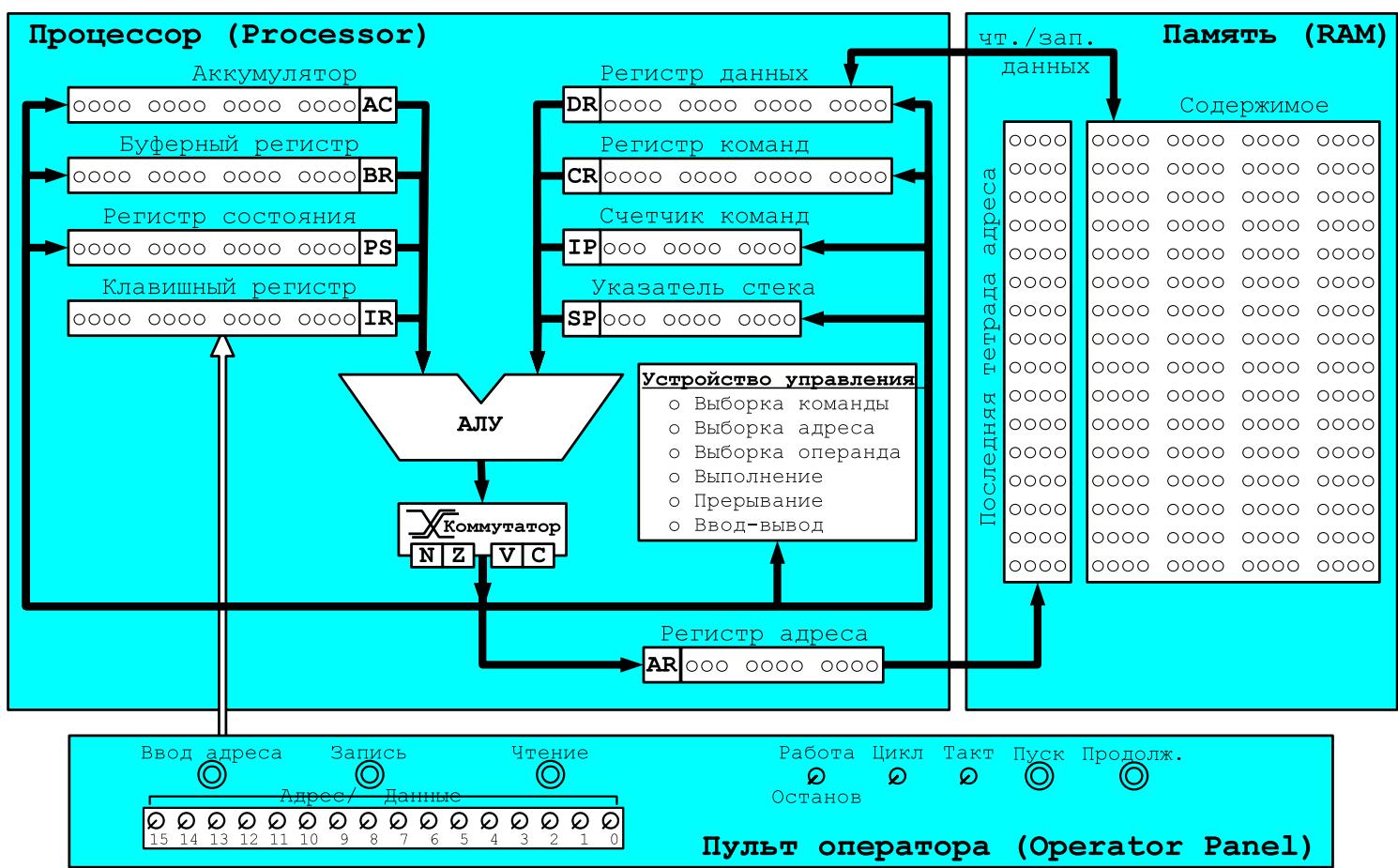


Методические указания к лабораторным работам по курсу "Основы профессиональной деятельности"



Настоящее методическое пособие предназначено для практического закрепления материала по дисциплине "Основы профессиональной деятельности" (Направление подготовки — 09.03.01 "Информатика и вычислительная техника" и 09.03.04 "Программная инженерия"), преподавание которой организовано по модульному принципу и включает лабораторные и контрольные работы.

В методическом пособии для каждой лабораторной работы описаны цель, задание, порядки подготовки и выполнения, содержание отчета, контрольные вопросы для подготовке к защите работы. Персональные варианты лабораторных работ доступны на сайте <https://se.ifmo.ru/courses/csbasics>.

В приложениях:

- описаны основные команды UNIX, знание которых необходимо для сдачи лабораторной работы №1;
- приведено справочное описание учебной (базовой) ЭВМ (БЭВМ-NG), которая обладает типичными чертами многих современных ЭВМ. На базовой ЭВМ выполняются лабораторные работы, позволяющие исследовать порядок функционирования ЭВМ при выполнении программ различных типов, подходы к организации ввода-вывода информации, принципы микропрограммного управления;
- справочно изложены инструкция по работе с базовой ЭВМ и описание языка ассемблера БЭВМ-NG;
- установлена система оценки различных видов работ в рамках дисциплины.

Содержание

Раздел 1. Знакомство с факультетом ПИиКТ.....	4
Факультет Программной инженерии и компьютерной техники.....	4
Лабораторные работы курса ОПД.....	4
Лабораторная работа №1. Основные команды ОС семейства UNIX.....	5
Рубежный контроль №1.....	6
Раздел 2. Введение в базовую ЭВМ.....	7
Лабораторная работа №2. Исследование работы БЭВМ.....	7
Рубежный контроль №2.....	8
Лабораторная работа №3. Выполнение циклических программ.....	9
Лабораторная работа №4. Выполнение комплекса программ.....	9
Раздел 3. Организация ввода-вывода информации в БЭВМ.....	10
Лабораторная работа №5. Асинхронный обмен данными с ВУ.....	10
Лабораторная работа №6. Обмен данными с ВУ по прерыванию.....	11
Раздел 4. Организация микропрограммного устройства БЭВМ.....	12
Лабораторная работа №7. Синтез команд БЭВМ.....	13
Рубежный контроль №3.....	15
Литература.....	15
Приложение А. Краткий перечень и функциональность команд UNIX.....	16
Приложение Б. Накопление баллов БАРС в течении курса.....	17
Приложение В. Состав, структура и функционирование БЭВМ-NG.....	18
Часть 1. Базовая ЭВМ.....	18
1.1 Назначение базовой ЭВМ.....	18
1.2 Структура базовой ЭВМ.....	18
1.3. Система команд базовой ЭВМ.....	20
1.4 Представление данных в БЭВМ.....	22
1.4.1 Представление чисел в БЭВМ.....	23
1.5 Операции с памятью и арифметические операции.....	25
1.6 Сдвиги и логические операции.....	26
1.7 Управление вычислительным процессом.....	26
1.8 Подпрограммы и стек.....	28
1.9 Выполнение машинных команд.....	29
Часть 2. Организация ввода-вывода в базовой ЭВМ.....	33
2.1 Устройства ввода-вывода базовой ЭВМ.....	33
2.2 Команды ввода-вывода.....	37
2.3 Программно-управляемый асинхронный обмен.....	37
2.4 Управляемый по прерыванию программы ввод-вывод.....	38
Часть 3. Микропрограммное устройство управления.....	40
3.1. Микропрограммное управление вентильными схемами.....	40
3.2 Интерпретатор базовой ЭВМ.....	45
Приложение Г. Инструкция по работе с моделью БЭВМ.....	50
Приложение Д. Ассемблер БЭВМ. Краткий справочник.....	51
Замеченные (и не исправленные) очепятки и логические несостыковки.....	53

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 были разработаны 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, Карапетяна Эрика Акоповича, Клименкова Сергея Викторовича, Лемешева Алексея Сергеевича, Максимова Андрея Николаевича, Майорова Сергея Александровича, Мартынова Николая Васильевича, Перминова Илью Валентиновича, Приблуду Андрея Анатольевича, Приблуду Константина Анатольевича, Перцева Тимофея Сергеевича, Щелокова Ивана Викторовича, а также большое количество студентов, аспирантов и выпускников кафедры ВТ.

В 2019 году Афанасьев Д.Б. и Клименков С.В. выступили инициаторами и главными исполнителями проектирования и реализации модели БЭВМ-NG в которую были добавлены необходимые функциональные возможности для обучения студентов, к которым относились стек, режимы адресации, расширенный набор команд, организация ввода-вывода и прерываний. В разработке системы команд и эмулятора также активное участие приняли Медведева Елизавета, Доморацкий Эридан, Щербаков Виктор, Гаврилов Антон Валерьевич и Перминов Илья Валентинович.

В методическом пособии содержится информация, необходимая для успешной сдачи всех лабораторных работ, включая специально разработанную в образовательных целях учебную ЭВМ (базовую ЭВМ), обладающую типичными чертами многих современных ЭВМ. Лабораторная работа №1 предназначена для демонстрации студентам первого курса предмета "Системное программное обеспечение", 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 факультетом ПИиКТ

Факультет Программной инженерии и компьютерной техники

Факультет Программной инженерии и компьютерной техники ведет подготовку бакалавров по направлениям подготовки 09.03.01 - Информатика и Вычислительная техника (профиль подготовки "Компьютерные системы и технологии") и 09.03.04 - Программная инженерия (профили подготовки "Системное и прикладное программное обеспечение" и "Нейротехнологии и программирование").

Лабораторные работы курса ОПД

В рамках курсах предусмотрено 7 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! Для дистанционной сдачи допускается скреплять листы форматом PDF.

