. Кириллов, Г. Ю. Громов. — СПб.: БХВ-Петербург, 2009. — 464 с.: ил. + CD-ROM — (Учебная литература для вузов) ISBN 978-5-94157-770-5
3. Кириллов В.В. Архитектура базовой ЭВМ — СПб: СпбГУ ИТМО, 2010. - 144с.

Приложение А. Краткий перечень и функциональность команд UNIX

Команда	Назначение и синтаксис
mkdir	mkdir [-m mode] [-p] dir... - Создать директорию и задать ей права (-m). Создать родительские каталоги при необходимости (-p).
echo	echo [string]... - Вывести все аргументы командны в stdout.
cat	cat [-n] [file...] [-] - Слить (con**cat**enate) файлы и вывести результат в stdout. Нумеровать строки (-n).
touch	touch [-am]... file... - Изменить время последнего доступа (-a) или модификации (-m) файла. Создать файл, если он отсутствует.
ls	ls [options] [file/dir]... Вывести список файлов/директорий.
pwd	pwd – Вывести текущую/рабочую директорию.
cd	cd [argument]- Сменить директорию на указанную в аргументе.
more	more [file...] - Интерактивная утилита постраничного вывода файлов в терминале.
cp	cp [options] SOURCE ... DEST - Копировать файлы/директории в DEST. Выполнять копирование рекурсивно (-r) интерактивно (-i).
rm	rm [options] [file/dir] - Удалить файлы/директории. Выполнять удаление рекурсивно (-r) интерактивно (-i) без подтверждения (-f).
rmdir	rmdir[dir] - Удалить непустые директории.
mv	mv [-fi] SOURCE ... DEST - Переименовать или перенести файлы/директории в DEST. См. -f и -i в rm.
find	find path... expression - Утилита поиска файлов и директорий в заданном пути по имени, атрибутам и другим параметрам.
head	head [-num] [file...] - Вывести num первых строк из файла.
tail	tail [-/+num] [-bcl] [file...] - Вывести num последних (-) или первых (+) блоков (-b); байт (-c); строк (-l) из файла.
echo	echo [string]... - Вывести аргументы в стандартный вывод.
cat	cat [-n] [file...] - Слить файлы и вывести в стандартный вывод. Нумеровать строки (-n) файлов.
wc	wc [-c | -m] [-lw] [file...] - Подсчитать количество байт (-c); букв (-m); строк (-l); слов (-w) и вывести на стандартный вывод.

Приложение Б. Инфологическая модель базы данных "Учебный процесс"




Н_КОМПОНЕНТЫ
ИД
АББРЕВИАТУРА
НАИМЕНОВАНИЕ
КТО_СОЗДАЛ
КОГДА_СОЗДАЛ
КТО_ИЗМЕНИЛ
КОГДА_ИЗМЕНИЛ
Н_СВОЙСТВА_ВР
ИД
НАИМЕНОВАНИЕ
ПРИМЕЧАНИЕ
КТО_СОЗДАЛ
КОГДА_СОЗДАЛ
КТО_ИЗМЕНИЛ
КОГДА_ИЗМ ЕНИЛ
Н_ХАРАКТЕРИСТИКИ_ВИДОВ_РАБОТ
СВР_ИД
ВР_ИД
КТО_СОЗДАЛ
КОГДА_СОЗДАЛ
КТО_ИЗМЕНИЛ
КОГДА_ИЗМЕНИЛ
Н_СТРОКИ_ПЛАНОВ
ИД
КОМ_ИД
ЦД_ИД
НОМЕР_В_ЦИКЛЕ
ДИС_ИД
НОМЕР_ДИСЦИПЛИНЫ_ПО_ВЫБОРУ
ПЛАН_ИД
КТО_СОЗДАЛ
КОГДА_СОЗДАЛ
КТО_ИЗМЕНИЛ
КОГДА_ИЗМЕНИЛ
Н_ЦИКЛЫ_ДИСЦИПЛИН
ИД
АББРЕВИАТУРА
НАИМЕНОВАНИЕ
КТО_СОЗДАЛ
КОГДА_СОЗДАЛ
КТО_ИЗМЕНИЛ
КОГДА_ИЗМЕНИЛ
Н_ВИДЫ_РАБОТ
ИД
ПОРЯДОК
АББРЕВИАТУРА
НАИМЕНОВАНИЕ
ЕДИНИЦА_ИЗМЕРЕНИЯ
КТО_СОЗДАЛ
КОГДА_СОЗДАЛ
КТО_ИЗМЕНИЛ
КОГДА_ИЗМЕНИЛ
Н_СОДЕРЖАНИЯ_ЭЛЕМЕНТОВ_СТРОК
ИД
ВР_ИД
ОБЪЕМ
НОМЕР_КОНТРОЛЯ
ЭСТ_ИД
КТО_СОЗДАЛ
КОГДА_СОЗДАЛ
КТО_ИЗМЕНИЛ
КОГДА_ИЗМЕНИЛ
Н_ТИПЫ_ВЕДОМОСТЕЙ
ИД
НАИМЕНОВАНИЕ
КТО_СОЗДАЛ
КОГДА_СОЗДАЛ
КТО_ИЗМЕНИЛ
КОГДА_ИЗМЕНИЛ
Н_ЭЛЕМЕНТЫ_СТРОК
ИД
ОТД_ИД
НОМЕР_СЕМЕСТРА
СПЛ_ИД
НЕДЕЛЬ
КТО_СОЗДАЛ
КОГДА_СОЗДАЛ
КТО_ИЗМЕНИЛ
КОГДА_ИЗМ ЕНИЛ
Н_СЕССИЯ
ИД
СЭС_ИД
ЧЛВК_ИД
ДАТА
ВРЕМЯ
АУДИТОРИЯ
ДАТА_К
ВРЕМЯ_К
АУДИТОРИЯ_К
УЧГОД
ГРУППА
СЕМЕСТР
КТО_СОЗДАЛ
КОГДА_СОЗДАЛ
КТО_ИЗМЕНИЛ
КОГДА_ИЗМ ЕНИЛ
Н_ОЦЕНКИ
КОД
ПРИМЕЧАНИЕ
СОРТ
Н_ВЕДОМОСТИ
ИД
ЧЛВК_ИД
НОМЕР_ДОКУМЕНТА
ОЦЕНКА
СРОК_СДАЧИ
ДАТА
СЭС_ИД
ТВ_ИД
КТО_СОЗДАЛ
КОГДА_СОЗДАЛ
КТО_ИЗМЕНИЛ
КОГДА_ИЗМ ЕНИЛ
ВЕД_ИД
СОСТОЯНИЕ
ОТД_ИД
БУКВА
Н_ЛЮДИ_ПО_ПАДЕЖАМ
ЧЛВК_ИД
ПАДЕЖ
ФАМИЛИЯ
ИМЯ
ОТЧЕСТВО
КТО_СОЗДАЛ
КОГДА_СОЗДАЛ
КТО_ИЗМЕНИЛ
КОГДА_ИЗМ ЕНИЛ
Н_ЛЮДИ
ИД
ФАМИЛИЯ
ИМЯ
ОТЧЕСТВО
ПИН
ИНН
ДАТА_РОЖДЕНИЯ
ПОЛ
МЕСТО_РОЖДЕН
ИЯ
ИНОСТРАН
КТО_СОЗДАЛ
КОГДА_СОЗДАЛ
КТО_ИЗМЕНИЛ
КОГДА_ИЗМ ЕНИЛ
ДАТА_СМЕРТИ
Н_ИЗМ_ЛЮДИ
ЧЛВК_ИД
ДАТА
ФАМИЛИЯ
ИМЯ
ОТЧЕСТВО
ДАТА_РОЖДЕНИЯ
ПОЛ
ПИН
ИНН
КТО_СОЗДАЛ
КОГДА_СОЗДАЛ
КТО_ИЗМЕНИЛ
КОГДА_ИЗМ ЕНИЛ

Приложение В. Состав, структура и функционирование БЭВМ

Часть 1. Базовая ЭВМ

1.1 Назначение базовой ЭВМ

Базовая ЭВМ - это простая гипотетическая машина, обладающая типичными чертами многих конкретных ЭВМ. Знание принципов построения и функционирования этой ЭВМ является хорошей базой для освоения микропроцессорных систем любых типов и моделей, поэтому она названа базовой ЭВМ. Естественно, начинать изучение ЭВМ целесообразно с подобной машины низкого уровня, что можно делать практически, используя ее модели, построенные для разных типов персональных ЭВМ. При построении базовой ЭВМ за прототип выбраны ЭВМ "Электроника 100" и "Саратов", а также схожие с ними ЭВМ типа PDP-8, однокристальный микропроцессор IM6100 и персональная ЭВМ DECmate 11.

1.2 Структура базовой ЭВМ

На рис. В.1 приведена упрощенная структура базовой ЭВМ. Это одноадресная ЭВМ аккумуляторного типа, работающая с 16-разрядными словами. В ней реализованы два вида адресации: прямая и косвенная.

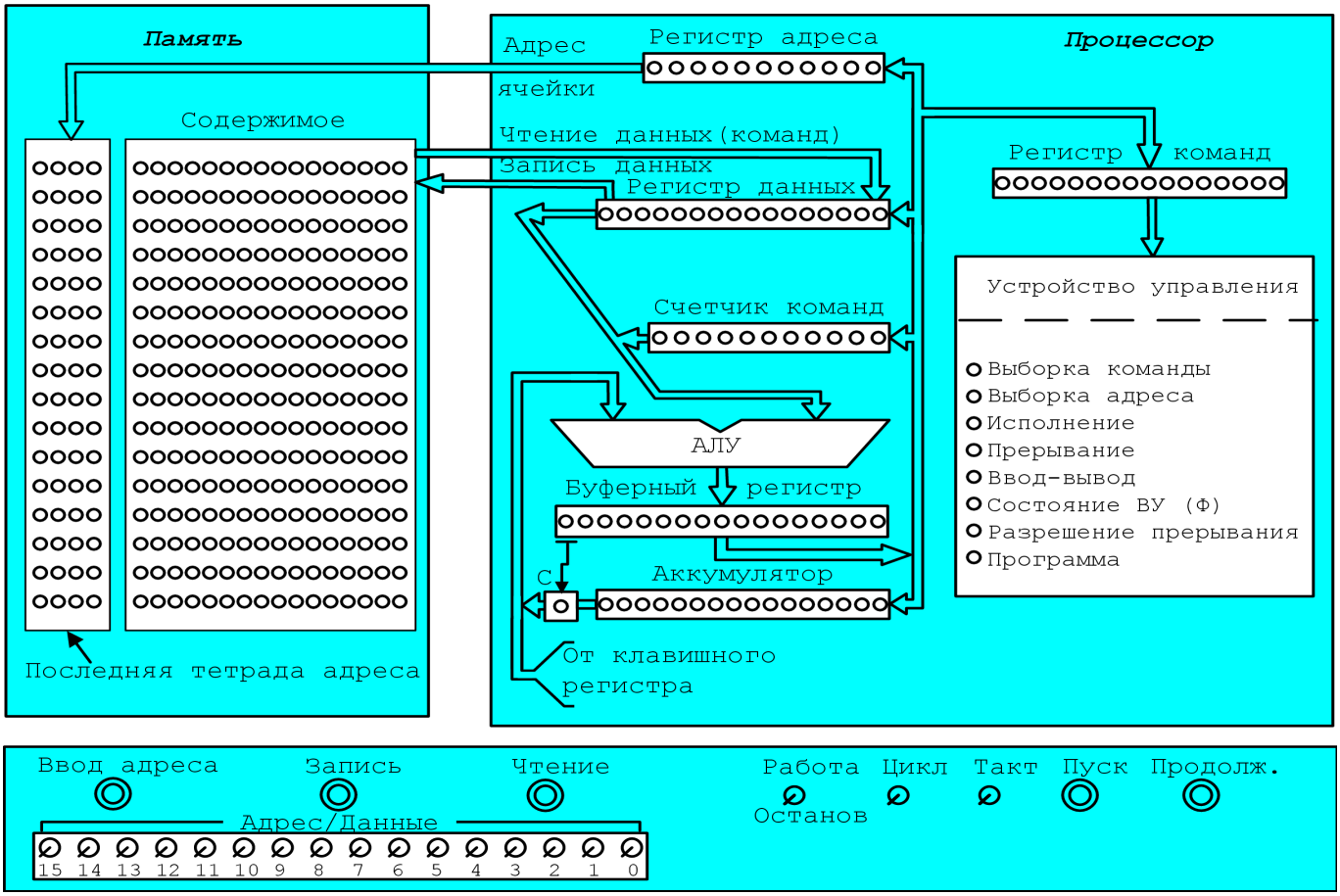


Рисунок В.1 Модель базовой ЭВМ

Рассмотрим составные части базовой ЭВМ, пока не касаясь устройств ввода-вывода (УВВ) и пульта управления (ПУ).

Память. Состоит из 2048 ячеек (16-битовых) с адресами 0, 1, ..., 2046, 2047. Восемь ячеек памяти с адресами с 008 по 00F несколько отличаются от остальных. Эти ячейки называются индексными и они предназначены для организации автоинкрементной адресации и используются в циклических программах (п. 1.5).

Процессор. Состоит из ряда регистров, арифметическо-логического устройства и устройства управления.

Арифметическо-логическое устройство (АЛУ) может выполнять арифметические операции, такие как сложение и сложение с учетом переноса, полученного в результате выполнения предыдущей операции, операции логического умножения и инвертирования. Все пересылки между регистрами в БЭВМ также выполняются через АЛУ.

Счетчик команд (СК) — регистр, который хранит адрес ячейки памяти, содержащей следующую исполняемую команду программы. Так как команды могут размещаться в любой из $2048 = 2^{11}$ ячеек памяти, то СК имеет 11 разрядов.

Регистр адреса (РА) — 11-разрядный регистр, служит для организации обращений к ячейкам памяти и содержит адрес ячейки памяти, к которой обращается процессор.

Регистр команд (РК) — 16-разрядный регистр, используемый для хранения кода выполняемой в данный момент команды.

Регистр данных (РД) — 16-разрядный регистр для временного хранения 16-разрядных слов при обмене информацией между памятью и процессором.

Буферный регистр (БР) — 17-разрядный регистр для временного хранения результатов вычислений в АЛУ. Также этот регистр используется при выполнении сдвигов.