Лабораторная работа №1. 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 `cat`, `wc`, `ls`, `head`, `tail`, `echo`, `sort`, `grep` 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 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 по одной команде на строке (командный скрипт) для возможности автоматического повтора команд этого и последующих пунктов. Изменить права на файлы, согласно п.2 задания. Выполнить п.3, включив в скрипт выполняемые команды. В случае недостатка прав для выполнения операции п.3, необходимо изменить командой `chmod` права файлов перед выполнением команды, а после выполнения вернуть их обратно. Выполнить команды вывода содержимого дерева (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 `ls -lR` из директории `lab0`, после выполнения п.3 задания.
- Задания, команды и результаты их выполнения (если есть), включая сообщения о возникающих ошибках и, для п.3 и п.5, команды, исправляющую ошибки.
- Файл с последовательностью команд по всей лабораторной работе.

Контрольные вопросы:

- Расскажите про команду {имя команды}. Какие она принимает аргументы (их количество, формат, способ задания)? Является ли данная команда фильтром?
- Стандартные потоки ввода-вывода, назначение. Способы управления стандартными потоками в shell.
- Стандартные права доступа к файлам и каталогам. Способы задания прав при помощи команды `chmod`. Интерпретация вывода команды `ls -l`.
- Файлы в ОС UNIX. Типы файлов. Символические и жесткие ссылки.

Рубежный контроль №1

В рубежном контроле раздела три задания.

Задание 1. Привести последовательность команд, создающих приведенное дерево каталогов и файлов с содержимым, начиная с создания директории `lab0`.

```
/home/s113369/lab0 (каталог)
+--darumaka1 (каталог)
|   +--wurmple (каталог)
|   +--chinchou (каталог)
+--houndour1 (файл)
```

Содержимое файлов

`houndour1`:

```
Способности Howl Smog Roar Bite Odor Sleuth
Beat Up Fire Flamethrower Crunch
Nasty Plot Inferno
```

Различными способами установить права на файлы и директории:

```
darumaka1: -wxrwx-wx
wurmple: rwx-wx-wx
chinchou: права 307
houndour1: rw----r--
```

Решение. Создадим последовательно необходимые файлы и директории в домашнем каталоге и назначим им права. Один из множества возможных вариантов.

```
cd /home/s113369; umask 022
mkdir lab0; cd lab0
mkdir -p darumaka1/wurmple darumaka1/chinchou
echo "Способности Howl Smog Roar Bite Odor Sleuth" > houndour1
echo "Beat Up Fire Flamethrower Crunch" >> houndour1
echo "Nasty Plot Inferno" >> houndour1
chmod go=rx darumaka1/wurmple
chmod 307 darumaka1/chinchou
chmod g-r houndour1
chmod 373 darumaka1 #должно быть последним, чтобы корректно
#установились вложенные права
```

Задание 2. Перевести заданные числа ($X=31681_{10}$, $Y=13825_{10}$) в 16-ричную систему счисления и представить разрядной сетке памяти БЭВМ. Вычислить значение $R=X+Y$, результат представить как число со знаком в 16-ричной системе счисления разрядной сетке БЭВМ. Укажите, произошло ли переполнение при операции? Был ли перенос?

Решение. Перевод чисел в шестнадцатеричную систему счисления и упаковка

исходных чисел в разрядную сетку БЭВМ (см. Приложение В, раздел 1.4) следующая: $X=7BC1_{16}$, $Y=3601_{16}$. Результат сложения $R=7BC1_{16}+3601_{16}=B1C2_{16}$. В ответе необходимо привести последовательность перевода, упаковки в дополнительный код (при необходимости) и сложения в столбик. Разрешается использовать промежуточное представление в виде двоичного числа.

Перенос в 17 разряд (формирование признака С) не произошел потому, что результат меньше 65535. Переполнение (признак результата V) произошло потому, что результат больше максимально допустимого положительного числа в знаковом представлении (32767). В знаковом представлении результат (-20030_{10}) является отрицательным числом, что является ошибкой, т. к. ведет к потере ожидаемого результата сложения (45506).

Задание 3. Какую операцию выполняет команда ADD AD? Перечислите все результаты этой команды.

Решение. Команда ADD AD относится к адресным командам с прямой абсолютной адресацией и складывает содержимое указанной в коде команды ячейки с аккумулятором (AC).

Результатом команды является измененные значения регистров и ячеек памяти, а именно: счетчик команд будет содержать адрес следующей выполняемой команды; регистр AC — сумму содержимого AC до исполнения команды и ячейки ADA; AR и DR — содержать число AD; CR содержать код команды 40AD; в BR будет содержимое IP до выполнения команды; будут установлены признаки результата NZVC в PS по вычисленной сумме.

Раздел 2. Введение в базовую ЭВМ

В рамках этого раздела студент должен изучить состав, структуру и принцип функционирования БЭВМ на уровне машинных команд, систему команд БЭВМ, детальную последовательность исполнения команд с прямой и косвенной адресациями, подпрограммы, основные подходы, применяемые для низкоуровневой обработки данных.

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 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, отделить ячейки данных от кода программы, написать назначение программы и реализуемую функцию, которую представить в виде формулы.

Во время допуска к работе получить у преподавателя исходные данные для переменных, согласовать вариант программы для исполнения, занести в память базовой ЭВМ заданный вариант программы и, выполняя ее по командам, заполнить таблицу трассировки выполненной программы. Занесение программы с данными, а также запуск программы в пультовом режиме продемонстрировать преподавателю.

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 функция (формула);
- описание и назначение исходных данных, область представления и область допустимых значений исходных данных и результата;
- расположение в памяти ЭВМ программы, исходных данных и результатов;
- адреса первой и последней выполняемой команды 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	IP	CR	AR	DR	SP	BR	AC	NZVC	Адрес	Новый код
xxx	xxxx	xxx	xxxx	xxxx	xxxx	xxx	xxxx	xxxx	xxxx	xxx	xxxx

4. Вариант программы с меньшим числом команд.

Контрольные вопросы:

- Форматы представления, области представления и области допустимых значений в БЭВМ для знаковых и беззнаковых чисел с фиксированной точкой и логических значений.
- Представление чисел в разрядной сетке в прямом, обратном и дополнительном кодах.
- Адресные и безадресные команды БЭВМ.
- Описание команды находящейся, по указанному адресу: наименование, назначение, тип команды и вид адресации. Количество и название машинных циклов, потактовое выполнение команды.
- Какую формулу реализует программа? Как можно упростить программу?
- Где находятся аргументы программы? Где находится результат? Как они представлены? Какие дополнительные ячейки использует программа? Для чего?
- Какое количество обращений к ячейкам памяти при выполнении безадресной команды? На каких циклах оно выполняется?

Рубежный контроль №2

Задание 1. Заполнить таблицу трассировки правильными значениями и восстановить формулу, вычисляемую программой. Пример варианта задания и правильного решения приведен в табл. 2.3.

Пример задания №2 и правильного ответа

Таблица 2.3

Исходные данные		Пример правильного ответа. Формула: C=+2*A-B+1								
Адрес	Команда/данные	Адрес	Команда/ данные	IP	CR	AR	DR	BR	AC	NZVC
00A	3BBD	00A	3BBD	-	-	-	-	-	-	-
00B	CF34	00B	CF34	-	-	-	-	-	-	-
00C	0000	00C	0000	-	-	-	-	-	-	-
00D	+LD 00A	00D	+LD 00A	00E	A00A	00A	3BBD	000D	3BBD	----
00E	ASL	00E	ASL	00F	0500	00E	3BBD	000E	777A	----
00F	SUB 00B	00F	SUB 00B	010	600B	00B	CF34	000F	A846	N-V-
010	INC	010	INC	011	0700	010	0700	0010	A847	N---
011	ST 00C	011	ST 00C	012	E00C	00C	A847	0011	A847	N---
012	HLT	012	HLT	013	0100	012	0100	0012	A847	N---

Лабораторная работа №3.

Выполнение циклических программ

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

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). Восстановить текст заданного варианта программы. Составить описание программы.

Порядок выполнения работы. Получить допуск к лабораторной работе, предъявив преподавателю подготовленные материалы. Получить у преподавателя данные для элементов массива. Элементы массива из начального задания используются только для определения функциональности программы! 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. Организация одномерных массивов данных в памяти. Организация и обработка массивов с числом измерений, больше чем одно.
2. Сравнение значений в БЭВМ. Команды условных и безусловного переходов.
3. Организация циклических вычислений. Команда LOOP.
4. Режимы адресации БЭВМ.
5. Описание адресных команд и команд переходов с различными режимами адресации: наименование, назначение, тип команды и вид адресации. Количество и название машинных циклов, потактовое выполнение команд.
6. Количество обращений к памяти команд БЭВМ с различными режимами адресации.
7. Где находятся аргументы программы? Где находится результат? Как они представлены?
8. Какое максимальное количество элементов данных может поддерживать ваша программа?

Лабораторная работа №4.

Выполнение 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 их предназначение и составить описание, определить область представления и область допустимых значений для исходных данных и возвращаемых значений подпрограммы, 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, нарисовать график функции, которая вычисляется в подпрограмме.

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 график функции, реализуемый подпрограммой.

Контрольные вопросы:

1. Организация подпрограмм в БЭВМ. Команды вызова подпрограммы и возврата.
2. Аргументы и возвращаемые значения подпрограммы. Способы организации передачи аргументов и возвращаемых значений.
3. Рекурсивный вызов подпрограмм. Организация стека.
4. Описание команд CALL и RET: наименование, назначение, тип команды и вид адресации. Количество и название машинных циклов, потактовое выполнение команды, количество обращений к памяти.

Раздел 3. 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, табл В.14). 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!
6. Опционально, по дополнительному заданию преподавателя, переделать устройство ввода-вывода (ВУ-1...ВУ-3) на ВУ-4...ВУ-10.

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 и текст исходной программы на языке Ассемблера БЭВМ. (синтаксис и особенности приведены в Приложении Д).

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, - 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 (DR DEV), регистра состояния ВУ (CR DEV)?
9. Какие элементы БЭВМ участвуют в обмене с ВУ? Укажите направление передачи данных между элементами при операциях ввода и вывода.

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 изменять содержимое заданной ячейки памяти (X), которое должно быть представлено как знаковое число. Область допустимых значений изменения X должна быть ограничена заданной функцией F(X) и конструктивными особенностями регистра данных ВУ (8-ми битное знаковое представление). Программа обработки прерываний должна модифицировать ячейку памяти для хранения X в соответствии с вариантом задания и выводить его на ВУ, а также игнорировать все необрабатываемые прерывания.