Аккумулятор (А) — 16-разрядный регистр, являющийся одним из главных элементов процессоров аккумуляторного типа. Машина может одновременно выполнять арифметические и логические операции только с одним или двумя операндами. Один из операндов находится в аккумуляторе, а второй (если их два) - в регистре данных. Результат, в конечном счете, помещается в А.

При выполнении операций над аккумулятором процессор БЭВМ сохраняет некоторые признаки результата в специальных одноразрядных регистрах:

Флаг переноса (С) выступает в качестве продолжения аккумулятора и заполняется при переполнении А. При выполнении арифметических операций и операций сдвига в него попадает 16-й бит буферного регистра.

Флаг нуля (Z) сохраняет информацию о том, равно ли содержимое аккумулятора нулю, заполняется при выполнении операций над аккумулятором.

Флаг знака (N) сохраняет знак числа в аккумуляторе, фактически дублируя его 15-й разряд.

1.3. Система команд базовой ЭВМ

Классификация команд. БЭВМ способна исполнять точно определенный набор команд. При составлении программы пользователь ограничен этими командами. В зависимости от того, к каким блокам базовой ЭВМ обращается команда или на какие блоки она ссылается, команды можно разделить на три группы:

- адресные команды;
- безадресные команды;
- команды ввода-вывода.

Адресные команды предписывают машине производить действия с ячейкой памяти, адрес которой указан в адресной части команды.

Безадресные команды выполняют различные действия без ссылок на ячейку памяти. Например, команда `CLA` (табл. В.1) предписывает ЭВМ очистить аккумулятор (записать в А код нуля). Это команда обработки операнда, расположенного в конкретном месте, "известном" машине. Другой пример безадресной команды - команда `HLT`.

Команды ввода-вывода управляют обменом данными между процессором и внешними устройствами ЭВМ.

Полный перечень команд базовой ЭВМ приведен в таблице В.1.

Форматы команд и способы адресации. Разработчики БЭВМ выбрали три формата 16-битовых (однословных) команд с 4-битовым кодом операции (рис. В.2).

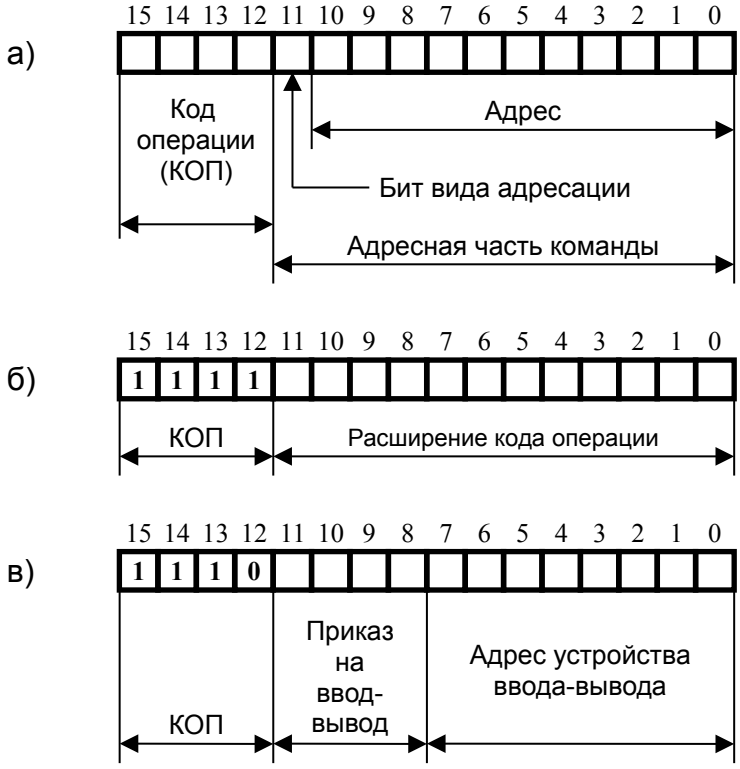


Рисунок В.2. Форматы команд: а - адресных, б - безадресных, в - команд ввода-вывода

В командах обращения к памяти на адрес отведено 11 бит. Следовательно, можно прямо адресоваться к 2^{11} = 2048 ячейкам памяти, т.е. ко всей памяти базовой ЭВМ (прямая адресация). В этом случае бит вида адресации должен содержать 0. Если же в этом же бите установлена 1, то адрес, размещенный в адресной части команды, указывает на ячейку, в которой находится адрес операнда (косвенная адресация).

Отметим, что при мнемонической записи команд указание косвенной адресации производится путем заключения адреса в скобки. Например, команда `ADD (25)` - сложить содержимое А с содержимым ячейки, адрес которой хранится в ячейке 25 (косвенная адресация).

1.4 Представление целых чисел в БЭВМ

Целые двоичные числа без знака можно использовать для представления нуля и целых положительных чисел. При размещении таких чисел в одном 16-разрядном слове они могут изменяться от $(0000\ 0000\ 0000\ 0000)_2 = (0000)_{16} = 0$ до $(1111\ 1111\ 1111\ 1111)_2 = (FFFF)_{16} = 2^{16} - 1 = 65535$. Такая запись называется прямым кодом числа.

Подобные числа (так же как и рассмотренные ниже двоичные числа со знаком) относятся к числам с фиксированной запятой, разделяющей целую и дробную части числа. В числах, используемых в базовой ЭВМ, положение запятой строго фиксировано после младшего бита слова.

Целые двоичные числа со знаком используются тогда, когда необходимо различать положительные и отрицательные числа. В современных ЭВМ для представления целых чисел со знаком используется дополнительный код, в котором старший бит формата определяет знак числа: 0 - для положительных чисел и 1 - для отрицательных чисел. При этом дополнительный код положительного числа совпадает с его прямым кодом. А для представления отрицательного числа в дополнительном коде производится инвертирование прямого кода модуля числа (получение обратного кода числа) и добавление к результату единицы. Такая же операция используется при изменении знака числа, представленного в дополнительном коде.

Таблица В.1

Система команд базовой ЭВМ

Наименование	Мнемоника	Код	Описание
Адресные команды			
Логическое умножение	AND M	1XXX	M) & (A) ➔ A
Пересылка	MOV M	3XXX	(A) ➔ M
Сложение	ADD M	4XXX	(M) + (A) ➔ A
Сложение с переносом	ADC M	5XXX	(M) + (A) + (C) ➔ A
Вычитание	SUB M	6XXX	(A) – (M) ➔ A
Переход, если перенос	BCS M	8XXX	Если (C) = 1, то M ➔ CK
Переход, если плюс	BPL M	9XXX	Если (A) >= 0, то M ➔ CK
Переход, если минус	BMI M	AXXX	Если (A) < 0, то M ➔ CK
Переход, если ноль	BEQ M	BXXX	Если (A) = 0, то M ➔ CK
Безусловный переход	BR M	CXXX	M ➔ CK
Приращение и пропуск	ISZ M	0XXX	M + 1 ➔ M, если (M) >= 0,то (CK) + 1 ➔ CK
Обращение к подпрограмме	JSR M	2XXX	(CK) ➔ M, M + 1 ➔ CK
Безадресные команды			
Очистка аккумулятора	CLA	F200	0 ➔ A
Очистка рег. переноса	CLC	F300	0 ➔ C
Инверсия аккумулятора	CMA	F400	(!A) ➔ A
Инверсия рег. переноса	CMC	F500	(!C) ➔ C
Циклический сдвиг влево на 1 разряд	ROL	F600	Содержимое A и C сдвигается влево, A(15) ➔ C, C ➔ A(0)
Циклический сдвиг вправо на 1 разряд	ROR	F700	Содержимое A и C сдвигается вправо, A(0) ➔ C, C ➔ A(15)
Инкремент аккумулятора	INC	F800	(A) + 1 ➔ A
Декремент аккумулятора	DEC	F900	(A) – 1 ➔ A
Останов	HLT	F000	
Нет операции	NOP	F100	
Разрешение прерывания	EI	FA00	
Запрещение прерывания	DI	FB00	
Команды ввода-вывода			
Очистка флага	CLF B	E0XX	0 ➔ флаг устр.
Опрос флага	TSF B	E1XX	Если (флаг устр. B) = 1, то (CK) + 1 ➔ CK
Ввод	IN B	E2XX	(B) ➔ A
Вывод	OUT B	E3XX	(A) ➔ B

Примечания:
(M), (A), (CK), (C), (B) – содержимое ячейки с адресом M, аккумулятора, счетчика команд, регистра переноса и регистра данных устройства ввода-вывода с адресом B.
XXX – адрес ячейки памяти.
XX – адрес устройства ввода-вывода.

Таким образом, для представления числа -709_{10} в дополнительном коде потребуется:

1. Записать прямой код модуля заданного числа:

```
0 000 0010 1100 0101      Модуль числа 709
```

2. Выполнить его инверсию (все его 0 заменить на 1, а все 1 - на 0):

```
1 111 1101 0011 1010      Инверсия
```

3. Прибавить единицу к полученному результату:

```
    1 111 1101 0011 1010
+
                       1
    ----------------------
    1 111 1101 0011 1011     Число -709 в дополнительном коде
```

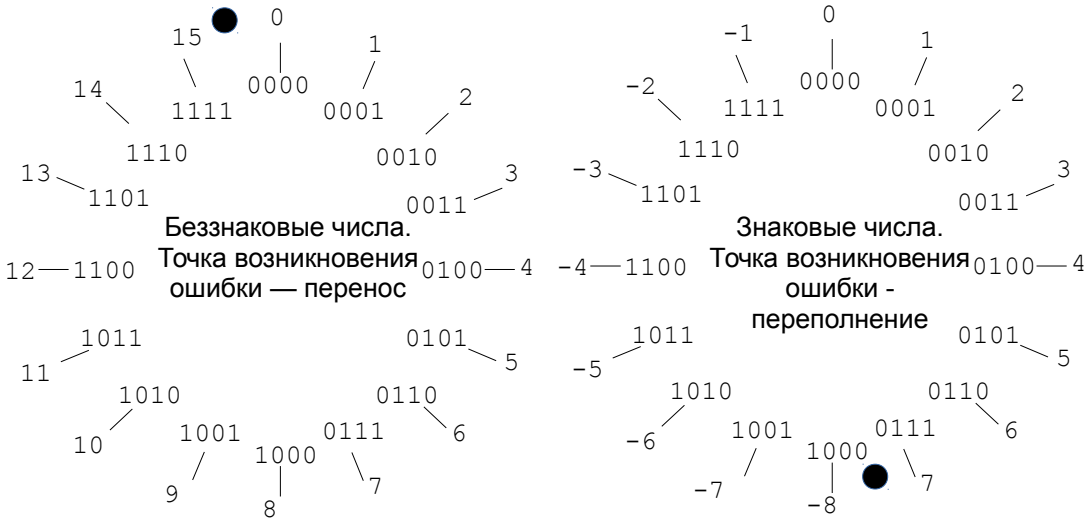
Проверим правильность вычислений путем сложения чисел 709_{10} и -709_{10} записанных в дополнительном коде:

```
    0 000 0010 1100 0101     Число 709
  +
    1 111 1101 0011 1011     Число -709
```

```
0 000 0000 0000 0000      Результат равен 0
```

Так как перенос из старшего разряда выходит за пределы разрядной сетки, то он не учитывается. Оставшаяся же 16-разрядная сумма равна нулю, что подтверждает правильность преобразования.

Использование дополнительного кода упрощает конструкцию ЭВМ, так как при сложении двух таких чисел, имеющих разные знаки, не требуется переходить к операциям вычитания меньшего (по модулю) числа из большего и присвоения результату знака большего числа. Кроме того, одной и той же схемой сумматора можно воспользоваться для выполнения операций над знаковым и беззнаковым представлением числа. Признаком выхода за границы разрядной сетки для беззнакового представления числа является перенос в старший разряд (бит C - Carry). Признаком переполнения разрядной сетки для знакового представления является бит переполнения (OVerflow). В БЭВМ, к сожалению, такой признак результата нереализован. Рассмотрим возникновение этих ситуаций на примере представления чисел в четырехразрядной сетке (рис. В.3).



0011 ← поразрядные +0011 3 переносы 0001 1 -------- 0**0**0100 4 C=0 OV=0	**11**11 +0011 3 1111 -1 -------- 1**0**0010 2 C=1 OV=0	**11**11 +1101 -3 1111 -1 -------- 1**1**1100 -4 C=1 OV=0
1101 +1101 -3 0101 +5 -------- 1**0**0010 2 C=1 OV=0	**10**00 ← биты различны +1000 -8 1111 -1 --------- 1**0**0111 (-9) ←ошибка C=1 **OV=1**	**01**11 ← биты различны +0111 7 0001 1 ------- 0**1**1000 (8) ←ошибка C=0 **OV=1**

Рисунок В.3 Возникновение переполнения и переноса

Процессор определяет переполнение по следующему правилу: если поразрядные переносы в знаковом и старшем разряде одновременно отсутствуют или присутствуют - значит переполнения нет, если присутствует только в одном – значит переполнение знаковой разрядной сетки есть.

1.5 Арифметические операции

Сложение целых двоичных чисел со знаком и без знака выполняется в базовой ЭВМ с помощью команды ADD.

Увеличение на 1 (INCREMENT) и уменьшение на 1 (DECREMENT). По команде INC к содержимому аккумулятора прибавляется единица, а по команде DEC - единица вычитается. Если при этом возникает перенос из старшего разряда А, то в регистр переноса заносится 1, в противном случае в него заносится 0.

Вычитание (X-Y) может выполняться путем изменения знака вычитаемого (CLA, ADD Y, CMA, INC) и последующего, сложения с уменьшаемым (ADD X). Однако это требует выполнения нескольких команд. Для сокращения программ и времени выполнения вычитания в базовой ЭВМ предусмотрена команда SUB Y (CLA, ADD X, SUB Y), которая реализует те же действия за меньшее время.

Умножение и деление. В базовой ЭВМ нет команд для выполнения этих действий (АЛУ не выполняет таких операций). Поэтому произведение и частное необходимо получать программным путем.

1.6 Сдвиги и логические операции

Побитовая обработка данных обеспечивается базовой ЭВМ командами логического умножения, циклических сдвигов, а также командами инвертирования и очистки аккумулятора и регистра переноса.

Команда AND M (Логическое умножение) выполняет над каждым разрядом содержимого аккумулятора и содержимым ячейки М операцию логического умножения ("И").

Результат выполнения команды для каждой пары битов операндов равен единице только тогда, когда оба бита равны единице, а в остальных случаях бит результата равен нулю, т. е., например, команда позволяет выделять или очищать определенные биты слова.