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 проверку работоспособности комплекса программ в автоматическом режиме.
4. В присутствии преподавателя, используя методику проверки разработанного комплекса программ, получить 3 пары результатов, указывая для каждого выведенного значения величину Х.
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. Описание программы, аналогично п.2 требований к отчету для ЛР№2.
2. Текст исходной программы на языке Ассемблера БЭВМ (синтаксис и особенности приведены в Приложении Д).
3. Полностью разработанную и проверенную на БЭВМ методику проверки.

Пример. Начальный фрагмент методики проверки.

1. Загрузить комплекс программ в память базовой ЭВМ.
2. Изменить значения точки останова по адресу на HLT
3. Запустить основную программу в автоматическом режиме с адреса
4. Установить "Готовность ВУ-3".
5. Дождаться останова
6. Записать содержимое аккумулятора в момент останова программы
7. Продолжить выполнение программы
8. ...

Контрольные вопросы:

1. Особенности организации программ обмена данными с использованием прерываний. Сохранение и восстановление значений регистров.
2. Команды работы разрешения/запрещения прерываний БЭВМ, команда программного прерывания, команда возврата из прерывания. Название, назначение и тип команды. Количество и название машинных циклов, потактовое выполнение команды.
3. Вектора прерываний. Преимущества использования векторов прерываний.
4. Регистр управления ВУ (MR DEV).
5. Сигналы шины БЭВМ, назначение, временные диаграммы сигналов Input и Output.
6. Когда выполняется цикл обработки прерывания? После каких команд он не выполняется? Почему?
7. Обрабатываются ли прерывания в пошаговом режиме (режиме "ОСТАНОВ") работы программы? Почему?
8. Что происходит при одновременном поступлении сигнала готовности нескольких внешних устройств? В какой последовательности они будут обработаны?
9. За что отвечают биты 5, 6, 7 и 8 регистра состояния? Когда изменяется их значение?

Раздел 4. 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 команды, соответствующей выданному преподавателем варианту задания. Разработать тестовую программу, которая проверяет синтезированную команду. Загрузить в микропрограммную память БЭВМ цикл исполнения синтезированной команды, модифицировать, при необходимости, основную микропрограмму, загрузить в основную память БЭВМ тестовую программу. 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 микрокоманды цикла исполнение одной из следующих команд в соответствии с вариантом задания:

- команда 9xxx — команда, предназначенная для выполнения адресных команд;
- команда Fxxx — команда, осуществляющая переход по заданному условию;
- безадресная команда с кодом 0FXX.

2. Написать тестовую программу для проверки правильности исполнения синтезированных команд базовой ЭВМ. Данная программа должна 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 адресных и безадресных команд результат их выполнения должен быть зафиксирован в выбранной ячейке памяти БЭВМ.
- Если проверяемая арифметическая или безадресная команда устанавливает признаки результата (биты N,Z,V,C), 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 разрабатываемых командах может понадобится цикл выборки операнда, а может оказаться нужен только цикл выборки адреса. Команда 9XXX выполняет оба цикла. Разумным решением будет изменение основной микропрограммы.
- После перехода на участок реализации команды в микропрограмме, возможно будет необходимым продолжить декодирование команды, чтобы убедиться, что будет исполнена только одна микрокоманда с помощью анализа CR.
- Все микропрограммы реализуемых команд должны заканчиваться микрокомандой 80C4101040 (GOTO INT @ C4), осуществляющей переход к циклу прерывания БЭВМ.

- Для реализации микропрограммы необходимо использовать режим командной строки БЭВМ java -Dmode=cli -jar bcomp-ng.jar. Подсказка по командам доступна по инструкции help.
- Для вывода листинга текущей микропрограммы необходимо использовать команду mdecodeall — вывод всех ненулевых ячеек микрокода.

Пример. 0F00 - инверсия содержимого аккумулятора и очистка регистра С. Дополнительного декодирования не требует.

Изменения памяти микрокоманд для команды "COMACLC".

Таблица 4.1

Адрес МП	Микро-команда	Действие ; Комментарии
BB	81F0014002	Исправляемые ячейки интерпретатора if CR(8) = 1 then GOTO F0 ; изменение адреса перехода
F0	0010E09210	Микрокод команды ~AC → AC,N,Z,V,C; Инверсия AC и установка признаков результата
F1	80C4101040	GOTO INT @ C4 ; Переход на цикл прерывания

Таблица трассировки микрокоманд.

Таблица 4.2

MP до выборки MK	Содержимое памяти и регистров процессора после выборки и исполнения микрокоманды								
	MR	IP	CR	AR	DR	BR	AC	NZVC	MP (СчМК)
01	0000000000	00E	0000	000	0000	0000	0000	00000	02

Порядок выполнения работы

- Получить допуск к лабораторной работе, предъявив преподавателю подготовленные материалы.
- Занести разработанные изменения основной микропрограммы и микропрограммы циклов исполнения заданной команды в микропрограммную память базовой ЭВМ, а разработанную тестовую программу в основную память базовой ЭВМ.
- Выполнить в пошаговом режиме тестовые программы, проверив работоспособность синтезированных команд. Заполнить таблицу трассировки цикла исполнения для разработанных микрокоманд по форме таблицы 4.2 для одного варианта выполнения каждой микрокоманды.

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

- Текст синтезированных микропрограмм по форме таблицы 4.1
- Текст тестовых программ на языке Ассемблера БЭВМ (см. Приложение Д).
- Таблицу трассировки циклов исполнения разработанных микрокоманд по форме таблицы 4.2
- Методику проверки команды с использованием тестовой программы.

Контрольные вопросы:

- Микропрограммное устройство ЭВМ, назначение, состав, принцип работы.
- Формат микрокоманд БЭВМ. Для чего существуют два формата команд?
- Исполнение горизонтальных (ОМК, УМК) микрокоманд на примере заданной микрокоманды.
- Структура и принципы функционирования АЛУ.
- Структура и принципы функционирования коммутатора.
- Выполнение операций суммирования и логического умножения, схема сумматора. Инверторы входов, схема инверторов входов.
- Выполнение операций сдвигов, симметричной и несимметричной передачи.
- Принципы построения PS, значение отдельных битов.
- Как организована и выполняется микрокоманда безусловного перехода?
- В какой момент происходит увеличение СчМК?
- Что будет если на вентили SORA и PLS1 одновременно подать единицы?

Рубежный контроль №3

Рубежный контроль №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 №2.

Таблица 4.3

СчМК до выборки МК	Исходные данные: команда 00E SUB (000), код команды: 6800 Содержимое памяти и регистров процессора после выборки и исполнения микрокоманды											
	яч. 000	РМК	СК	РА	РК	РД	А	С	БР	Н	З	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

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. Кириллов В.В. Архитектура базовой ЭВМ — СПб: СпбГУ ИТМО, 2010. - 144с.

Приложение А. Краткий перечень и функциональность команд UNIX

Таблица А.1

Основные команды юникс для изучения на лабораторной работе №1

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 команды в stdout.
cat	cat [-n] [file...] [-] - Слить (concatenate) 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]... Вызвести 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.
head	head [-num] [file...] - Вывести num первых строк из файла.
tail	tail [-/+num] [-bc1] [file...] - Вывести num последних (-) или первых (+) блоков (-b); байт (-c); строк (-l) из файла.
sort	sort [-unr] [-k num] [file...] - Сортировать содержимое по алфавиту, как числа (-n), в обратном порядке (-r), без дублей (-u), по колонке с номером (-k) файлов или стандартного ввода. Результат вывести в стандартный вывод.
grep	grep [-v] regexp [file...] - Оставить в выводе строки, удовлетворяющие регулярному выражению, или вывести все строки, кроме удовлетворяющих (-v) для файлов или стандартного ввода.
wc	wc [-c -m] [-lw] [file...] - Подсчитать количество байт (-c); букв (-m); строк (-l); слов (-w) и вывести на стандартный вывод.

Приложение Б. Накопление баллов БАРС в течении курса

Накопление баллов распределено равномерно в семестре. Удовлетворительная успеваемость подразумевает посещение лекций и практических занятий с получением средних оценок. Точные даты прохождения рубежных и текущих тестирований зависят от конкретных сроков чтения лекции и проведения практических занятий и могут варьироваться. Баллы, выставляемые за выполнение заданий:

- Лабораторные работы оцениваются в разное количество баллов (см. табл. Б.1) баллов, при этом лабораторная работа 7 не является обязательной.
- Текущее тестирование проводится на лекции, и оценивается в 0-5 балла.
- Рубежный контроль может проводится как на лекционных, так и на практических занятиях и приносит от 6 до 10 баллов.
- Личностные качества: 0-3 балла в каждом семестре. Выставляется преподавателем, принимающим лабораторные работы, оценивается качество подготовки и срок сдачи лабораторных работ.
- Зачет 0-40 баллов, Диф.зачет: 12-20 баллов. Сдача зачетов является обязательной.

Таблица Б.1

Распределение баллов

№	Название	Контрольная точка	Баллы, макс.	Баллы, мин.	Неделя сдачи
Осенний семестр (1)					
1	Лабораторная работа 1	Ключевая	10	6	7
2	Лабораторная работа 2	Ключевая	30	18	15
3	Контрольная работа 1	Ключевая	10	6	16
4	Тест 1		5	0	?
5	Тест 2		5	0	?
6	Зачет	Ключевая	40	24	Рассп. ПА
Дополнительные баллы			3	0	Рассп. ПА
Итого			103	Не меньше 60	
Весенний семестр (2)					
1	Лабораторная работа 3	Ключевая	10	6	4
2	Лабораторная работа 4	Ключевая	10	6	6
3	Лабораторная работа 5	Ключевая	10	6	11
4	Лабораторная работа 6	Ключевая	20	12	13
5	Лабораторная работа 7		10	6	16
6	Контрольная работа 1	Ключевая	10	6	16
7	Тест 3		5	0	?
8	Тест 4		5	0	?
9	Дифференциальный Зачет	Ключевая	20	12	Рассп. ПА
Дополнительные баллы			3	0	Рассп. ПА
Итого			103	Не меньше 60	

Приложение В. Состав, структура и функционирование БЭВМ-NG

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. В новую версию БЭВМ-NG вошли некоторые элементы архитектуры PDP-11.

1.2 Структура базовой ЭВМ

На рис. В.1 приведена упрощенная структура базовой ЭВМ. Это одноадресная ЭВМ аккумуляторного типа, выполняющая простейшие операции с 16-разрядными машинными словами.

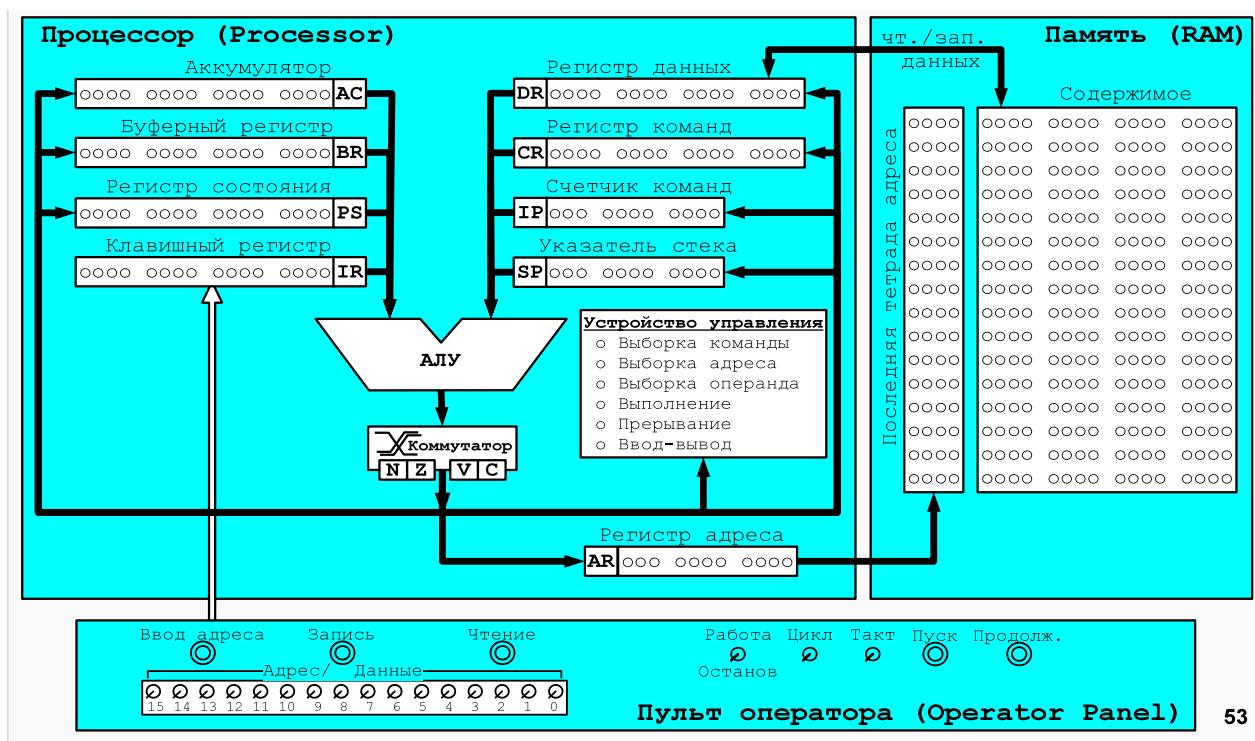


Рисунок В.1 Модель базовой ЭВМ

БЭВМ-NG включает в себя нескольких функциональных блоков и регистров:

Память. Состоит из 2048 ячеек (16-битовых) с адресами 0, 1, ..., 2046, 2047. Ячейки с 0 по 0xF имеют специальное назначение, в них хранятся вектора прерываний, которые будут рассмотрены далее.

Процессор. Состоит из ряда регистров, арифметико-логического устройства с коммутатором и блоком установки признаков результата, а также устройства управления.

Устройство управления (УУ, CU - Control Unit), его также называют микропрограммным устройством (МПУ), выполняет машинные команды процессора при помощи элементарных микроопераций: открытия вентилей и проверкой состояния бита заданного регистра. Работа МПУ разбита на циклы или стадии

исполнения команды и циклы пультовых операций. Подробно МПУ будет рассмотрено далее.

Арифметико-логическое устройство (АЛУ, ALU - Arithmetic-n-Logic Unit) может выполнять арифметические операции, такие как сложение и сложение с учетом переноса, полученного в результате выполнения предыдущей операции, операции логического умножения и инвертирования. Выход из АЛУ подключен к коммутатору.

Коммутатор это устройство, на которое поступают 18 разрядов результата операции из АЛУ (16-ти разрядный результат сложения и биты, необходимые для формирования признака переноса С), а также предыдущее значение переноса из регистра состояния. Коммутатор выполняет операции прямой и перекрестной передачи информации между байтами слова, осуществления арифметических и циклических побитовых сдвигов влево и вправо, а также для расширения знака младшего байта в старший байт. Информация из коммутатора поступает на шину данных для записи в регистры БЭВМ и на блок установки признаков результата.

Блок установки признаков результата предназначен для формирования однобитовых признаков результата, которые в конечном итоге сохраняются в младших 4-х битах регистра состояния. К ним относятся:

Флаг переноса (С - Carry) выступает в качестве продолжения аккумулятора и заполняется при выходе результата за границу 16-ти разрядного слова. При выполнении арифметических операций и операций сдвига в него попадает выход C_{new} коммутатора. Флаг переноса необходимо контролировать при выполнении арифметических операций с беззнаковыми числами.

Флаг переполнения (V — оOverflow) сигнализирует о переполнении разрядной сетки при операциях АЛУ со знаковыми числами, и формируется как операция сложения по модулю 2 поразрядных переносов из 14 в 15 разряд и из 15 в 16 разряд АЛУ, т.е. на выходе коммутатора осуществляется операция $V = C_{14} \oplus C_{new}$.

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

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

Аккумулятор (AC - Accumulator) — 16-ти разрядный регистр, являющийся одним из главных элементов процессоров аккумуляторного типа. Машина может одновременно выполнять операции только с одним или двумя операндами, которые находятся на левом или правом входе АЛУ. Результаты арифметико-логических операций обычно помещаются в АС и изменяются соответствующим образом признаки результата.

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

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

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

Регистр команд (CR - Command Register) — 16-ти разрядный регистр, используемый для хранения кода выполняемой в данный момент команды с целью ее поэтапного декодирования и выполнения требуемых операций.

Буферный регистр (BR - Buffer Register) это 16-ти разрядный регистр, который используется для организации промежуточного хранения данных во время работы.

Указатель стека (SP - Stack Pointer), как и IP и AR 11-ти разрядный, и всегда указывает на вершину стека - особого участка памяти, который предназначен для хранения адресов возвратов и параметров подпрограмм и прерываний.

Клавишный регистр (IR - Input Register) – 16-разрядный регистр, находится в

составе пульта оператора ЭВМ и предназначен для ввода адреса программы, кодов программы и данных. Пульт оператора одержит набор тумблеров и клавиш, позволяющих оператору осуществлять ввод данных в БЭВМ, запуск программы на выполнение и управление режимами работы БЭВМ.

Регистр состояния (PS - Program State) - 16-разрядный регистр, хранит биты управляющие работой БЭВМ (работа, прерывание и пр.) и признаки результата. В актуальной программной реализации используются только 9 младших разрядов.

1.3. Система команд базовой ЭВМ

Классификация команд. БЭВМ способна исполнять точно определенный набор команд. При составлении программы пользователь ограничен этими командами. Полный перечень команд базовой ЭВМ приведен в таблице В.3. В зависимости от особенностей выполнения различных операций в БЭВМ команды можно разделить на четыре группы:

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

Выбор одного из типов команды осуществляется МПУ при помощи анализа старших четырех бит кода команды (биты с 12 по 15), которые называются кодом операции (КОП, Opcode - Operation code). Разработчики БЭВМ выбрали шесть форматов 16-битовых (однословных) команд с 4-битовым кодом операции (рис. В.2).

а) Безадресная команда

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
КОП=0000	Расширение КОП

б) Команда ввода-вывода

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
КОП=0001	Приказ	Устройство

в) Адресная команда с абсолютной адресацией

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
КОП	0	Адрес

г) Адресная команда с относительной адресацией;

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0			
КОП	1	Режим	Смещение

д) Команда с прямой (непосредственной) загрузкой операнда

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0			
КОП	1	111	Число

е) Команда ветвлений

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
КОП=1111	Расш. КОП	Смещение

Рисунок В.2. Форматы команд

Безадресные команды выполняют различные действия без ссылок на ячейку памяти. Например, команда CLA предписывает ЭВМ очистить аккумулятор (записать в АС код нуля). Это команда обработки операнда, расположенного в конкретном месте, "известном" машине. Другой пример безадресной команды - команда HLT. Формат команды состоит из значения 0 в КОП и расширения кода операции (биты 0-11), которое задает необходимую операцию без использования явного указания ячейки памяти. Следует отметить, что безадресные операции могут использовать ячейки памяти неявно, например, команды POP или RET.

Команды ввода-вывода управляют обменом данными между процессором и внешними устройствами ЭВМ. Эти команды будут подробно рассмотрены в части 2.