Команды ROL (циклический сдвиг влево на один разряд) и ROR (циклический сдвиг вправо на один разряд) замыкают аккумулятор и регистр переноса в кольцо и сдвигают все биты кольца на один разряд влево или вправо (рис. В.4). Сдвигами числа влево или вправо можно реализовать операции умножения или деления на два (один сдвиг), на четыре (два сдвига), на восемь (три сдвига) и т.д.

Команда CMA (инверсия аккумулятора) побитно инвертирует содержимое аккумулятора.

Команды CMC (инверсия флага переноса) и CMA (сброс флага переноса) соответственно инвертируют и сбрасывают состояние флага переноса.

	Флаг С	**Аккумулятор**
До сдвига	0	1011100000101011
После сдвига влево	1	0111000001010110
После сдвига вправо	1	0101110000010101

Рисунок В.4. Циклические сдвиги: а - влево, б - вправо

1.7 Управление вычислительным процессом

Задача управления вычислительным процессом, т.е. требуемой последовательностью выполнения команд, решается в базовой ЭВМ при помощи команд переходов (BCS, BPL, BMI, BEQ, BR), команд "Приращение и пропуск" (ISZ) и "Останов" (HLT). Все эти команды (кроме HLT) являются адресными, т.е. в них указывается адрес той ячейки памяти, из которой должна быть выбрана следующая команда программы при выполнении того или иного условия. Если же условия не выполняются, то должна исполняться команда, расположенная вслед за данной командой управления. Как и в других адресных командах, здесь можно использовать косвенную адресацию. Команды переходов не изменяют состояния аккумулятора и регистра переноса. Они могут лишь изменить содержимое счетчика команд, поместив в него адрес, определяемый адресной частью команды.

BCS M (Переход, если перенос). Переход к команде, расположенной в ячейке с адресом M, если содержимое регистра переноса равно 1.

BPL M (Переход, если плюс). Переход к команде, расположенной в ячейке с адресом M, если содержимое аккумулятора больше или равно нулю, т.е. в его старшем разряде (знаковом разряде) содержится 0.

BMI M (Переход, если минус). Переход к команде, расположенной в ячейке с адресом M, если содержимое аккумулятора меньше нуля, т.е. в его старшем (знаковом) разряде содержится 1.

BEQ M (Переход, если нуль). Переход к команде, расположенной в ячейке с адресом M, если содержимое аккумулятора равно нулю.

BR M (Переход безусловный). Переход к команде, расположенной в ячейке с адресом M.

Команды переходов широко применяются для организации циклических программ, которые используются в тех случаях, когда требуется несколько раз выполнить набор одинаковых действий с различными наборами данных. Базовая ЭВМ обладает рядом средств для упрощения циклических программ. Целесообразность введения этих средств удобнее рассмотреть на примерах.

Пример 1. Получить произведение Z = Y * 50.

Так как в системе команд базовой ЭВМ нет команды умножения, умножение на константу можно было бы выполнить комбинируя операции сдвига и сложения. Но для наглядности воспользуемся простейшим и не оптимальным способом: будем 50 раз складывать значение Y, используя программу, приведенную в табл. В.2.

Так как в этой программе аккумулятор используется не только для накопления произведения, но еще для изменения количества выполненных циклов и сравнения их со значением множителя, то промежуточные результаты Z и C пришлось сохранить в памяти ЭВМ. Очевидно, что обсуждаемую программу можно существенно упростить, при наличии такого средства учета числа выполненных циклов и проверки условия завершения циклической программы, которое не затрагивает содержимого аккумулятора. Таким средством является команда ISZ (Приращение и пропуск). При каждом выполнении команды `ISZ M`, расположенной по адресу A, к содержимому ячейки с адресом M добавляется 1 и если результат меньше нуля, то выполняется команда, следующая за `ISZ M` (команда, расположенная по адресу A+1), в противном случае эта команда *пропускается, т.е. выполняется команда, расположенная по адресу A+2. Программа с использованием команды `ISZ` приведена в табл. В.3.

Таблица В.2

*Первый вариант программы для получения Z = Y * 50*

Адрес	Содержимое		Комментарии
	Код	Мнемо-ника	
5	0078	Y	Множимое (здесь – десятичное значение 120)
6	0000	Z	Ячейка, отведенная для накопления результата
7	0032	M	Множитель 50 = $(32)_{16}$
8	0000	C	Ячейка, используемая для накопления числа выполненных циклов, - счетчик циклов
..	...	...	
10	F200	CLA	
11	4006	ADD 6	К промежуточному результату, находящемуся в ячейке 6, добавляется еще одно значение множимого Y
12	4005	ADD 5	
13	3006	MOV 6	
14	F200	CLA	
15	4008	ADD 8	Содержимое счетчика циклов увеличивается на 1, а его копия пока сохраняется в аккумуляторе
16	F800	INC	
17	3008	MOV 8	
18	6007	SUB 7	Если содержимое счетчика циклов меньше значения

19	A010	BMI 10	множителя, то выполняется переход к командам, осуществляющим новое суммирование Y с промежуточным значением Z
1A	F000	HLT	Останов. В ячейке 6 хранится искомый результат

Таблица В.3

*Второй вариант программы для получения Z = Y * 50*

Адрес	Содержимое		Комментарии
	Код	Мнемо-ника	
5	0078	Y	Множимое
6	0000	Z	Ячейка, отведенная для накопления результата.
7	FFCE	M	Отрицательное значение множителя (-50)
...	...		
10	F200	CLA	Очистка аккумулятора
11	4005	ADD 5	К содержимому аккумулятора добавляется значение Y
12	0007	ISZ 7	Содержимое M наращивается на 1 и, если оно еще
13	C011	BR 11	меньше нуля, то выполняется команда BR 11. При
14	3006	MOV 6	M = 0 команда BR 11 пропускается
15	F000	HLT	Результат 50 сложений Y заносится в ячейку 6
			Останов ЭВМ

Пример 2. Получить в ячейке 005 сумму 32 элементов массива, элементы которого размещены в ячейках памяти с 010 по 02F.

В отличие от предыдущей задачи, где многократно суммировалось содержимое одной ячейки (Y), здесь надо суммировать содержимое разных ячеек. Если бы команды БЭВМ позволяли лишь прямо адресовать ячейки памяти, то в программе решения поставленной задачи пришлось бы либо использовать 32 команды сложения (4010, 4011,...,402E, 402F), либо применять модификацию адресной части команды сложения. Последнее реализовано в программе табл. В.4.

Таблица В.4

Первый вариант программы суммирования элементов массива

Адрес	Содержимое		Комментарии
	Код	Мнемо-ника	
5	0000		Ячейка, отведенная для накопления результата
6	FFE0		Отрицательное число элементов массива
...			
10			Численные значения элементов массива
.			
.			
.			
2F			
30	F200	CLA	Промежуточный результат (ячейка 5) суммируется с содержимым элемента массива, адрес которого расположен в адресной части команды, находящейся в ячейке 32 (сначала этот адрес равен 10, а затем он увеличивается при каждом прохождении цикла на 1 командами с 34 по 37)
31	4005	ADD 5	
32	4010	ADD 10	
33	3005	MOV 5	
34	F200	CLA	Пересылка в аккумулятор команды, расположенной в ячейке 32, добавление к ее содержимому 1 и запись модифицированной команды на старое место (в ячейку 32)
35	4032	ADD 32	
36	F800	INC	
37	3032	MOV 32	
38	0006	ISZ 6	Наращивание на 1 содержимого счетчика элементов массива и переход к команде 30 пока оно < 0
39	C030	BR 30	
3A	F000	HLT	Останов ЭВМ

Модификация команд практически не используется в современных ЭВМ. Для сближения языка команд с алгоритмическими языками и для обеспечения возможности работы с программами, записанными в постоянные запоминающие устройства (откуда можно лишь читать команды), разработаны специальные средства адресации, одним из которых является косвенная адресация.

При использовании косвенной адресации нужно выбрать в памяти ЭВМ какую-либо ячейку (например, 007), записать в нее адрес первого элемента суммируемого массива (адрес 010), заменить в программе табл. В.4 команду 4010 на команду 4807 (ячейка 32) и заменить команды модификации командами вычисления текущего адреса суммируемого элемента массива. Если же вычисление текущего адреса суммируемого элемента выполнять с помощью команды `ISZ 7` (не затрагивая содержимого аккумулятора), то можно получить достаточно компактную программу, приведенную в табл. В.5.

Таблица В.5

Второй вариант программы суммирования элементов массива

Адрес	Содержимое		Комментарии
	Код	Мнемоника	
5	0000		Ячейка, отведенная для накопления результата
6	FFE0		Отрицательное число элементов массива
7	0010		Текущий адрес элемента массива
...			
10			Численные значения элементов массива
.			
.			
.			
2F			
30	F200	CLA	Очистка аккумулятора
31	4807	ADD (7)	Суммирование очередного элемента массива
32	0007	ISZ 7	Текущий адрес элемента массива наращивается на 1
33	F100	NOP	Команда “Нет операции”
34	0006	ISZ 6	Наращивание на 1 содержимого счетчика элементов массива и переход к команде 31, пока оно < 0
35	C031	BR 31	
36	3005	MOV 5	Запись результата в ячейку 5
37	F000	HLT	Останов ЭВМ

Таблица В.6

Третий вариант программы суммирования элементов массива

Адрес	Содержимое		Комментарии
	Код	Мнемоника	
5	0000		Ячейка, отведенная для накопления результата
6	FFE0		Отрицательное число элементов массива
...			
F	0010		Адрес первого элемента массива
10			Численные значения элементов массива
.			
.			
.			
2F			
30	F200	CLA	Очистка аккумулятора
31	480F	ADD (F)	Суммирование очередного элемента массива. Так как сначала в индексную ячейку F помещен адрес первого элемента массива (10), то после первого выполнения данной команды содержимое ячейки F увеличится на 1 и будет указывать на второй элемент массива, после второго выполнения команды – на третий элемент массива и т.д.
32	0006	ISZ 6	Наращивание на 1 содержимого счетчика элементов массива и переход к команде 31, пока оно < 0
33	C031	BR 31	
34	3005	MOV 5	Запись результата в ячейку 5
35	F000	HLT	Останов ЭВМ

Так как по команде `ISZ 7` (табл. В.5) производится наращивание положительной величины (адреса), то после ее выполнения счетчик команд будет указывать на команду 34 (команда по адресу 33 будет пропущена). Поэтому в ячейку 33 помещена команда "Нет операции", но можно было бы поместить даже число.

Наконец, рассмотрим еще одно средство: позволяющее упростить циклические программы базовой ЭВМ, - индексные ячейки (ячейки с адресами с 008 по 00F). Если произвести косвенное адресование какой-либо из этих ячеек, то сначала ее содержимое будет использовано в качестве адреса операнда, а затем оно автоматически увеличится на единицу. При прямом адресовании индексные ячейки не изменяются (их содержимое может измениться лишь в случае записи информации в ячейку). Указанное свойство индексных ячеек позволяет составить оптимальную программу для суммирования элементов массива (табл. В.6).

1.8 Подпрограммы

Достаточно часто встречаются ситуации, когда отдельные части программы должны выполнить одни и те же действия по обработке данных (например, вычисление тригонометрической функции). В подобных случаях повторяющиеся части программы выделяют в подпрограмму, а в соответствующие места программы заносят лишь команды обращения к этой подпрограмме. В базовой ЭВМ для этой цели используется команда `JSR` (Обращение к подпрограмме). На рис. В.5 показана часть основной программы, содержащая две команды `JSR 300`, с помощью которых осуществляется переход к выполнению команд подпрограммы.

По команде `JSR 300`, расположенной в ячейке 25, выполняется запись числа 25 + 1 = 26 (значение счетчика команд после цикла выборки команды) в ячейку с адресом 300 и запись числа 300 + 1 = 301 в счетчик команд (адрес первой команды подпрограммы). Таким образом осуществляется переход к выполнению команд подпрограммы. Далее начинается процесс выполнения команд подпрограммы, который завершается командой `BR (300)`, расположенной в ячейке 326. Эта команда безусловного перехода с косвенной адресацией предписывает ЭВМ выполнить переход к команде, расположенной по адресу, сохраненному в ячейке 300. Так как в эту ячейку ранее было записано число 26, то будет исполняться команда, находящаяся в ячейке 26, т.е. следующая за обращением к подпрограмме. Аналогично выполняется команда `JSR 300`, расположенная в ячейке 72 (после выполнения команд подпрограммы будет выполнен переход к ячейке 73).

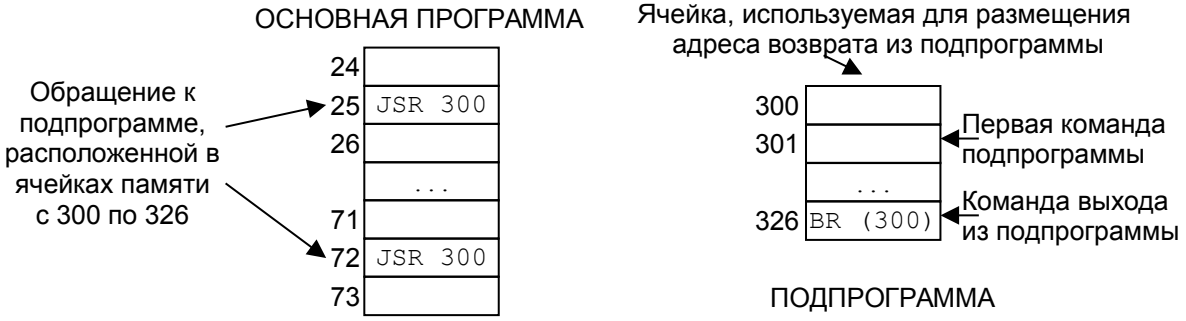


Рисунок В.5. Обращение к подпрограмме и возврат из нее

Таким образом, при оформлении подпрограммы перед ее первой командой следует разместить ячейку, в которую будет записываться адрес возврата из подпрограммы. В команде обращения к подпрограмме указывается адрес именно этой ячейки, например, адрес М в команде `JSR M`. Для возврата из подпрограммы можно использовать любую команду перехода, косвенно адресующуюся к ячейке М, например, `BR(M)`. По ней осуществляется переход к команде, адрес которой сохраняется в первой ячейке подпрограммы.

1.9 Выполнение машинных команд

В процессе исполнения команд устройство управления БЭВМ производит анализ и пересылку команд, отдельных ее частей (кода операции, признака адресации и адреса) или операнда из одного регистра ЭВМ в другой ее регистр, АЛУ, память или устройство ввода-вывода. Эти действия (микрооперации) протекают в

определенной временной последовательности и скоординированы между собой. Для обеспечения такой координации в ЭВМ используется генератор тактовых импульсов.

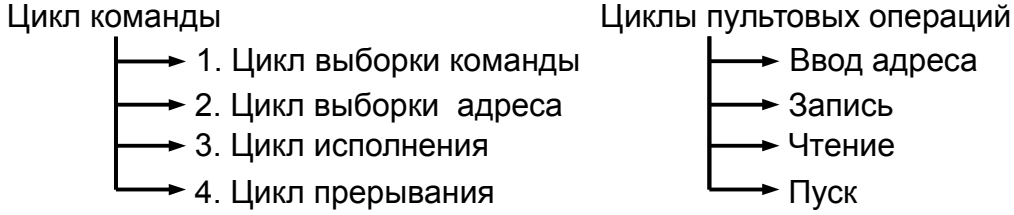


Рисунок В.6 Циклы устройства управления

Устройство управления хранит в себе последовательность действий для исполнения команд и выполнения пультовых операций, называемых циклами.

Цикл команды. Для реализации одной команды требуется выполнить определенное количество действий, каждое из которых инициируется одним тактовым импульсом. Общее число тактовых импульсов, требуемых для выполнения команды, определяет время ее выполнения, называемое *циклом команды*. Цикл команды включает несколько *машинных циклов*: выборки команды, выборки адреса, исполнения и прерывания. Основные действия, выполняемые ЭВМ во время каждого из машинных циклов, проиллюстрированы и описаны ниже.

Выборка команд. В данном машинном цикле выполняется чтение команды из памяти и ее частичное декодирование.

1. Содержимое счетчика команд через АЛУ записывается в буферный регистр.
2. Содержимое буферного регистра записывается в регистр адреса (рис. В.7).
3. Содержимое ячейки памяти, на которую указывает регистр адреса, читается из памяти в регистр данных (рис. В.8), а содержимое счетчика команд попадает в АЛУ, где увеличивается на 1, результат записывается в буферный регистр.
4. Содержимое буферного регистра записывается в счетчик команд.
5. Содержимое регистра данных через АЛУ пересылается в буферный регистр.
6. Содержимое буферного регистра записывается в регистр команд.
7. Происходит частичное декодирование кода команды в регистре команд для выявления типа команды (адресная, безадресная или ввода-вывода) и вида адресации для адресных команд (рис. В.9)

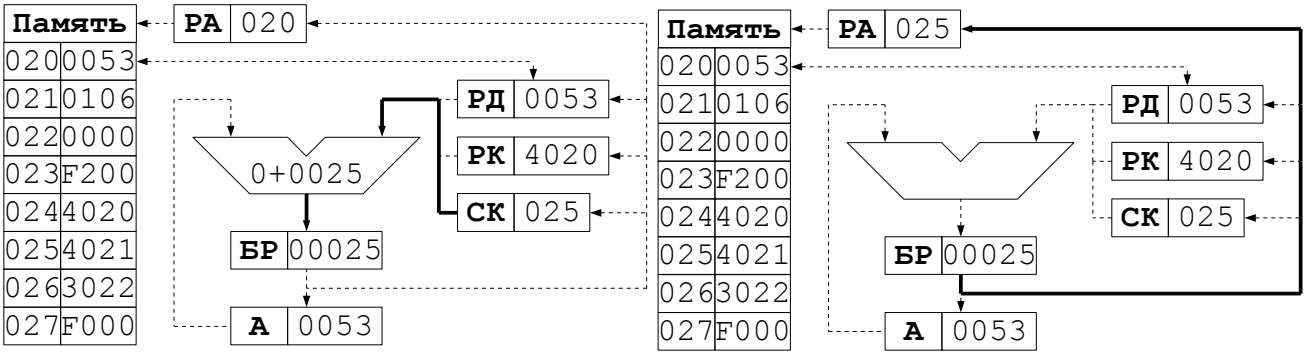


Рисунок В.7 Передача счетчика команд в регистр адреса (такты 1 и 2 выборки команды)

Рисунок В.8 Выборка команды с одновременным увеличением СК (такты 3 и 4 выборки команды)

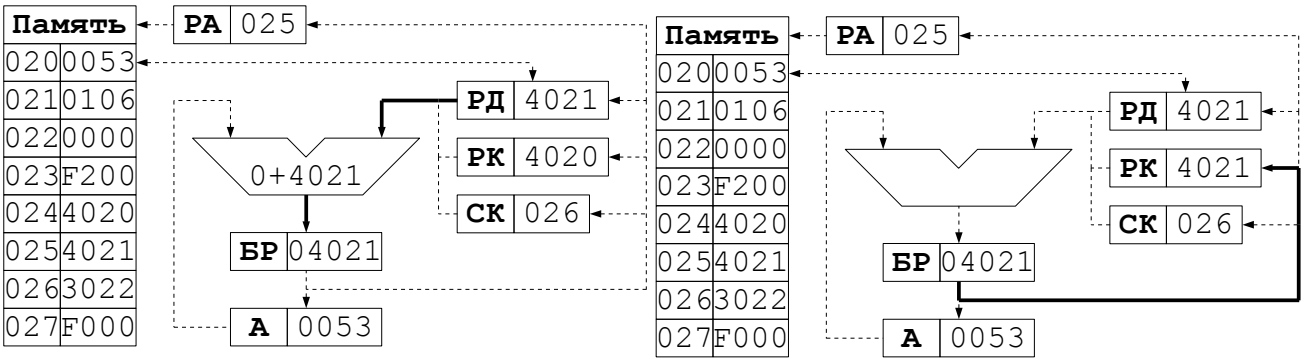


Рисунок В.9 Запись кода команды в РК (такты 5,6 и 7 выборки команды)

Выборка адреса. Этот машинный цикл следует за циклом выборки команды *только для адресных команд с косвенной адресацией*, т.е. команд, бит вида адресации которых равен 1. Цикл используется для чтения из памяти адреса операнда, результата или перехода и состоит из следующих шагов:

1. Содержимое регистра данных через АЛУ пересылается в буферный регистр.
2. Адрес (младшие 11 бит) содержимого буферного регистра записываются в регистр адреса.
3. Содержимое ячейки памяти, указываемой регистром адреса, читается в регистр данных. Теперь в этом регистре находится либо адрес операнда, либо адрес результата, либо адрес перехода, который будет использоваться в цикле исполнения команды. Если косвенно адресуется одна из индексных ячеек (адреса 8...F), то цикл выборки адреса операнда (результата) продолжается, иначе машинный цикл завершается.
4. Содержимое регистра данных попадает в АЛУ, где увеличивается на 1, результат записывается в буферный регистр.
5. Содержимое буферного регистра записывается в регистр данных.
6. Измененное содержимое регистра данных пересылается в ячейку памяти по адресу, указанному регистром адреса.
7. Содержимое регистра передается в АЛУ, где уменьшается на 1 и попадает в буферный регистр.
8. Содержимое буферного регистра записывается в регистр данных.

После последней операции в регистре данных восстанавливается значение адреса, находившегося в индексной ячейке до выполнения шага 3. Содержимое же индексной ячейки увеличилось на 1 и при следующем обращении к ней будет выбран новый адрес.

Исполнение. Последовательность действий, выполняемых в этом машинном цикле, определяется самой выполняемой командой.

1. Для команд, при выполнении которых требуется выборка операнда из памяти ЭВМ (`AND`, `ADD`, `ADC`, `SUB`, `ISZ`), цикл исполнения используется для чтения операнда в регистр данных и выполнения операции, указываемой кодом операции команды. Пример цикла исполнения команды `ADD 21` приведен на рис. В.10.
2. По команде пересылки (`MOV`) в этом машинном цикле производится запись содержимого аккумулятора в ячейку памяти с адресом, расположенным в регистре данных. Для этого содержимое регистра данных пересылается в регистр адреса, а содержимое аккумулятора - в регистр данных и далее в ячейку памяти, указываемую регистром адреса.

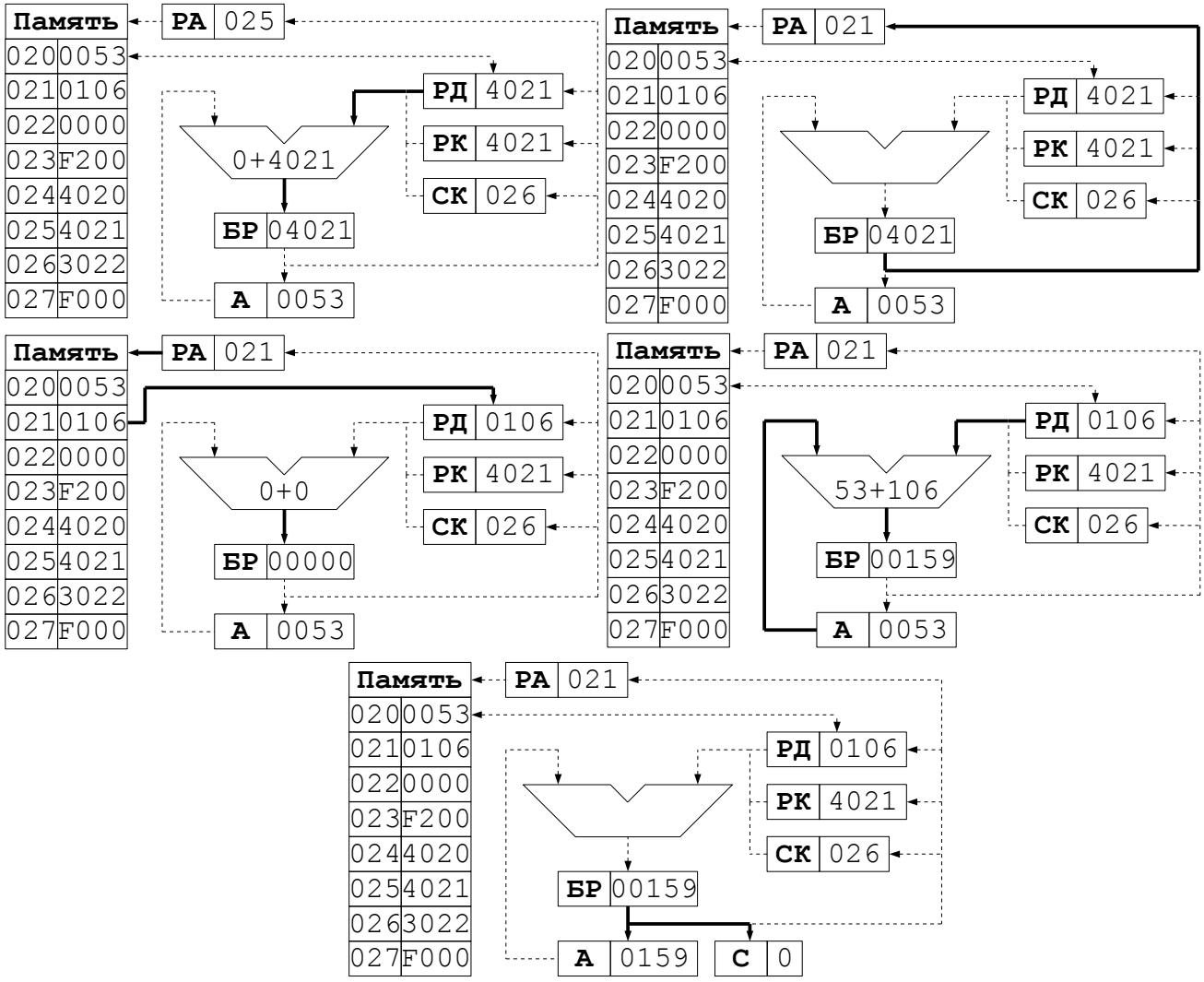


Рисунок В.10 Цикл "Исполнение" команды ADD 21

3. При исполнении команд переходов (BCS, BPL, BMI, BEQ) производится проверка соответствующего условия (1 - в регистре переноса, 0 — во флаге N и т.п.) и пересылка адреса из регистра данных в счетчик команд при выполнении этого условия. Иначе будет выбрана команда, расположенная вслед за командой перехода. При исполнении команды безусловного перехода (BR) пересылка адреса перехода в счетчик команд выполняется без какой-либо проверки.
4. Для команды обращения к подпрограмме (JSR) во время этого машинного цикла осуществляется пересылка содержимого счетчика команд в ячейку памяти, адрес которой содержится в регистре данных, и занесение в счетчик команд увеличенного на единицу содержимого регистра данных.
   Машинный цикл прерывания будет рассмотрен в разделе 2.4.

Цикл пультовых операций выключает в себя выполнение соответствующих действий для выполнения пультовых операций: ввод адреса, запись, чтение, пуск.

Пультовая операция "Ввод адреса" записывает содержимое клавишного регистра в счетчик команд.

Пультовая операция "Запись" записывает содержимое клавишного регистра в ячейку памяти, адрес которой указан в счетчике команд, после чего увеличивает на единицу содержимое счетчика команд, т.е. переходит к следующей ячейке.

Пультовая операция "Чтение" считывает в регистр данных содержимое ячейки памяти, адрес которой указан в счетчике команд, после чего увеличивает на

единицу содержимое счетчика команд.

Пультовая операция "Пуск" сбрасывает содержимое аккумулятора и флага переноса, сбрасывает готовность всех внешних устройств, запрещает прерывания и, если установлен режим Работа, переходит к выполнению команды, адрес которой указан в счетчике команд.

Часть 2. Организация ввода-вывода в базовой ЭВМ

Обмен информацией с внешним устройством состоит из *инициации обмена,* где осуществляются предварительные действия по подготовке к вводу или выводу данных (установка соединения, ожидание готовности и пр.) и собственно *обмена данными* (их передачей или приемом).

Если и инициацией и обменом занимается центральный процессор, то такой обмен называется *программно-управляемым.* Программно-управляемый обмен по способу инициации разделяется на *синхронный,* когда обмен начинается в заранее известный промежуток времени (например, каждую минуту) и *асинхронный,* когда программе неизвестно время начала обмена данными и она вынуждена периодически проверять возможность обмена (например, готовность внешнего устройства).

Чтобы исключить периодическую проверку готовности, устройства могут сами инициировать обмен по специальному аппаратному сигналу, который называется *запрос прерывания*, а соответствующий обмен — *управляемый прерываниями ввод-вывод*. При таком способе внешнее устройство сигнализирует процессору о необходимости начать обмен, процессор приостанавливает (*прерывает*) текущую программу, осуществляет ввод-вывод с помощью *программы обработки прерывания,* а затем продолжает выполнять основную программу.