Таблица В.3

Система команд базовой ЭВМ

Код	Команда	Признаки ¹				Описание
		N	Z	V	C	
0XXX		Безадресные команды				
0000	NOP	-	-	-	-	Нет операции
0100	HLT	-	-	-	-	Останов
0200	CLA	*	*	0	-	0 → AC
0280	NOT	*	*	0	-	(^AC) → AC
0300	CLC	-	-	-	0	0 → C
0380	CMC	-	-	-	*	(^C) → C
0400	ROL	*	*	*	*	AC и C сдвигается влево. AC15 → C, C → AC0
0480	ROR	*	*	*	*	AC и C сдвигается вправо. AC0 → C, C → AC15
0500	ASL	*	*	*	*	AC сдвигается влево. AC15 → C, 0 → AC0
0580	ASR	*	*	*	*	AC сдвигается вправо. AC0 → C, AC15 → AC14
0600	SXTB	*	*	0	-	Расширение знака мл. байта AC7 → AC15...AC8
0680	SWAB	*	*	0	-	Обмен ст. и мл. байта AC7...AC0 ↔ AC15...AC8
0700	INC	*	*	*	*	AC + 1 → AC
0740	DEC	*	*	*	*	AC - 1 → AC
0780	NEG	*	*	*	*	^AC + 1 → AC
0800	POP	*	*	0	-	(SP)+ → AC
0900	POPF	*	*	*	*	(SP)+ → PS
0A00	RET	-	-	-	-	(SP)+ → IP
0B00	IRET	*	*	*	*	(SP)+ → PS, (SP)+ → IP
0C00	PUSH	-	-	-	-	AC → -(SP)
0D00	PUSHF	-	-	-	-	PS → -(SP)
0E00	SWAP	*	*	0	-	Обмен А и вершины стека
1XXX		Команды ввода-вывода				
10XX	DI	-	-	-	-	Запрет прерывания
11XX	EI	-	-	-	-	Разрешение прерываний
12XX	IN REG	-	-	-	-	Чтение из регистров ВУ
13XX	OUT REG	-	-	-	-	Запись в регистры ВУ
18XX	INT NUM	*	*	*	*	Программное прерывание с заданным вектором
XXXX		Адресные команды				
2XXX	AND M	*	*	0	-	M & AC → AC
3XXX	OR M	*	*	0	-	M AC → AC
4XXX	ADD M	*	*	*	*	M + AC → AC
5XXX	ADC M	*	*	*	*	M + AC + C → AC
6XXX	SUB M	*	*	*	*	AC - M → AC
7XXX	CMP M	*	*	*	*	Установить флаги по результату AC - M
8XXX	LOOP M	-	-	-	-	M - 1 → M; Если M <= 0, то IP + 1 → IP
9XXX						Резерв
AXXX	LD M	*	*	0	-	M → AC
BXXX	SWAM M	*	*	0	-	M ↔ AC
CXXX	JUMP M	-	-	-	-	M → IP
DXXX	CALL M	-	-	-	-	SP - 1 → SP, IP → (SP), M → IP
EXXX	ST M	-	-	-	-	AC → M
FXXX		Команды ветвления				
F0XX	BEQ (BZS)	-	-	-	-	Переход если равенство (Z==1)
F1XX	BNE (BZC)	-	-	-	-	Переход если неравенство (Z==0)
F2XX	BMI (BNS)	-	-	-	-	Переход если минус (N==1)
F3XX	BPL (BNC)	-	-	-	-	Переход если плюс (N==0)
F4XX	BHIS (BCS)	-	-	-	-	Переход если выше или равно/перенос (C==1)
F5XX	BLO (BCC)	-	-	-	-	Переход если ниже/нет переноса (C==0)
F6XX	BVS	-	-	-	-	Переход если переполнение (V==1)
F7XX	BVC	-	-	-	-	Переход если нет переполнения (V==0)
F8XX	BLT	-	-	-	-	Переход если меньше (N⊕V==1 / N!=V)
F9XX	BGE	-	-	-	-	Переход если больше или равно (N⊕V==0 / N==V)
FAXX						Резерв

Примечания:

1. Значения в столбцах признаков результатов показывают, как изменится соответствующий признак в результате выполнения операции. «-» - команда не влияет на признак, «0» - признак сбросится, «*» - значение признака установится по результату операции.

Адресные команды предписывают машине производить действия с ячейкой памяти, адрес которой определяется исходя из адресной части команды, состоящей из 12 бит (биты 0..11). КОП (биты 12..15) принимает значения от 0x2 до 0xE и задает операцию.

Команды ветвления позволяют продолжить вычислительный процесс с другого адреса программы в зависимости от состояния признаков результата NZVC.

Режимы адресации в адресных командах. Для адресных команд предусмотрено два различных формата:

1) С прямой абсолютной адресацией (рис. В.2.в) — в бите 11 у этих команд всегда 0, а в адресной части (битах с 0 по 10) записано абсолютное значение адреса операнда (т.е. номер ячейки в адресном пространстве) в памяти. При выполнении операции команда непосредственно обращается по заданному адресу выбирая или записывая operand.

2) С относительной адресацией (рис. В.2.г) — 11-й бит содержит 1, а биты 8-10 режим адресации. В биты 0-7 записано смещение, которое используется для вычисления адреса операнда в памяти с помощью прибавления смещения к значению IP. Смещение может быть и положительным и отрицательным, позволяя адресовать 127 ячеек до и 128 ячеек после текущей команды в памяти. Подчеркнем, что смещение 0 будет указывать на следующую за командой ячейку. Это происходит потому, что к моменту вычисления адреса операнда, счетчик команд уже увеличен на 1 в результате исполнения команды. Режимы адресации могут быть:

- Прямая относительная (код 0xE) или еще ее называют «прямая со смещением относительно IP». Адрес операнда получается сложением закодированного в команде смещения со счетчиком команд.
- Косвенная относительная (0x8). Косвенная адресация подразумевает, что в ячейке памяти, которая вычисляется из адресной части команды через сложение смещения со счетчиком команд, хранится адрес операнда. В результате после вычисления ячейки, где хранится адрес, ее значение снова используется в качестве адреса, вычисляя расположения операнда в памяти.
- Косвенная автоинкрементная (0xA). Эта адресация аналогична случаю косвенной адресации, однако после загрузки операнда из памяти, значение адреса в ячейке памяти увеличивается на 1. Режим обычно удобно использовать для обработки элементов массива, заданных начальным адресом и длиной в порядке возрастания порядкового номера элементов.
- Косвенная автодекрементная (0xB). Эта адресация аналогична случаю косвенной адресации, однако перед загрузкой операнда из памяти, значение адреса в ячейке памяти уменьшается на 1. Режим обычно удобно использовать для обработки элементов массива, заданных начальным адресом и длиной в порядке убывания порядкового номера элементов.
- Со смещением относительно SP (0xC). Адрес операнда получается сложением закодированного в команде смещения с указателем стека. Режим позволяет адресовать параметры с заданным номером, которые находятся в стеке.
- С непосредственной (прямой) загрузкой операнда (0xF) в аккумулятор (рис. В.2д). Для такого формата биты 8-11 установлены в единицы. Команда с режимом адресации «прямая загрузка» по факту не является адресной, а только использует формат адресной команды. Она берет число в битах 0-7 команды в качестве операнда и рассматривает его как знаковое, расширяя знак байта (бит 7) в биты 8-15 старшего байта.

Сводная информация о режимах адресации приведена в табл.В.10.

1.4 Представление данных в БЭВМ

Архитектура фон-Неймана предполагает использование общей памяти для данных и команд. При этом в ячейках памяти просто хранятся числовые значения, а их интерпретация ложится на разработчика программ для ЭВМ. Интерпретация

значения, содержащегося в ячейке памяти называется областью представления данных (ОП). Не зная, как организована программа в памяти, как она использует значения разрядов слова той или иной ячейки, очень сложно интерпретировать ее содержимое. Например, значение ячейки 3021_{16} может быть кодом команды OR 0x21, беззнаковым числом 12321_{10} , двумя символами ASCII «0!», набором логических переменных, чем нибудь другим или не быть ничем, представляя собой случайный набор бит в памяти после включения ЭВМ.

Количество использованных бит памяти, совместно с областью представления, задает область допустимых значений (ОДЗ) для данных, которые мы, как разработчики программы, размещаем в оперативной памяти. ОДЗ применяется в вычислительной технике по аналогии с алгеброй.

В любой ЭВМ в ячейке памяти можно хранить данные различных типов — числа с фиксированной и плавающей точкой, логические переменные, символы и строки символов, и т.д. В БЭВМ ОП и ОДЗ связаны с размером машинного слова, которое, как мы знаем, составляет 16 двоичных разрядов.

1.4.1 Представление чисел в БЭВМ

16-ти разрядное машинное слово БЭВМ может быть интерпретировано как знаковое или беззнаковое.

Если мы зафиксируем двоичную точку числа непосредственно за 0-ым разрядом и используем все 16 разрядов машинного слова для хранения значения числа, задав тем область представления, то беззнаковое представление можно использовать для представления нуля и натуральных чисел, не превышающих 65535. Подобные числа (так же как и рассмотренные ниже двоичные числа со знаком) называются числами с фиксированной точкой, разделяющей целую и дробную части числа. 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.

Целочисленное знаковое представление числа 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. Дополнительный код определен для любой системы счисления, включая, например, десятичную.

Итак, например, 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. Найти поразрядное дополнение для каждой цифры числа (для двоичной системы счисления это аналогично его инверсии, или замене всех 0 на 1, а всех 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 по правилам операций со знаковыми числами в дополнительном коде, 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). Рассмотрим возникновение этих ситуаций на примере представления чисел в четырехразрядной сетке (рис. В.4).