Ввод-вывод с использованием прямого доступа к памяти (ПДП, в английской литературе DMA) организует и инициацию и обмен данными при помощи контроллеров ПДП. Такие контроллеры передают данные непосредственно в память ЭВМ, при этом центральный процессор в обмене данными не участвует.

Обмен данными (прием и передача) может также быть организован *синхронно*, когда наличие данных на шине подтверждается специальным сигналом синхронизации с постоянной частотой, и *асинхронно*, с использованием сигналов готовности и/или подтверждения приема-передачи данных.

Задачу инициации и обмена данными в ЭВМ осуществляют специальные программы (такие программы еще называют драйверами), которые совместно с аппаратурой ЭВМ организуют и контролируют процесс ввода-вывода.

2.1 Устройства ввода-вывода базовой ЭВМ

Модель базовой ЭВМ с контроллерами устройств ввода-вывода представлена на рис В.11. В базовой ЭВМ используются простейшие внешние устройства (ВУ): устройство вывода (ВУ-1), устройства ввода ВУ-2 и устройство ввода-вывода ВУ-3. В модели устройства ввода-вывода представлены 8-разрядными регистрами данных (РД ВУ). Через регистры данных ВУ-2 и ВУ-3 информация может быть введена в базовую ЭВМ, а в регистры данных ВУ-1 и ВУ-3 принята из базовой ЭВМ.

Кроме того, в последних версиях БЭВМ реализован таймер (устройство ВУ-0), которое вызывает прерывание через заданное в его РД число секунд.

Между ВУ и процессором включены простейшие контроллеры, каждый из которых содержит:

- *дешифратор адреса*, позволяющий выделить обращение к данному ВУ среди всех обращений к устройствам ввода-вывода, подключенных к процессору;
- *дешифратор приказов*, декодирующий приказы от процессора на выполнение тех или иных операций;
- *регистр данных*, через который происходит обмен данными между процессором и внешним устройством;

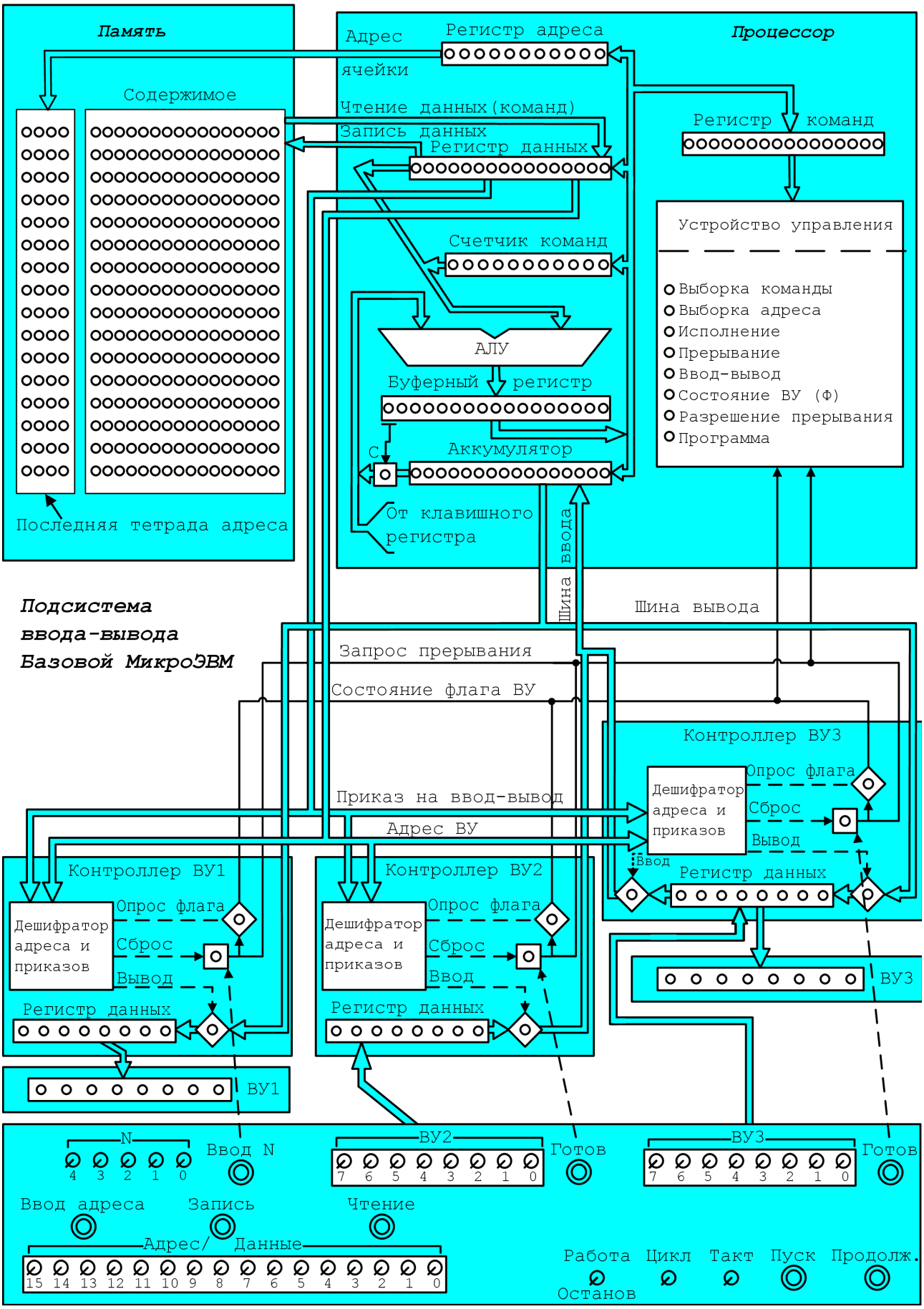


Рисунок В.11 Модель базовой ЭВМ с контроллерами устройств ввода-вывода

- *регистр состояния*, в котором хранится информация о готовности ВУ к обмену данными с процессором. В контроллерах простейших ВУ используются однобитовые регистры готовности, которые часто называют *флагом*.

Контроллеры ВУ связаны с процессором:

- *шинами ввода и вывода информации*, по которым происходит передача данных в или из процессора;
- *шиной адреса*, по которой передается адрес внешнего устройства;
- *шиной приказа на ввод-вывод*, по которой передается код операции ввода-вывода;
- *шиной состояния флага ВУ*, по которой передается значение однобитового признака готовности опрашиваемого ВУ;
- *шиной запроса прерывания*, по которой выставляется требование прерывания от внешнего устройства.

2.2 Команды ввода-вывода

Формат команд ввода-вывода приведен на рис. В.2 (в). Код операции 1110_2 (E_{16}) служит для отличия этих команд от других команд ЭВМ. Между собой они отличаются кодом приказа: пересылка данных (`IN B` - ввод и `OUT B` - вывод), *проверка готовности ВУ* (`TSF B`) и *сброс состояния готовности* (`CLF B`), где B - адрес ВУ. Адрес позволяет связать процессор с одним из подключенных к нему ВУ (их может быть до 2^8=256).

Флаг - однобитовый регистр готовности ВУ, устанавливаемый в единичное состояние, когда ВУ готово к обмену информацией и процессор имеет право на обмен данными с РДВУ.

Команда CLF B (E0xx, где xx - две последние 16-ричные цифры адреса ВУ) служит для сброса в нуль флага готовности ВУ с адресом B.

Команда TSF B (E1xx) служит для проверки готовности к обмену ВУ с адресом B. Если флажок этого ВУ сброшен (ВУ не готово к обмену), то выполняется команда, расположенная вслед за `TSF B`. В противном случае следующая команда *пропускается* и выполняется команда, расположенная через одну за `TSF B`.

Команда IN B (E2xx) служит для пересылки содержимого регистра данных контроллера ВУ с адресом B в восемь младших разрядов аккумулятора.

Команда OUT B (E3xx) служит для пересылки содержимого восьми младших разрядов аккумулятора в регистр данных контроллера ВУ с адресов B.

2.3 Программно-управляемый асинхронный обмен.

При использовании программно-управляемого асинхронного обмена должна быть составлена программа, обеспечивающая пересылку данных из памяти ЭВМ в аккумулятор и далее в регистр памяти контроллера ВУ (вывод данных) или из регистра данных контроллера ВУ в аккумулятор и затем в память ЭВМ (ввод данных). Программа такого обмена строится так: сначала проверяется готовность ВУ к обмену и если оно готово, то дается команда на обмен. ВУ сообщает о готовности установкой флага.

Пример 3 С помощью ВУ-2 записать в ячейку 006 коды символов слова "ДА" в кодировке KOI8-R. Программа для выполнения этого задания приведена в табл. В.7.

Таблица В.7

Ввод данных с использованием ВУ-2

Адрес	Содержимое		Комментарии
	Код	Мнемоника	
05	FFF8		Константа -8, используемая для сдвига
06	0000		Ячейка для записи слова "ДА"
...	...		
20	F200	CLA	Очистка аккумулятора

21	E102	TSF 2	Опрос флага контроллера ВУ-2 и повторение этой операции пока ВУ-2 не готово к обмену (флаг=0)
22	C021	BR 21	
23	E202	IN 2	Это действие выполняется лишь после готовности ВУ-2, т.е. когда при выполнении TSF 2 выяснятся, что флаг=1 и пропускается BR 20, По команде IN 2 содержимое регистра данных контроллера ВУ-2 пересылается в восемь младших разрядов аккумуляторов.
24	E002	CLF 2	Сброс готовности ВУ-2 (очистка флага ВУ-2)
25	F600	ROL	Код первого введенного символа сдвигается на восемь разрядов влево и освобождается место для ввода следующего символа.
26	0005	ISZ 5	
27	C025	BR 25	
28	E102	TSF 2	Повторное ожидание готовности ВУ-2
29	C028	BR 28	
2A	E202	IN 2	Ввод кода символа(если подан сигнал готовности ВУ-2)
2B	E002	CLF 2	Сброс готовности ВУ-2
2C	3006	MOV 6	Пересылка введенного значения в ячейку 006
2D	F000	HLT	Останов ЭВМ

Две первые команды этой программы "заставляют" ЭВМ ожидать его готовности ВУ-2 к выдаче данных. Поэтому необходимо ввести код символа "Д" в регистр данных ВУ-2. После сброса готовности ВУ-2 (команда `CLF 2`), которая подтверждает, что данные из регистра данных контроллера ВУ-2 переписаны в аккумулятор можно приступить к набору кода символа "А". В процессе набора этого кода ЭВМ занята сдвигом кода символа "Д" в старшие разряды аккумулятора, чтобы подготовиться к приему символа "А", и ожиданием поступления нового сигнала готовности ВУ-2 к выдаче информации. Так как ЭВМ выполняет эти операции значительно быстрее, чем человек, набирающий код нового символа. Теперь в аккумулятор перепишется все слово "ДА", затем оно перепишется в ячейку 006 и выполнение программы прекратиться.

Легко заметить, что при данной реализации асинхронного обмена ЭВМ тратит время на ожидание момента готовности циклически опрашивая флаг (spin) и не может выполнять никакой другой работы. Однако такое зацикливание не является обязательным и опрос флага может осуществляться периодически при выполнении основной программы.

2.4 Управляемый по прерыванию программы ввод-вывод.

Этот вид обмена отличается от асинхронного тем, что сигнал готовности ВУ к обмену анализируется не программным, а аппаратным путем. ЭВМ может выполнять любую не связанную с обменом программу (будем называть ее основной), а когда из ВУ по линии "Запрос прерывания" (рис. В.11) поступит сигнал готовности ВУ к приему или выдаче информации, прервать (приостановить) выполнение этой программы на время выполнения программы обмена данными. Все эти действия осуществляются с помощью входящего в состав устройства управления базовой ЭВМ контроллера прерываний, реализованного логической схемой "И".

Команды EI (Разрешение прерывания) и DI (Запрещение прерывания) переводят контроллер прерываний в одно из двух состояний, в которых процессор соответственно реагирует на сигналы готовности ВУ передаваемые по линии "Запрос прерывания", или игнорирует их.

Если прерывания разрешены, то после выполнения всех команд кроме `EI`, `DI`, `HLT` выполняются следующие действия:

Шаг 1. По завершению цикла исполнения текущей команды происходит переход на цикл прерывания. Если в этот момент на линии "Запрос прерывания" нет сигнала о прерывании от какого-либо ВУ, то происходит переход к следующей команде. При наличии запроса контроллер прерываний переходит в состояние

запрещения прерывания, в ячейку памяти с адресом 000 заносится содержимое СК (адрес следующей команды основной программы), и управление передается команде расположенной в ячейке 001, с которой начинается подпрограмма обработки прерывания.

Шаг 2. Подпрограмма обработки прерывания запоминает в памяти содержимое аккумулятора и регистра переноса. Для этого требуется минимум три команды: пересылка содержимого аккумулятора в специально отведенную ячейку SAVED_A, циклический сдвиг содержимого аккумулятора влево (для того, чтобы содержимое регистра переноса попало в аккумулятор) и запись этого содержимого в другую ячейку SAVED_C. Таким образом, необходимый минимум информации о прерванной программе сохраняется - в ячейке 000 хранится адрес продолжения прерванной программы, а в ячейках SAVED_A и SAVED_C хранится содержимое двух других основных регистров А и С.

Шаг 3. Производится поиск источника прерывания. Для этого в наиболее целесообразной последовательности опрашиваются флаги ВУ (команда `TSF`). При обнаружении ВУ с установленным флагом (флаг = 1) выполняется переход к тому участку подпрограммы, в котором описаны действия по обмену данными с этим ВУ.

Шаг 4. Выполняется передача данных и их предварительная обработка, если это необходимо.

Шаг 5. Восстанавливается содержимое регистра переноса и аккумулятора. Для этого требуется шесть команд: очистка аккумулятора, вызов содержимого ячейки SAVED_C в очищенный аккумулятор, циклический сдвиг вправо для восстановления содержимого регистра переноса, очистка и инверсия аккумулятора и его логическое умножение на содержимое ячейки SAVED_A. Использование операции логического умножения позволяет восстановить содержимое аккумулятора без изменения регистра С.