$ \begin{array}{r} 0011 \leftarrow \text{поразрядные} \\ +0011 \quad 3 \text{ переносы} \\ 0001 \quad 1 \\ \hline 00100 \quad 4 \end{array} $ $C=0 \quad OV=0$	$ \begin{array}{r} 1111 \\ +0011 \quad 3 \\ 1111 \quad -1 \\ \hline 10010 \quad 2 \end{array} $ $C=1 \quad OV=0$	$ \begin{array}{r} 1111 \\ +1101 \quad -3 \\ 1111 \quad -1 \\ \hline 11100 \quad -4 \end{array} $ $C=1 \quad OV=0$
$ \begin{array}{r} 1101 \\ +1101 \quad -3 \\ 0101 \quad +5 \\ \hline 10010 \quad 2 \end{array} $ $C=1 \quad OV=0$	$ \begin{array}{r} 1000 \leftarrow \text{биты различны} \\ +1000 \quad -8 \\ 1111 \quad -1 \\ \hline 10111 \quad 7 (-9) \leftarrow \text{ошибка} \end{array} $ $C=1 \quad OV=1$	$ \begin{array}{r} 0111 \leftarrow \text{биты различны} \\ +0111 \quad 7 \\ 0001 \quad 1 \\ \hline 01000 \quad -8 (\neq 8) \leftarrow \text{ошибка} \end{array} $ $C=0 \quad OV=1$

Рисунок В.4 Возникновение переполнения и переноса

Процессор определяет переполнение по следующему правилу: если поразрядные переносы в знаковый и из знакового разряда одновременно отсутствуют или присутствуют - значит переполнения нет, если присутствует только в одном - значит переполнение знаковой разрядной сетки есть. Для приведенного примера знаковый разряд имеет номер 3, для слова БЭВМ — номер 15.

Разрядная сетка слова, с которой БЭВМ выполняет операции, состоит из 16 разрядов. Если необходимо более высокая разрядность числа, то при помощи команды учета переноса при сложении (ADC) возможны операции с 32-х разрядными числами и числами более высокой разрядности. В зависимости от необходимого представления числа программист должен учитывать максимальное и минимальное значение для используемых чисел.

БЭВМ не умеет на уровне машинных команд работать с числами с плавающей точкой. Представление логической и символьной информации не отличаются от современных ЭВМ и выходят за рамки данных методических указаний.

1.5 Операции с памятью и арифметические операции

Как мы уже говорили, БЭВМ относится к процессорам аккумуляторного типа, в которых один operand находится в ячейке памяти, а второй в регистре общего назначения АС. Для осуществления передачи значений между АС и памятью предназначено несколько операций, которые выполняются с учетом выбранного режима адресации.

Загрузка значения для обработки или вычислений из ячейки памяти в аккумулятор производится при помощи команды LD. Сохранение результата производится командой ST. При необходимости, можно выполнить обмен содержимого ячейки памяти и аккумулятора при помощи команды SWAB.

Обмен старшего и младшего байтов 16-ти разрядного (двухбайтового) АС между собой бывает необходим в программах, работающих, например, со строками. Для этого предназначена команда SWAB.

Сложение целых двоичных чисел со знаком и без знака выполняется в базовой ЭВМ с помощью команды ADD. Для учета переноса из младших слов многословных (32 разряда и более) чисел необходимо использовать команду ADC.

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 АС, то в признак переноса заносится 1, в противном случае в него заносится 0, аналогичным образом обрабатывается переполнение. Исторически, в процессорах команды INC и DEC реализовывались быстрее, чем сложение с единицей или ее вычитание. В современных процессорах эта разница практически отсутствует за счет архитектурных оптимизаций.

Изменение знака числа производится при помощи команды NEG, которая является удобным сокращением от NOT+INC.

Расширение знака осуществляется при помощи команды SXTB. Она используется в случаях, когда в АС записано знаковое 8-ми разрядное число, и его для последующей арифметической операции в АЛУ, необходимо превратить в знаковое 16-ти разрядное знаковое число. Для этого разряд с номером 7 операнда команды (находится в АС) копируется в разряды с 8 по 15 результата (тоже в АС).

Для выполнения вычитания (X-Y) в базовой ЭВМ предусмотрена команда SUB Y. Например, вычитание X-Y реализуется командами LD X, SUB Y. Команда вычитания с заемом (по аналогии с ADC) в БЭВМ не реализована.

Сравнение чисел выполняется командой СМР аналогично вычитанию, однако результат вычитания АС-М в АС не сохраняется, по нему лишь устанавливаются признаки результатов NZVC.

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

Команды ASR и ASL (арифметические сдвиги вправо и влево) осуществляют деление и умножение на 2. Их последовательным использованием можно организовать умножение и деление на число, которое представлено степенью 2,

например на 2, 4, 8, 16...

1.6 Сдвиги и логические операции

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

Логическое умножение (команда AND) выполняет над каждым разрядом содержимого аккумулятора и содержимым ячейки памяти, заданной адресной частью команды, 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.

Логическое сложение (команда OR) выполняет над каждым разрядом содержимого аккумулятора и содержимым ячейки памяти, заданной адресной частью команды, операцию логического сложения («ИЛИ»).

Циклический сдвиг влево и вправо на один разряд (команды ROL и ROR) замыкают аккумулятор и регистр переноса в кольцо и сдвигают все биты кольца на один разряд влево или вправо (рис. В.5).

Очистка аккумулятора производится при помощи команды CLA. Побитная инверсия или отрицание содержимого аккумулятора производится при помощи команды NOT. Для нее предусмотрен синоним CMA. Очистка и инверсия флага переноса С производится командами CLC и CMС.

Флаг С	Аккумулятор
До сдвига	0 1011100000101011
После сдвига влево	1 011100001010110
После сдвига вправо	1 0101110000010101

Рисунок В.5. Циклические сдвиги: а - влево, б - вправо

1.7 Управление вычислительным процессом

Задача управления вычислительным процессом решается в языках высокого уровня операторами проверки условия, организацией циклических повторений, условным оператором и т. д. При этом, в зависимости от проверки условия, производится выполнение той или иной последовательности операторов, что получило название условное выполнение.

В базовой ЭВМ, как и в любой другой реализации процессора, условное выполнение состоит из установки признаков результатов и изменения счетчика команд в зависимости от состояния признаков.

Установка признаков результата происходит в блоке установки признаков результата, данные в который попадают из коммутатора. Анализ результата выполнения каждой машинной команды может изменять признаки, устанавливая или сбрасывая их в зависимости от получившегося числа. Некоторые команды оставляют признаки неизменными.

Например, команда CLA оставляет неизменным признак переноса С, сбрасывает признак переполнения V (какое переполнение может быть при сбросе значения АС в 0?), а признаки N и Z устанавливаются по результатам операции, что для операции очистки АС эквивалентно установке N в 0 (т.к. в 15 разряде для знакового представления будет 0), а Z в 1. Подробная информация о поведении каждой команды относительно изменения признаков результата после ее выполнения приведена в табл. В.3.

Анализ установленных признаков результата к текущему моменту и изменение счетчика команд в зависимости от результатов анализа происходит при помощи команд ветвлений.

Команды ветвлений позволяют организовать нелинейное исполнение программ. Эти команды при выполнении заданного признаками NZVC условия

осуществляют переход по указанному адресу (запись в IP нового адреса выполнения в программе). Если условие не выполняется, то происходит выполнение команды, которая следует за командой ветвления. КОП команд ветвления 1111₂. Операция проверки признаков результата задается в формате команды в битах 8-11 расширения КОП (см. рис. В.2e), а смещение (биты 0..7) указывается относительно текущего (увеличенного на 1 после цикла выборки команды) значения IP, при этом сумма текущего значения IP и смещения вычисляет абсолютный адрес для перехода в адресном пространстве БЭВМ.

Команды ветвления не изменяют состояния аккумулятора и признаки результата. Они могут лишь изменить содержимое счетчика команд, поместив в него адрес, определяемый значением IP и смещением.

Равенство или неравенство двух чисел можно проверить, если вычесть одно из другого (можно, также, применить команду сравнения CMP: LD Y \n CMP X). Если X-Y==0 то установится признак результата Z. Соответственно, команды ветвления BEQ и BNE осуществляют переход в случае равенства или неравенства заданных X и Y. Таким же образом проверяется равенство X==0. Например, мы хотим проверить результат операции логического умножения на равенство 0. Команда AND устанавливает признак NZ результата в зависимости от значения результата, поэтому нам будет достаточно выполнить последовательно команды AND и BEQ. В случае, если команда не устанавливает необходимые признаки результата (пример - IN), для их установки можно использовать команду CMP #0.

Сравнение величин чисел можно осуществлять по различным условиям. Следует различать использование ветвлений по признакам, которые подходят для знаковых или беззнаковых чисел, а также те, которые применимы для обоих типов чисел. Для беззнаковых величин, после установки признаков (например, так: LD Y, CMP X), можно проверить командами BLO (переход, если X меньше по значению Y), BHIS (переход если X больше или совпадает с Y). Эти команды используют для решения о переходе значение флага перенос и эквивалентны BCC и BCS. Для знаковых величин подходят команды BLT (переход, если X < Y) и BGE (переход, если X >= Y), эти команды используют значения признаков N и V. Для знаковых также используются команды BMI, BPL, BVS, BVC (см. табл. В.3).

Команды BCS и BCC удобны тогда, когда производится побитовый анализ числа. Например, производится сдвиг ASR и нулевой бит выдвигается в С. Командой BCS можно удобно проверить, что бит был установлен.

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. К ним относятся автоинкрементные и автодекрементные режимы адресации, а также средства для организации циклических повторений.

Циклы со счетчиком в БЭВМ организуются при помощи команды "Декремент и пропуск" (LOOP). Данная команда уменьшает заданную ячейку памяти, определяющую число повторений, и проверяет, что в ячейке еще находится положительное число. Если оно действительно положительное, то выполняется следующая после LOOP команда, если число отрицательно или равно нулю, то к счетчику команд добавляется единица, и следующей будет выполнена команда через одну после LOOP.

Для прекращения работы программы и переход в режим, когда оператор может взаимодействовать с БЭВМ при помощи пульта, используется команда "Останов" (HLT). Текущая реализация этой команды закрывает вентиль, разрешающий прохождению импульсов тактового генератора на схему МПУ, тем самым останавливая выполнение программы. В современных ЭВМ такой способ не применяется.

1.8 Подпрограммы и стек

Достаточно часто встречаются ситуации, когда отдельные части программы должны выполнить одни и те же действия по обработке данных (например, вычисление 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 CALL (вызов подпрограммы). На рис. В.6 показана часть основной программы, содержащая две команды CALL 300, с помощью которых осуществляется переход к выполнению команд подпрограммы.