Шаг 6. Контроллер прерываний вновь переводится в состояние разрешение прерывания командой `EI` и осуществляется возврат к выполнению прерванной программы, т.е. к команде, адрес которой хранится в ячейке 000 командой `BR (000)`. Здесь следует отметить, что команда `BR (000)` должна располагаться непосредственно за командой `EI`. В противном случае, если к моменту окончания выполнения команды `EI` появится запрос прерывания от внешнего устройства, то будет переписан адрес возврата в основную программу новым значением.

Пример 4. Составить программу, которая периодически (с периодом в три цикла команды) наращивает на 1 содержимое аккумулятора. Восемь младших разрядов аккумулятора должны выводиться на ВУ-1 по его запросу, а по запросу ВУ-3 код, записанный в регистр данных контроллера ВУ-3, должен помещаться в ячейку 25.

Таблица В.8

Ввод данных с использованием прерываний. Ассемблерный листинг.

```
ORG       000
RET:      WORD     ?      ; Ячейка для хранения адреса возврата
          BR INT          ; Переход к подпрограмме обработки прерываний

ORG       020             ; Основная программа
BEGIN:    EI              ; Установка состояния разрешения прерывания
          CLA             ; Очистка аккумулятора
LOOP:     INC             ; Цикл для наращивания содержимого аккумулятора
          NOP
          BR LOOP
IO3:      WORD     ?      ; Ячейка для хранения кодов, поступающих с ВУ-3

ORG       030
INT:      MOV      SAVED_A      ; Сохранение содержимого аккумулятора и С
          ROL
          MOV      SAVED_C
          TSF      3      ; Опрос флага ВУ-3
          BR CHECK1       ; Если он сброшен, то переход к опросу флага ВУ-1
```

```
          BR READY3      ; В противном случае переход на ввод данных из ВУ-3
CHECK1:   TSF     1      ; Опрос флага ВУ-1
          BR READY2      ; Если он сброшен, то переход к сбросу флага ВУ-2
          BR READY1      ; В противном случае переход на вывод данных в ВУ-1
READY3:   CLA            ; Ввод данных из ВУ-3
          IN 3
          CLF     3      ; Сброс флага ВУ-3
          MOV     IO3    ; Пересылка их в ячейку 25,
          BR RESTORE     ; Переход к восстановлению содержимого регистров
READY1:   CLA            ; Пересылка в аккумулятор содержимого ячейки SAVED_A
          ADD     SAVED_A
          OUT     1      ; Вывод на ВУ-1 восьми младших разрядов аккумулятора
          CLF     1      ; Сброс флага ВУ-1
          BR RESTORE     ; Переход к восстановлению содержимого регистров
READY2:   CLF     2
RESTORE:  CLA            ; Восстановление содержимого C и аккумулятора
          ADD     SAVED_C
          ROR
          CLA
          CMA
          AND     SAVED_A
          EI      ; Возобновление состояния разрешения прерывания
          BR (RET) ; Возврат из подпрограммы обработки прерывания

SAVED_A:  WORD    ?
SAVED_C:  WORD    ?
```

Если эту программу занести в память базовой ЭВМ, установить в СК пусковой адрес 20 и нажать кнопку ПУСК, то начнет выполняться бесконечный цикл наращивания содержимого аккумулятора. Когда же на пульте управления (рис В.11) будет нажата любая из трех кнопок ("Готов" ВУ-1, ВУ-2 или ВУ-3), то будет выполнен переход к подпрограмме обработки прерываний.

Часть 3. Микропрограммное устройство управления

3.1. Микропрограммное управление вентильными схемами.

Процесс выборки, дешифрации и исполнения команд ЭВМ состоит из последовательности элементарных операций (например, пересылка содержимого одного регистра в другой регистр или проверка определенного бита в каком-либо регистре). Для выполнения таких микроопераций, как правило, достаточно подать открывающий сигнал на одну или несколько вентильных схем, связывающих между собой два регистра, регистр и АЛУ и (или) перестраивающих АЛУ на выполнение заданной операции (сложения, логического умножения и т.п.). Требуемая последовательность сигналов на вентильные схемы ЭВМ вырабатывается ее устройством управления, связанным с тактовым генератором.

Микропрограммное устройство управления (МПУ) базовой ЭВМ - это, в свою очередь, очень простая ЭВМ, для которой регистры и вентильные схемы процессора являются как бы устройствами ввода-вывода (рис. В.12).

Программа работы такой ЭВМ называется микропрограммой, а ее команды, содержащие информацию об элементарных действиях, выполняемых в течение одного рабочего такта ЭВМ, - микрокомандами. В одном из вариантов реализации МПУ используется всего два типа микрокоманд - операционная и управляющая (рис. В.13).

Микропрограмма обычно хранится в постоянном запоминающем устройстве - памяти микрокоманд, но в эмуляторе БЭВМ реализована возможность изменения микропрограммы. В каждом такте работы ЭВМ из этой памяти в регистр микрокоманд (РМК) пересылается очередная микрокоманда, т.е. микрокоманда, на которую указывает счетчик микрокоманд (СчМК), одновременно выполняющий функции регистра адреса микрокоманд.

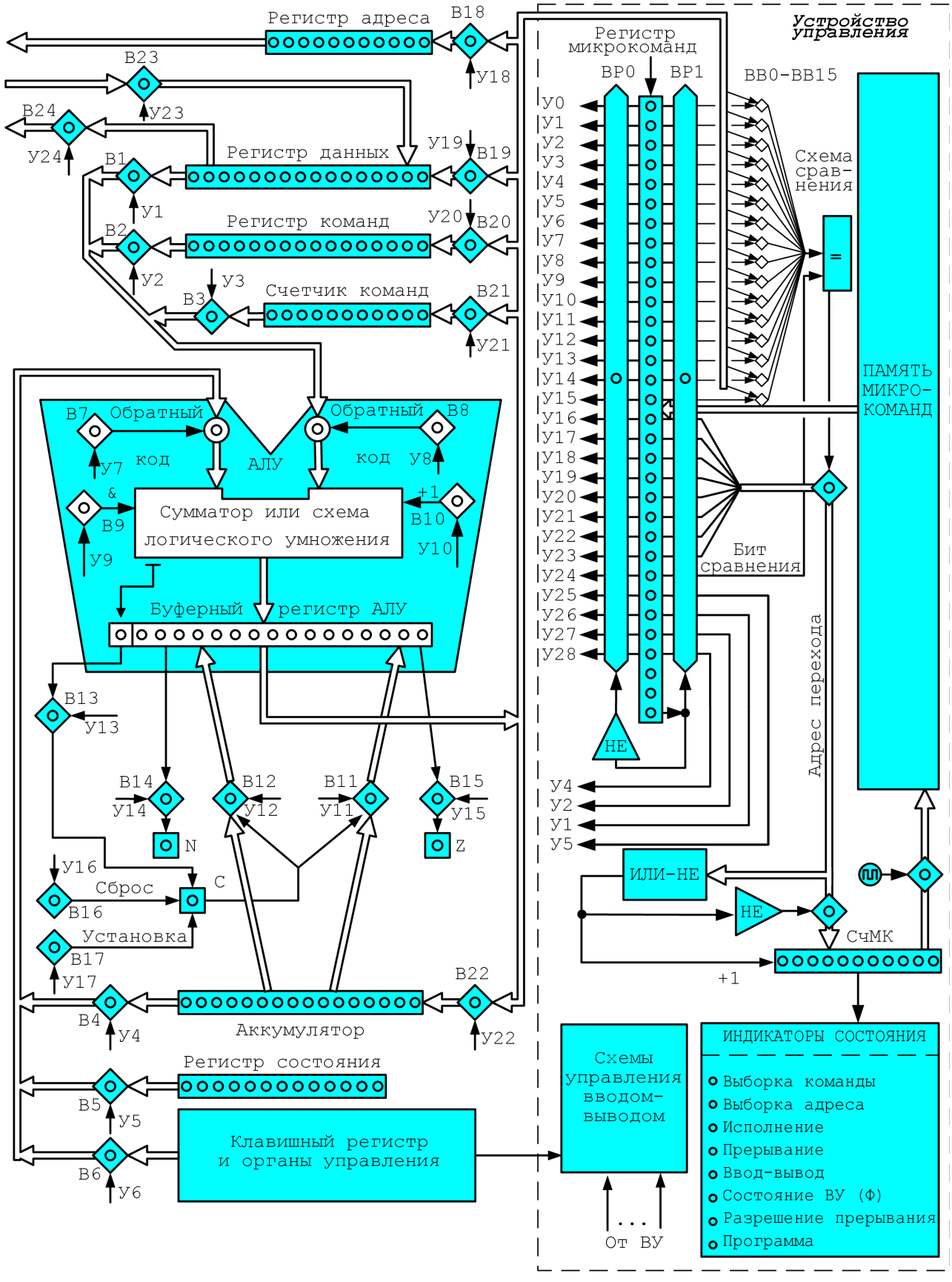


Рисунок В.12 Модель базовой ЭВМ с микропрограммным устройством управления

Если из памяти микрокоманд выбрана операционная микрокоманда, то в 31-ый бит РМК записывается 0 (код операции ОМК). Этот сигнал через инвертор НЕ открывает вентильную схему ВРО и обеспечивает передачу на В0-В28 состояний соответствующих битов РМК (управляющих сигналов У0-У28).

Разряды РМК, содержащие 1, создают открывающий управляющий сигнал, а

содержащие 0 - закрывающий. Подобная структура микрокоманды, где каждый бит используется для создания отдельного управляющего сигнала, называется горизонтальной.

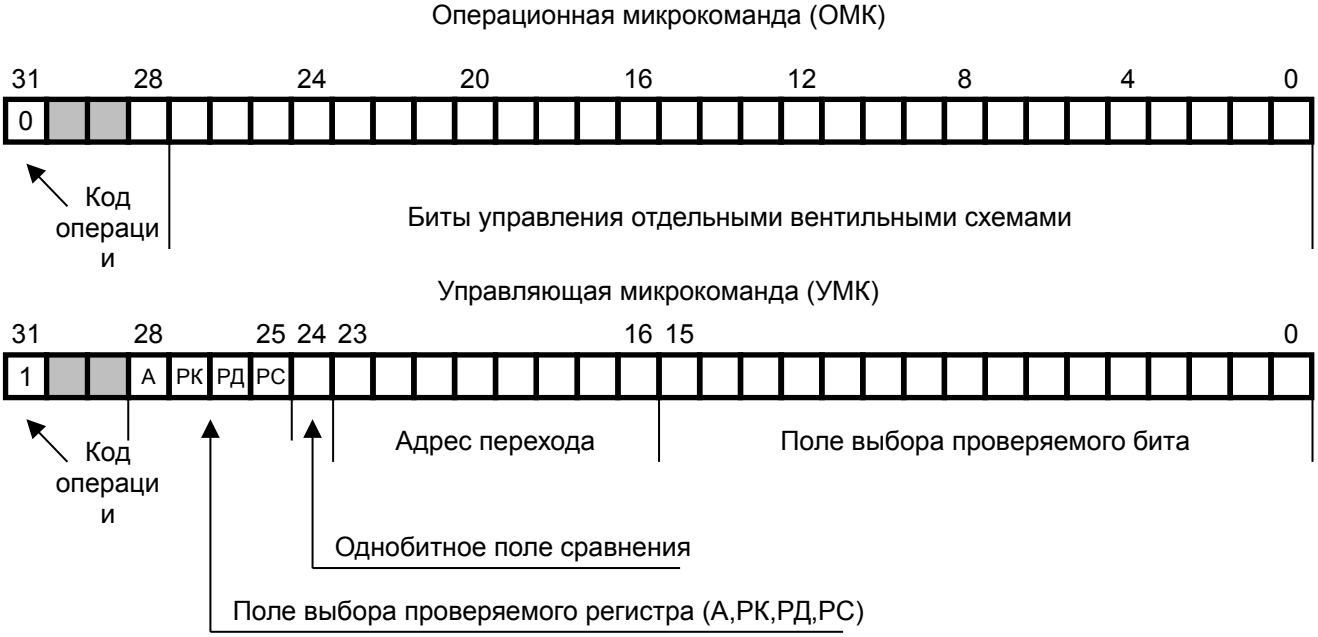


Рисунок В.13 Формат горизонтальной микрокоманды БЭВМ

Вентильные схемы В1, В2, В3 предназначены, соответственно, для передачи содержимого РД, РК, СК на правый вход АЛУ. Если все эти схемы закрыты (У1=У2=У3=0), то сигнал на правом входе АЛУ соответствует коду числа 0. Аналогично используются вентильные схемы В4, В5, В6, позволяющие передать на левый вход АЛУ содержимое А, РС, КР или кода числа 0.

Управляющие сигналы У7-У10 перестраивают АЛУ на выполнение различных микроопераций. При У7=...У10=0 в 17-разрядный буферный регистр АЛУ (БР) записывается сумма входных сигналов АЛУ: при У7=У8=У9=0 и У10=1 к такой сумме добавляется 1; при У7=У8=У10=0 и У9=1 в БР записывается результат логического умножения входных сигналов АЛУ; при У7=1 и (или) У8=1 можно получить аналогичные результаты, но для инверсных значений одного или двух входных сигналов.

Рассмотрим несколько примеров операционных микрокоманд.

1. Для вычитания содержимого РД из содержимого А и записи результата в буферный регистр (А - РД → БР) следует выполнить микрокоманду $(0000\ 0000\ 0000\ 0000\ 0000\ 0101\ 0001\ 0010)_2 = (0000\ 0512)_{16}$, т.е. одновременно подать единичные управляющие сигналы на В1, В4, В8 и В10. Тогда к уменьшаемому прибавится обратный код вычитаемого и к этой сумме добавится единица, что эквивалентно А + (-РД).

2. Для вычитания 1 из содержимого аккумулятора (А-1 → БР) надо выполнить микрокоманду $(0000\ 0110)_{16}$, т.е. подать единичные управляющие сигналы на В4 и В8 и сложить содержимое А с обратным кодом числа 0 или, что то же самое, с числом -1, записанным в дополнительном коде.

3. Для увеличения на 1 содержимого СК (СК + 1 → БР) надо выполнить микрокоманду $(0000\ 0408)_{16}$, т.е. открыть В3 и В10.

Вентильные схемы В11 и В12 позволяют записать в БР сдвинутое на один разряд вправо или влево содержимое аккумулятора. При этом "лишний" разряд БР заполняется содержимым регистра переноса С.