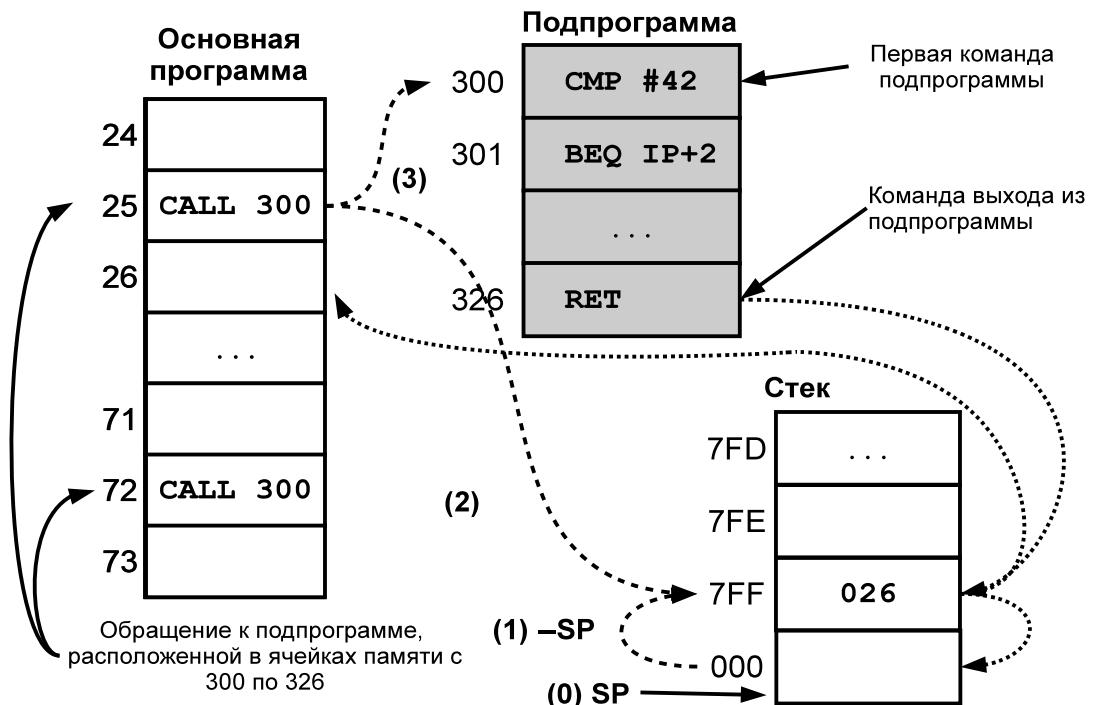


Рисунок В.6. Обращение к подпрограмме и возврат из нее

Подпрограммы осуществляют запись адресов возврата в структуру данных, которая называется стек. Стек работает по принципу «первым вошел — последним вышел» и его можно представить, как стопку книг, где вам видна только верхняя книга. В базовой ЭВМ стек организован при помощи специального регистра — указателя стека (SP), который изначально сброшен в значение 0, и так как SP 11-ти разрядный, то первый элемент стека имеет адрес 7FF, следующий 7FE, т.е. растет от старших адресов к младшим.

По команде `CALL 300`, расположенной в ячейке 25 (см. рис В.6), выполняется запись адреса возврата $25 + 1 = 26$ (значение счетчика команд после цикла выборки команды) на вершину стека. Для этого сначала уменьшается указатель стека на 1, и производится запись в ячейку памяти, на которую указывает SP. Если стек пуст, как в нашем случае, то это ячейка 7FF. Для перехода к подпрограмме производится запись аргумента команды `CALL` (прямого абсолютного адреса 300) в счетчик команд (адрес первой команды подпрограммы).

Далее начинается процесс выполнения команд подпрограммы, который завершается командой `RET`, расположенной в ячейке 326. Эта команда выхода из подпрограммы предписывает ЭВМ выполнить переход к команде, расположенной по адресу, сохраненному в текущей ячейке указателя стека (7FF). 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 `CALL 300`, расположенная в ячейке 72 (после выполнения команд подпрограммы будет выполнен переход к ячейке 73).

Положить на стек значение из аккумулятора в БЭВМ можно при помощи команды PUSH, которая сначала уменьшает текущее значение указателя стека на 1, а потом по адресу, получившемуся в SP записывает значение АС.

Операция снятия со стека POP действует противоположным образом - сначала по адресу, содержащемуся в SP читает значение из памяти, а потом увеличивает значение $SP=SP+1$. Для удобства программиста в БЭВМ определена операция обмена вершины стека с АС — SWAP.

В стеке могут размещаться не только адрес возврата из подпрограммы, но и передаваемые подпрограмме параметры и возвращаемый результат. Перед вызовом подпрограммы по команде CALL в стеке размещается заданное количество параметров, которые помещаются в стек командами PUSH. После возврата из подпрограммы стек должен быть приведен в исходное состояние, из него должны быть прочитаны результаты подпрограммы, а указатель стека должен быть возвращен в состояние до первой команды PUSH. В БЭВМ для этого предназначена команда POP.

Доступ к переданным в стеке параметрам и формирование результатов в БЭВМ из подпрограммы производится при помощи специального режима адресации — со смещением относительно SP. При этом загрузка самого верхнего параметра (который мы положили самым последним перед вызовом CALL) будет иметь смещение 1. Смещение 0 указывает на адрес возврата из подпрограммы, который в момент входа в подпрограмму лежит на вершине стека. Следует заметить, что передавать параметры подпрограмме и получать результаты можно не только через стек, а и через регистры общего назначения. В БЭВМ такой регистр один - АС.

1.9 Выполнение машинных команд

В процессе исполнения команд устройство управления (напомним, еще его называют микропрограммным устройством управления - МПУ) БЭВМ производит пересылку команд, operandов и промежуточных результатов из одного регистра ЭВМ в другой, осуществляет анализ отдельных частей команды (кода операции и режима адресации), а также управляет АЛУ, считывает/записывает содержимое памяти или регистров устройства ввода-вывода.

Эти действия называются микрооперациями, они происходят в заданной последовательности и скоординированы между собой импульсами генератора тактовых импульсов. Устройство управления хранит в себе последовательность действий для исполнения команд и выполнения пультовых операций, называемых циклами. Набор циклов, составляющих код программы МПУ, называется микрокодом или микропрограммой.

Цикл команды

1. Цикл выборки команды (Instruction Fetch, IF)
2. Цикл выборки адреса (Address Fetch, AF)
3. Цикл выборки операнда (Operand Fetch, OF)
4. Цикл исполнения (Execution, EX)
5. Цикл прерывания (Interruption, INT)

Циклы пультовых операций

- Ввод адреса (Set Instruction Pointer, SIP)
- Чтение (Read, RD)
- Запись (Write, WR)
- Пуск (Start, ST)

Рисунок В.7 Циклы устройства управления

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

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

"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.

"Чтение" считывает в регистр данных содержимое ячейки памяти, адрес которой указан в счетчике команд, после чего увеличивает на единицу содержимое счетчика команд.

"Пуск" сбрасывает содержимое регистров DR, CR, SP, AC, BR, AR, сбрасывает признаки результата, запрещает прерывания и, если установлен режим "РАБОТА", переходит к выполнению команды, адрес которой указан в счетчике команд.

На панели оператора расположены и другие органы управления — переключатель "РАБОТА/ОСТАНОВ", который вызывает останов программы после каждой команды; переключатель "ТАКТ", который может выполнить микрокод по одному такту, кнопка «Продолжение» - возобновляющая работу остановленной БЭВМ.

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, выборки операнда, исполнения и прерывания (рис. В.7). Основные действия, выполняемые ЭВМ во время каждого из машинных циклов, проиллюстрированы и описаны ниже на примере команды ADD.

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

1) Исходное состояние памяти представлено на рис. В.8а. В БЭВМ загружена программа, выполняющая вычисление $Z = -X + Y$:

020 Значение $X = -5316 = FFAD_{16}$

021 Значение $Y = 106_{16}$

022 Место хранения для переменной Z (обнулено)

023 CLA - обнуление аккумулятора

024 SUB 20 - содержимое ячейки 20 вычитается из AC, в нем будет $-X$

025 ADD 21 - Добавление ячейки 21 к AC. Т.е. AC сумма $-20 + 21$ ячеек.

026 ST 22 - запись значения суммы в ячейку 22

027 HLT - останов программы

Счетчик команд (IP), как мы помним, всегда содержит адрес следующей исполняемой команды. Перед началом цикла выборки команды в IP содержится значение 025, значит в настоящий момент выполнение программы будет производится с этого адреса.

2) Цикл выборки команды начинается с выбора кода команды (рис. В.8б). По фронту (на этой и последующей картинках изображен красным, сплошным) импульса тактового генератора содержимое счетчика команд (11 разрядов, 5 старших разрядов будут установлены в 0) попадает на правый вход АЛУ. На левый вход АЛУ ничего не подается (т. е. все разряды установлены в значение 0). АЛУ сложит (операция по умолчанию в АЛУ, если не установлены другие вентили операций АЛУ) 0 с 25 и подаст на коммутатор. Коммутатор в данном такте пропустит данные неизменными.

По спаду (изображен зеленым, прерывистым) сигнала тактового генератора, выход коммутатора будет записан в 16-ти разрядный буферный регистр и, одновременно с этим, с выхода коммутатора 11 младших разрядов по шине передастся в 11-ти разрядный регистр адреса. Таким образом в AR появится адрес исполняемой в настоящий момент инструкции.