Вентильные схемы В13-В15 используются для передачи в младшие три бита регистра состояния, являющимися признаками результата С, N, Z операции, выполненной в АЛУ. В С и N копируются соответственно 16-й и 15-й бит БР. В Z записывается 1 в случае, если младшие 16 бит БР равны 0 (данная операция

реализуется на аппаратном уровне схемой ИЛИ-НЕ). Управляющие сигналы У16 и У17 позволяют установить признак С в 0 или 1 независимо от результата выполнения операции, сохраняемого в БР.

Вентильные схемы В18-В22 позволяют переписать содержимое 16 или 11 младших разрядов в РА, РД, РК, СК и А соответственно.

Вентильные схемы В23-В28 используются для организации обмена информацией между регистрами процессора и другими подсистемами ЭВМ (памятью и устройствами ввода-вывода).

Вентильная схема В0 используется в команде HLT для передачи сигнала прекращения выполнения программы и записывает 0 в 8-й бит РС.

Если из памяти микрокоманд выбрана управляющая микрокоманда, то в 31-ый бит РМК записывается 1 (код операции УМК). Этот сигнал открывает вентильную схему БР1 и тем самым создает условия для выполнения УМК. Теперь по сигналу, создаваемому каким-либо битом поля выбора проверяемого регистра (У1, У2, У4 или У5) открывается из вентильных схем В1, В2, В4 или В5 и на вентили ВВ0-ВВ15 поступает через АЛУ содержимое соответствующего регистра (РД, РК, А или РС). Одновременно на эти же вентили поступает с РМК содержимое поля выбора проверяемого бита. Так как в этом поле записана только одна 1 (на месте, соответствующем проверяемому биту), то открывается лишь один из вентилей ВВ0-ВВ15, через который на схему сравнения поступает содержимое проверяемого бита из проверяемого регистра. На другой вход этой схемы поступает содержимое однобитового поля сравнения (24-ый бит УМК), в которое при кодировании УМК записали 0 или 1.

Если проверяемый бит и бит из поля сравнения идентичны, то схема сравнения формирует единичный сигнал, который открывает вентильную схему ВА и на СчМК пересылается адрес перехода (16-24-ый биты УМК). В противном случае на СчМК передается нулевое значение, которое инициирует увеличение значения СчМК на единицу (используется схема ИЛИ-НЕ для всех битов адреса перехода).

При организации ветвлений в микропрограмме используется содержимое регистра состояний, являющегося объединением признаков результата и служебных битов состояния ЭВМ (табл. В.9). На структурной схеме (рис. В.12) признаки результата C, N, Z для удобства изображены в виде самостоятельных регистров, хотя они являются частью РС.

Таблица В.9.

Содержимое регистра состояния

Разряд	Содержимое
0	Перенос
1	Нуль
2	Знак
3	0 - используется для организации безусловных переходов в МПУ
4	Разрешение прерывания
5	Прерывание
6	Состояние ВУ (Ф)
7	Состояние тумблеров РАБОТА/ОСТАНОВ (1 - РАБОТА)
8	Программа

Рассмотрим две управляющих микрокоманды:

1. После увеличения на 1 содержимого РД в команде ISZ надо проверить знаковый разряд РД (разряд с номером 15). Если этот разряд равен 1 (содержимое РД меньше нуля), то выполнение команды ISZ завершается. В противном случае необходимо прибавить 1 к содержимому СК, т.е. организовать пропуск команды, следующей за ISZ. Это разветвление (переход по адресу 8F) осуществляется с помощью микрокоманды $(858F\ 8000)_{16}$.
2. Для организации безусловного перехода (например, по адресу 90) используется 3-й бит регистра состояний, содержащий константу 0. Сравнение этого разряда с нулем, записанным в 24-ый разряд УМК, всегда дает положительный результат и позволяет переслать в СчМК нужный адрес перехода. Микрокоманда, реализующая эту операцию, имеет вид: (828F

$0008)_{16}$.

3.2 Интерпретатор базовой ЭВМ.

Полный текст микропрограммы (интерпретатора команд) приведен в табл. В.10. Первые микрокоманды интерпретатора служат для выборки команды из основной памяти (ОП) базовой ЭВМ и определения ее типа: адресная, безадресная или ввода-вывода. Для этого содержимое СК (в котором хранится адрес исполняемой команды) пересылается через БР в РА (СК → БР и БР → РА). Затем из ячейки ОП, на которую указывает РА, пересылается в РД команда, а содержимое СК увеличивается на единицу и пересылается в БР: ОП(РА) → РД, СК+1 → БР. Далее содержимое БР, т.е. адрес следующей команды, пересылается в СК, а команда пересылается из РД в РК, после чего начинается ее дешифрация.

Так как адресные команды (команды с кодами операции от 0 до D) обязательно содержат ноль в 15, 14 или 13 бите, то проверкой этих битов РК можно выделить адресную команду и перейти к проверке ее 2-го бита (бита вида адресации). Для разделения команд ввода-вывода (код операции E) и безадресных команд (код операции F) достаточно проанализировать 12-ый бит РК: если этот бит равен 1, то надо переходить к микрокомандам продолжения дешифрации безадресных команд, расположенных, начиная с адреса 5E (метка БАД). В комментариях микрокоманда анализа 12-го бита РК записана в виде:

```
IF BIT(12,РК)=1 THEN БАД(5E).
```

В памяти микрокоманд нет полных микропрограмм для адресных команд с кодами операций 7 и D, а также для безадресных команд FC00, FD00, FE00 и FF00. Когда при декодировании команды выясняется, что выбрана команда 7xxx, управление передается ячейке с адресом B0. Начиная с этой ячейки, могут располагаться микрокоманды какой-либо новой арифметической команды (например, умножения). Для микропрограмм реализации команды перехода и безадресных команд выделены участки памяти микрокоманд с начальными адресами D0 и E0.

В базовой ЭВМ реализован и другой вариант интерпретатора, использующий более короткие - вертикальные микрокоманды (столбец "ВЕРТ." табл. В.10). Эти микрокоманды состоят из полей, в которых закодированы требуемые наборы управляющих сигналов (рис. В.14). Для декодирования используются дополнительные устройства - дешифраторы.

Таблица В.10

Интерпретатор базовой ЭВМ (микропрограмма)

Адрес	Микрокоманды		Комментарии	
	Горизонт.	Верт.	Метка	Действие
Цикл выборки команды				
01	0000 0008	0300	НАЧ	СК → БР
02	0004 0000	4001		БР → РА
03	0080 0408	0311		ОП(РА) → РД, СК + 1 → БР
04	0020 0000	4004		БР → СК
05	0000 0002	0100		РД → БР
06	0010 0000	4003		БР → РК
				Определение типа команды
07	880C 8000	AF0C		IF BIT(15,РК) = 0 THEN АДЦ(0C)
08	880C 4000	AE0C		IF BIT(14,РК) = 0 THEN АДЦ(0C)
09	880C 2000	AD0C		IF BIT(13,РК) = 0 THEN АДЦ(0C)
0A	895E 1000	EC5E		IF BIT(12,РК) = 1 THEN БАД(5E)
0B	828E 0008	838E		GOTO В/В(8E)
				Определение вида адресации
0C	881D 0800	AB1D	АДЦ	IF BIT(11,РК) = 0 THEN АДР(1D)
Цикл выборки адреса операнда				
0D	0000 0002	0100		РД → БР
0E	0004 0000	4001		БР → РА
0F	0080 0000	0001		ОП(РА) → РД
10	881D 0008	A31D		IF BIT(3,РК) = 0 THEN АДР(1D)
11	891D 0010	E41D		IF BIT(4,РК) = 1 THEN АДР(1D)
12	891D 0020	E51D		IF BIT(5,РК) = 1 THEN АДР(1D)
13	891D 0040	E61D		IF BIT(6,РК) = 1 THEN АДР(1D)
14	891D 0080	E71D		IF BIT(7,РК) = 1 THEN АДР(1D)
15	891D 0100	E81D		IF BIT(8,РК) = 1 THEN АДР(1D)
16	891D 0200	E91D		IF BIT(9,РК) = 1 THEN АДР(1D)
17	891D 0400	EA1D		IF BIT(10,РК) = 1 THEN АДР(1D)
18	0000 0402	0110		РД + 1 → БР
19	0008 0000	4002		БР → РД
1A	0100 0000	0002		РД → ОП(РА)
1B	0000 0082	0140		РД + COM(0) = РД - 1 → БР
1C	0008 0000	4002		БР → РД
Цикл исполнения адресных команд				
				Декодирование адресных команд
1D	892D 8000	EF2D	АДР	IF BIT(15,РК) = 1 THEN ПРХ(2D)
1E	0000 0002	0100		РД → БР
1F	0004 0000	4001		БР → РА
20	8927 4000	EE27		IF BIT(14,РК) = 1 THEN АРФ(27)
21	8824 2000	AD24		IF BIT(13,РК) = 0 THEN A1(24)
22	8857 1000	AC57		IF BIT(12,РК) = 0 THEN JSR(57)
23	8238 0008	8338		GOTO MOV(38)
24	0080 0000	0001	A1	ОП(РА) → РД
25	8850 1000	AC50		IF BIT(12,РК) = 0 THEN ISZ(50)
26	8235 0008	8335		GOTO AND(35)
27	0080 0000	0001	АРФ	ОП(РА) → РД
28	882B 2000	AD2B		IF BIT(13,РК) = 0 THEN СУМ(2B)
29	8843 1000	AC43		IF BIT(12,РК) = 0 THEN SUB(43)
2A	82B0 0008	83B0		GOTO Р - А(B0)
2B	883C 1000	AC3C	СУМ	IF BIT(12,РК) = 0 THEN ADD(3C)
2C	823F 0000	833F		GOTO ADC(3F)
2D	8830 4000	AE30	ПРХ	IF BIT(14,РК) = 0 THEN УПХ(30)
2E	8847 1000	AC47		IF BIT(12,РК) = 0 THEN BR(47)
2F	82D0 0008	83D0		GOTO Р - П(D0)
30	8833 2000	AD33	УПХ	IF BIT(13,РК) = 0 THEN П1(33)
31	884C 1000	AC4C		IF BIT(12,РК) = 0 THEN BMI(4C)
32	824E 0008	834E		GOTO BEQ(4E)
33	8846 1000	AC46	П1	IF BIT(12,РК) = 0 THEN BCS(46)
34	824A 0008	834A		GOTO BPL(4A)
				Исполнение адресных команд
35	0000 0212	1120	AND	А & РД → БР
36	0040 C000	4035		БР → А, N, Z
37	8290 0008	8390		GOTO ПРЕ(90)

Адрес	Микрокоманды		Комментарии	
	Горизонт.	Верт.	Метка	Действие
38	0000 0010	1000	MOV	A → БР
39	0008 0000	4002		БР → РД
3A	0100 0000	0002		РД → ОП(РА)
3B	8290 0008	8390		GOTO ПРЕ(90)
3C	0000 0012	1100	ADD	A + РД → БР
3D	0040 E000	4075		БР → A, C, N, Z
3E	8290 0008	8390		GOTO ПРЕ(90)
3F	823C 0001	803C	ADC	IF BIT(0,PC) = 0 THEN ADD(3C)
40	0000 0412	1110		A + РД + 1 → БР
41	0040 E000	4075		БР → A, C, N, Z
42	8290 0008	8390		GOTO ПРЕ(90)
43	0000 0512	1190	SUB	A + COM(РД) + 1 = A - РД → БР
44	0040 E000	4075		БР → A, C, N, Z
45	8290 0008	8390		GOTO ПРЕ(90)
46	8290 0001	8090	BCS	IF BIT(0,PC) = 0 THEN ПРЕ(90)
47	0000 0002	0100	BR	РД → БР
48	0020 0000	4004		БР → СК
49	8290 0008	8390		GOTO ПРЕ(90)
4A	8390 0004	C290	BPL	IF BIT(2,PC) = 1 THEN ПРЕ(90)
4B	8247 0008	8347		GOTO BR(47)
4C	8290 0004	8290	BMI	IF BIT(2,PC) = 0 THEN ПРЕ(90)
4D	8247 0008	8347		GOTO BR(47)
4E	8290 0002	8190	BEQ	IF BIT(1,PC) = 0 THEN ПРЕ(90)
4F	8247 0008	8347		GOTO BR(47)
50	0000 0402	0110	ISZ	РД + 1 → БР
51	0008 0000	4002		БР → РД
52	0100 0000	0002		РД → ОП(РА)
53	8590 8000	DF90		IF BIT(15,РД) = 1 THEN ПРЕ(90)
54	0000 0408	0310	SKP	СК + 1 → БР
55	0020 0000	4004		БР → СК
56	8290 0008	8390		GOTO ПРЕ(90)
57	0000 0402	0110	JSR	РД + 1 → БР
58	0010 0000	4003		БР → РК
59	0000 0008	0300		СК → БР
5A	0008 0000	4002		БР → РД
5B	0100 0004	0202		РД → ОП(РА), РК → БР
5C	0020 0000	4004		БР → СК
5B	8290 0008	8390		GOTO ПРЕ(90)
Цикл исполнения безадресных команд				
				Декодирование безадресных команд
5E	8861 0800	AB61	БАД	IF BIT(11,РК) = 0 THEN Б0(61)
5F	886C 0400	AA6C		IF BIT(10,РК) = 0 THEN Б1(6C)
60	82E0 0008	83E0		GOTO Р - Б(E0)
61	8867 0400	AA67	Б0	IF BIT(10,РК) = 0 THEN Б2(67)
62	8865 0200	A965		IF BIT(9,РК) = 0 THEN Б3(65)
63	8882 0100	A882		IF BIT(8,РК) = 0 THEN ROL(82)
64	8285 0008	8385		GOTO ROR(85)
65	887B 0100	A87B	Б3	IF BIT(8,РК) = 0 THEN CMA(7B)
66	827E 0008	837E		GOTO CMC(7E)
67	886A 0200	A96A	Б2	IF BIT(9,РК) = 0 THEN Б4(6A)
68	8876 0100	A876		IF BIT(8,РК) = 0 THEN CLA(76)
69	8279 0008	8379		GOTO CLC(79)
6A	8888 0100	A888	Б4	IF BIT(8,РК) = 0 THEN HLT(88)
6B	8287 0008	8387		GOTO NOP(87)
6C	886F 0200	A96F	Б1	IF BIT(9,РК) = 0 THEN Б5(6F)
6D	888A 0100	A88A		IF BIT(8,РК) = 0 THEN EI(8A)
6E	828C 0008	838C		GOTO DI(8C)
6F	8873 0100	A873	Б5	IF BIT(8,РК) = 0 THEN INC(73)
				Исполнение безадресных команд
70	0000 0110	1080	DEC	A + COM(0) = A - 1 → БР
71	0040 E000	4075		БР → A, C, N, Z
72	8290 0008	8390		GOTO ПРЕ(90)
73	0000 0410	1010	INC	A + 1 → БР
74	0040 E000	4075		БР → A, C, N, Z

Адрес	Микрокоманды		Комментарии	
	Горизонт.	Верт.	Метка	Действие
75	8290 0008	8390		GOTO ПРЕ(90)
76	0000 0200	0020	CLA	0 → БР
77	0040 C000	4035		БР → A, N, Z
78	8290 0008	8390		GOTO ПРЕ(90)
79	0001 0000	4080	CLC	0 → C
7A	8290 0008	8390		GOTO ПРЕ(90)
7B	0000 0090	1040	CMA	COM(A) → БР, инверсия A
7C	0040 C000	4035		БР → A, N, Z
7D	8290 0008	8390		GOTO ПРЕ(90)
7E	8280 0001	8080	CMC	IF BIT(0,PC) = 0 THEN Б6(80)
7F	8279 0008	8379		GOTO CLC(79)
80	0002 0000	40C0	Б6	1 → C
81	8290 0008	8390		GOTO ПРЕ(90)
82	0000 1000	0008	ROL	RAL(A) → БР, сдвиг влево
83	0040 E000	4075		БР → A, C, N, Z
84	8290 0008	8390		GOTO ПРЕ(90)
85	0000 0800	0004	ROR	RAR(A) → БР, сдвиг вправо
86	0040 E000	4075		БР → A, C, N, Z
87	8290 0008	8390	NOP	GOTO ПРЕ(90)
88	0000 0001	4008	HLT	Останов машины
89	8201 0008	8301		GOTO НАЧ(01)
8A	1000 0000	4800	EI	Разрешение прерывания
8B	8201 0008	8301		GOTO НАЧ(01)
8C	0800 0000	4400	DI	Запрещение прерывания
8D	8201 0008	8301		GOTO НАЧ(01)
Цикл исполнения команд ввода-вывода				
8E	0200 0000	4100	В/В	Передача адреса и приказа на ВУ
8F	8354 0040	C654		IF BIT(6,PC) = 1 THEN SKP(54)
Цикл прерывания				
90	8288 0080	8788	ПРЕ	IF BIT(7,PC) = 0 THEN HTL(88)
91	8201 0020	8501		IF BIT(5,PC) = 0 THEN НАЧ(01)
92	0000 0200	0020		0 → БР
93	0004 0000	4001		БР → РА
94	0000 0008	0300		СК → БР
95	0008 0000	4002		БР → РД
96	0100 0400	0012		РД → ОП(РА), 1 → БР
97	0020 0000	4004		БР → СК
98	0800 0000	4400		Запрещение прерывания
99	8201 0008	8301		GOTO НАЧ(01)
Пультовые операции				
				Ввод адреса
9A	0000 0040	3000	В/А	КР → БР
9B	0020 0000	4004		БР → СК
9C	8288 0008	8388		GOTO HTL(88)
				Чтение
9D	0000 0008	0300	ЧТ	СК → БР
9E	0004 0000	4001		БР → РА
9F	0080 0408	0311		ОП(РА) → РД, СК + 1 → БР
A0	0020 0000	4004		БР → СК
A1	8288 0008	8388		GOTO HTL(88)
				Запись
A2	0000 0008	0300	ЗАП	СК → БР
A3	0004 0000	4001		БР → РА
A4	0000 0040	3000		КР → БР
A5	0008 0000	4002		БР → РД
A6	0100 0408	0312		РД → ОП(РА), СК + 1 → БР
A7	0020 0000	4004		БР → СК
A8	8288 0008	8388		GOTO HTL(88)
				Пуск
A9	0000 0200	0020	ПУС	0 → БР
AA	005C E000	4077		БР → A, C, N, Z
AB	0400 0000	4200		Сброс флагов ВУ
AC	0800 0000	4400		Запрещение прерывания
AD	8290 0008	8390		GOTO ПРЕ(90)

Адрес	Микрокоманды		Комментарии	
	Горизонт.	Верт.	Метка	Действие
. . .				
B0			P - A	Арифметическая команда 7###
. . .				
D0			P - П	Команда перехода D###
. . .				
E0			P - Б	Безадресные команды FC## - FF##
. . .				
FF				

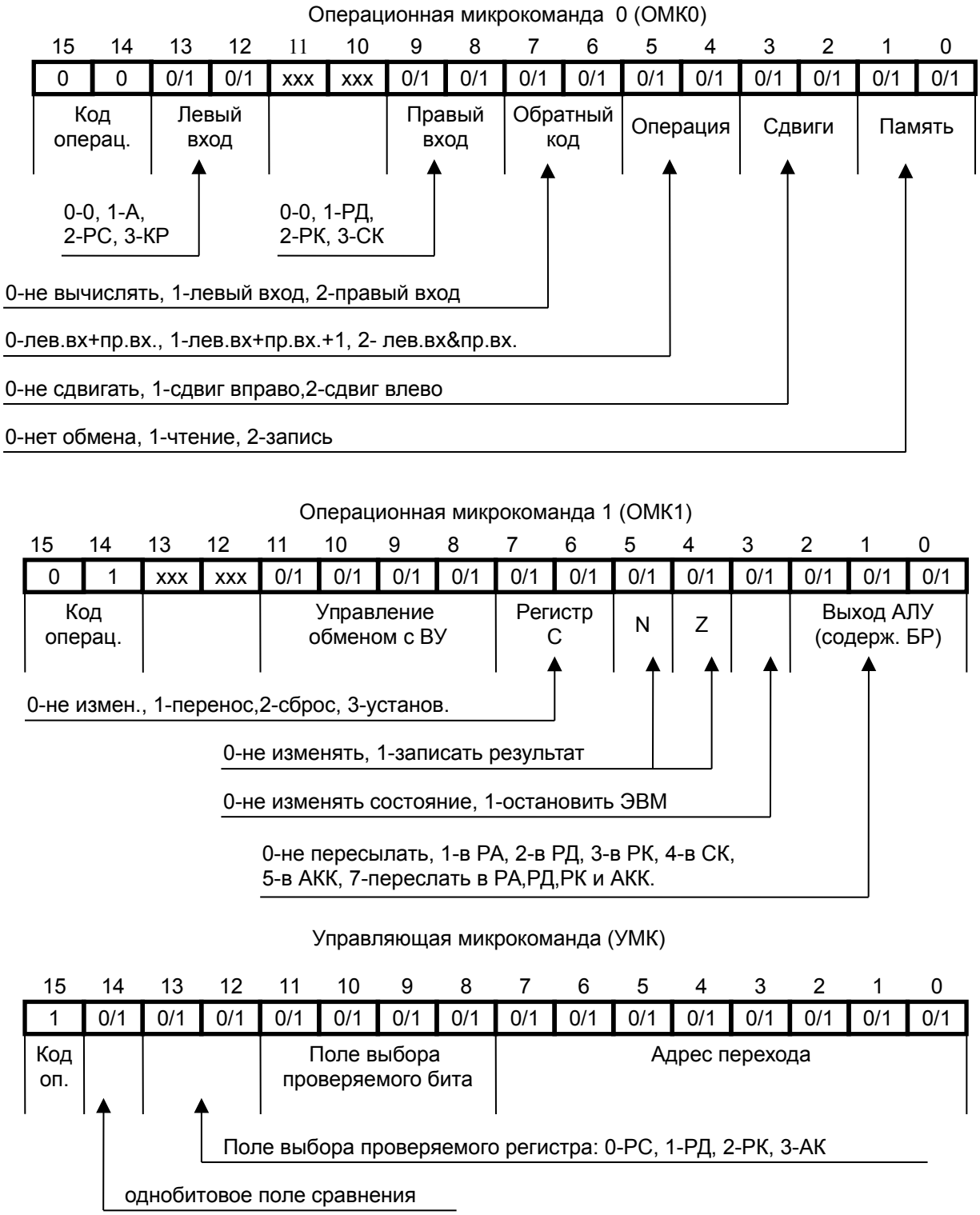


Рис. В.14 Форматы вертикальных микрокоманд

Приложение Г. Инструкция по работе с моделью БЭВМ

Для перемещения в клавишном регистре используются следующие клавиши:

RIGHT	Перемещение указателя на одну позицию вправо.
LEFT	Перемещение указателя на одну позицию влево.
UP	Инверсия бита (изменение значения на противоположное) по текущему положению указателя
1	Занесение 1 по текущему положению указателя и перемещение его на на следующую позицию
0	Занесение 0 по текущему положению указателя и перемещение его на на следующую позицию

В процессе работы также используются клавиши:

F4	Ввод адреса. По этой клавише содержимое клавишного регистра заносится в счетчик команд.
F5	Запись. Информация из клавишного регистра заносится в память по текущему содержимому счетчика команд.
F6	Чтение. Из ячейки памяти (по адресу расположенному в счетчике команд) информация читается в регистр данных.
F7	Пуск. Операция сбрасывает содержимое аккумулятора и флага переноса, сбрасывает готовность всех внешних устройств, запрещает прерывания и, если установлен режим Работа, переходит к выполнению команды, адрес которой указан в счетчике команд.
F8	Продолжение. В режиме "ОСТАНОВ" происходит исполнение одной инструкции, а в режиме "ОСТАНОВ" продолжение выполнения программы с адреса в регистре команд
F9	Клавиша, управляющая переключением режима работы базовой ЭВМ. Производит переключение режимов "РАБОТА" и "ОСТАНОВ".
F10	Выход из базовой ЭВМ.
Shift+F4	Смена маски.

Работа с внешними устройствами обеспечивается клавишами:

F1,F2,F3	Готовность внешнего устройства 1,2,3 соответственно.
Tab	Переход в режим ввода в регистры данных ВУ2 и ВУ3.

Для работы с микрокомандами используйте клавиши:

Tab	Переключение ввода в обычную память и память микрокоманд. При вводе в память микрокоманд слева от клавишного регистра загорается индикатор МК.
Shift+F9	Включение/Отключение режима ТАКТ. В этом режиме при нажатии клавиши F8 (Продолжение) происходит выполнение одной микрокоманды.

Приложение Д. Ассемблер БЭВМ. Краткий справочник

Для упрощения разработки программ для БЭВМ и более наглядного их представления разработан язык ассемблера, позволяющий использовать дополнительные возможности для разработки программ.

Синтаксис

Назначение	**Синтаксис**	**Пример использования**
Управление размещением в памяти	ORG адрес	ORG 10
Адресная команда с прямой адресацией	[метка:] МНЕМОНИКА АРГУМЕНТ	MOV R
Адресная команда с косвенной адресацией	[метка:] МНЕМОНИКА (АРГУМЕНТ)	ADD (K)
Безадресная команда	[метка:] МНЕМОНИКА	BEGIN: CLA
Команда ввода-вывода	[метка:] МНЕМОНИКА АДРЕСВУ	OUT 3
Константы	[метка:] WORD значение [,значение...] [[метка:] WORD количество DUP (значение)	X: WORD ? Y: WORD X VALUES: WORD 1,2,3 ARRAY: WORD 10 DUP (?)

Метки, команды, их аргументы и т.п. должны быть отделены друг от друга пробелом или символом табуляции.

Описание директив

1. `ORG address` - указывает компилятору, что следующее значение необходимо располагать по указанному адресу. Похожа на пультовую команду "Ввод адреса". Обычно данную директиву достаточно использовать один раз в начале программы.
2. `WORD` - ввод констант и резервирование памяти. Может быть указано одно или более значений. Если в качестве значения указан вопросительный знак, то соответствующая ячейка памяти остается неинициализированной. Если указанное значение не удалось распознать как шестнадцатеричное число, то в качестве значения будет использован адрес метки с указанным именем. При использовании в синтаксисе `WORD` количество `DUP` (значение) соответствующее значение будет продублировано указанное количество раз.

Метки

Метки являются ссылками на соответствующие ячейки памяти. Могут использоваться как аргументы для адресных команд и для инициализации других ячеек адресом, на которую ссылается метка. В имени метки могут использоваться любые символы, однако, в связи с особенностями обработки констант, не рекомендуется использовать имена меток, которые могут быть восприняты как шестнадцатеричное число.

Специальные метки

1. BEGIN - указывает компилятору на первую выполняемую команду программы. Должна быть указана в любой программе.
2. R - указывает на ячейку, в которой будет располагаться результат. После успешной компиляции, если в программе была обнаружена метка R, консольная версия БЭВМ выведет адрес соответствующей ячейки памяти.

Комментарии

Любой текст в строке после символа ; или # считается комментарием, и не обрабатывается.

Пример. Подсчет количества неотрицательных элементов в массиве

Решение 1. Старый стиль

```
ORG     00F
        WORD    0020
BEGIN:  WORD    F200,480F,A017,F200,401B,F800
        WORD    301B,001A,C010,F000,FFFA,0000
ORG     020
        WORD    0001,FFFF,0002,FFFE,0003,FFFD
```

+ Ввод программы максимально приближен к обычной работе с БЭВМ.
- Ассемблер не используется
- Непонятно, где программа, где данные и результат

Решение 2. Студенческое

```
ORG     00F
        WORD    0020

BEGIN:  CLA
        ADD     (00F)
        BMI     SKIP
        CLA
        ADD     R
        INC
        MOV     R
SKIP:   ISZ     01A
        BR      BEGIN
        HLT

        WORD    -6
R:      WORD    ?

ORG     020
X:      WORD    0001, FFFF, 0002, FFFE, 0003, FFFD
```

+ Отсутствие необходимости использовать пультовые операции
- Нереэнтерабельно. Повторный запуск программы не будет работать.
- Жестко заданный массив и количество элементов

Решение 3. Рекомендуемое

```
ORG     00F
K:      WORD    ? ; Адрес первого
                  ; элемента массива
BEGIN:  CLA
        MOV     R
        ADD     (K)
        BMI     SKIP
        CLA
        ADD     R
        INC
        MOV     R
SKIP:   ISZ     N
        BR      BEGIN
        HLT

N:      WORD    ? ; Количество элементов массива
R:      WORD    ? ; Результат

ORG     020
X:      WORD    6 DUP (?) ; Элементы массива
```

+ Видны все исходные данные, используемые программой.
+ При повторном запуске программы с новыми исходными данными результат будет корректным.
- Перед запуском программы пользователь должен самостоятельно ввести все исходные данные.
- Пользователь может ошибочно посчитать, что программа может работать только с массивом из 6 элементов.