3) В следующем такте (рис. В.8в) По содержимому регистра адреса (AR)

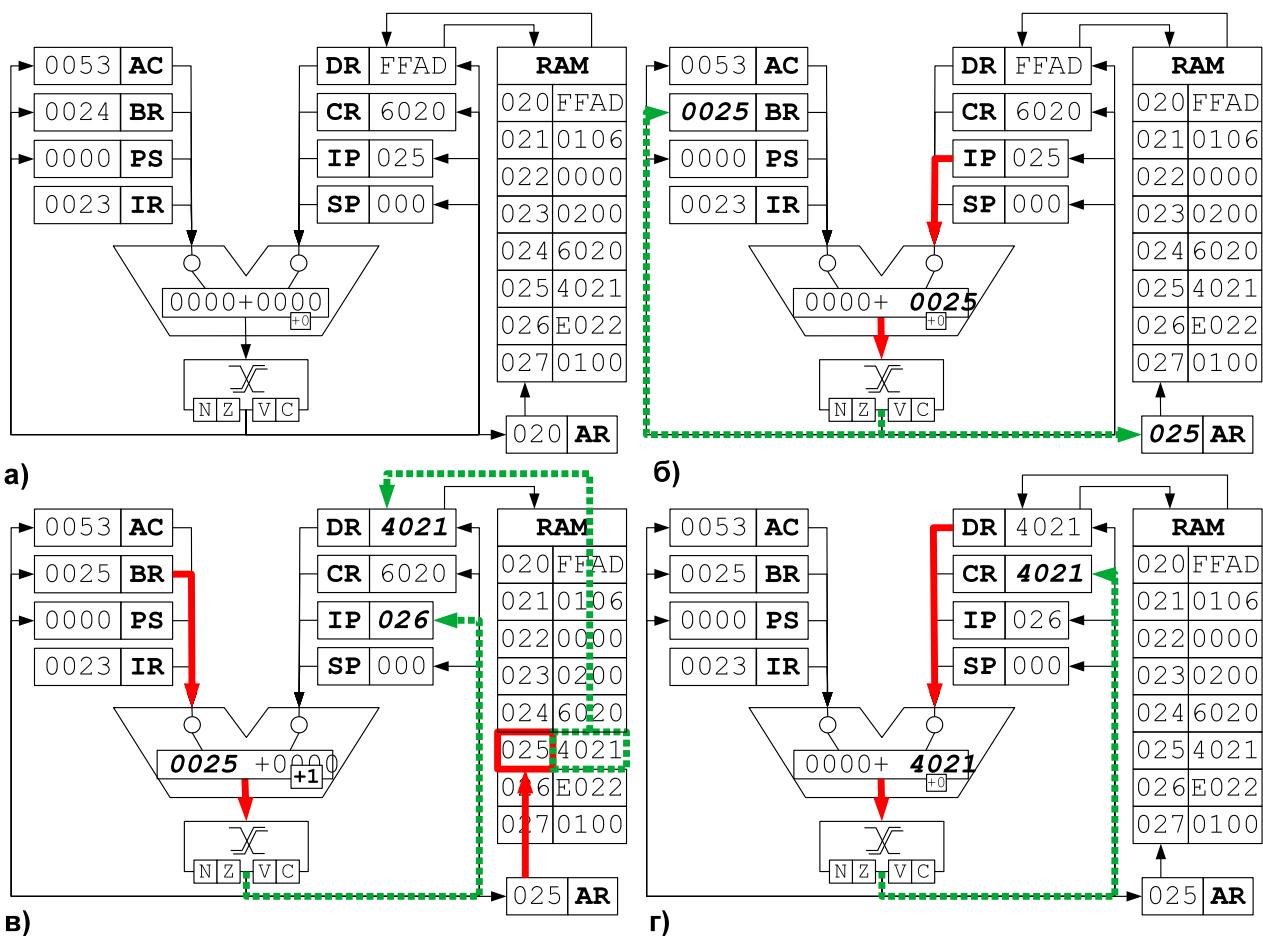


Рисунок В.8 Цикл выборки команды

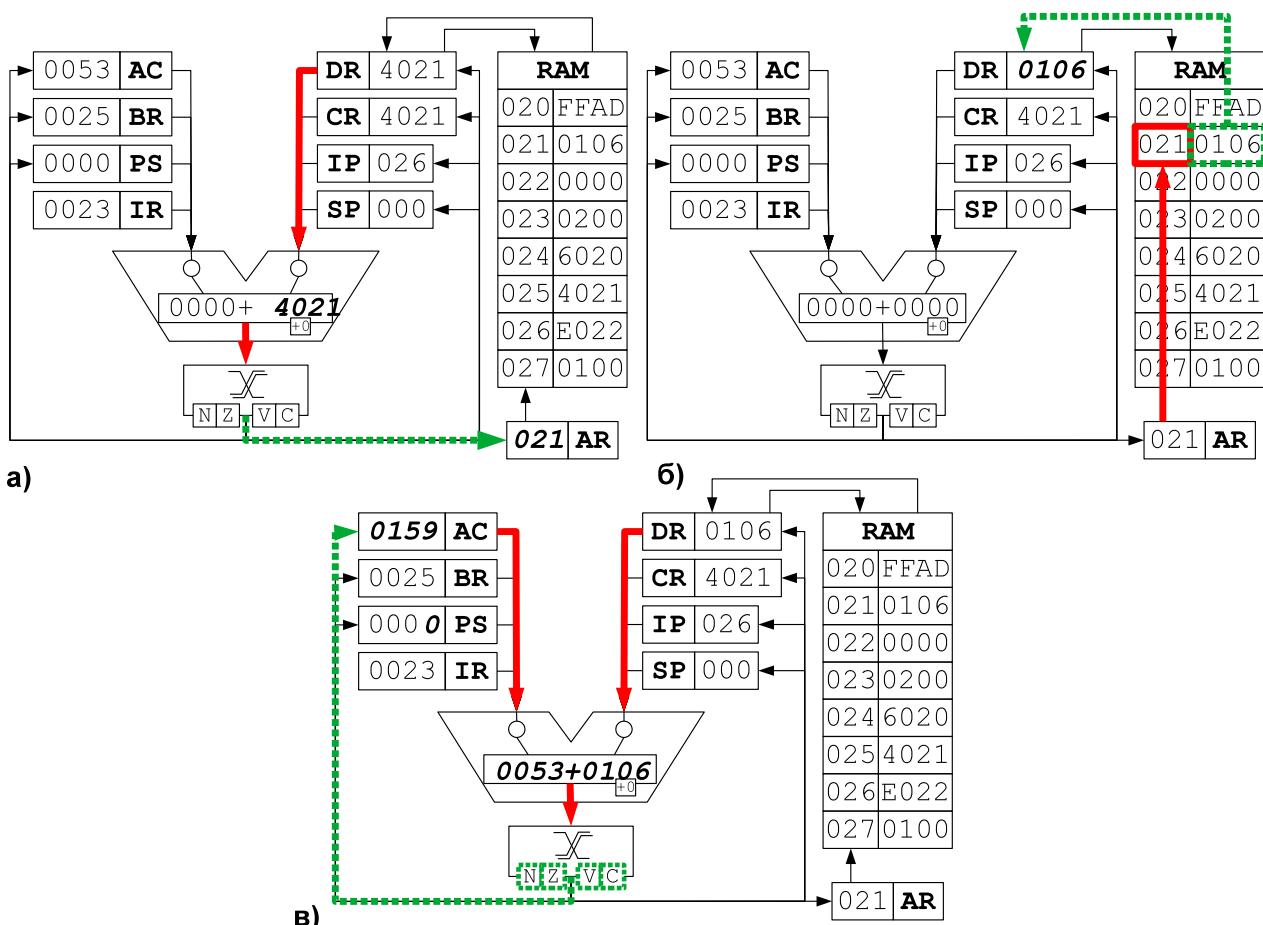


Рисунок В.9 Цикл выборки операнда (а, б) и цикл исполнения (в)

содержимое 25 ячейки выбирается из памяти в регистр данных (DR). Т.к. при обращении к памяти левая часть схемы остается незадействованной, то возможны одновременные с обращением к памяти другие операции с АЛУ и регистрами. Соответственно содержимое BR подается на левый вход АЛУ, правый вход АЛУ при этом закрыт, и содержимое BR (равное IP) увеличивается на 1 в АЛУ, результат этой операции попадает в IP. Счетчик команд теперь содержит адрес следующей исполняемой команды.

4) Для завершения цикла выборки команды (рис. В.8г) необходимо переслать код команды, на предыдущих шагах выбранный из памяти в регистр команд. Для этого содержимое регистра данных через правый вход АЛУ по фронту передается в коммутатор, а по спаду выход коммутатора записывается в регистр команд (CR). Теперь CR содержит код исполняемой команды для его дальнейшего декодирования и определения типа исполняемой команды и режимов адресации.

Цикл выборки адреса в адресной команде с абсолютной адресацией отсутствует, т.к. полностью сформированный адрес непосредственно записан в коде команды. БЭВМ переходит к следующему циклу.

Цикл выборки операнда. Для команды ADD 21 БЭВМ должна выбрать содержимое ячейки памяти с адресом 21 в DR, чтобы на следующем машинном цикле исполнения сложить содержимое регистра данных с аккумулятором. Напомним, что перед циклом выборки операнда код команды находится в регистре данных. По тактам происходит следующее:

1) Содержимое DR подается на правый вход АЛУ (рис. В.9а). Младшие 11 разрядов выхода АЛУ и коммутатора передаются в регистр адреса. На левый вход АЛУ в это время подается 0.

2) По адресу 021 в AR из памяти данные загружаются в DR (рис. В.9б).

Циклы выборки адреса и выборки операнда подробно описаны для каждого вида адресации на рис. В.10.

Код					Мнемоника	Описание	Реализация машинных циклов Address Fetch, Operand Fetch
11	10	9	8				
0	M	M	M	ADD 0ADDR ADD \$L	Прямая абсолютная		$DR \rightarrow AR; MEM(AR) \rightarrow DR$
1	0	0	0	ADD (L)	Косвенная относительная		$SXT_CR(0..7) \rightarrow BR,$ $BR + IP \rightarrow AR, MEM(AR) \rightarrow DR,$ $DR \rightarrow AR; MEM(AR) \rightarrow DR$
1	0	0	1		Резерв		
1	0	1	0	ADD (L)+	Косвенная автоинкрементная (постинкремент)		$SXT_CR(0..7) \rightarrow BR,$ $BR + IP \rightarrow AR, MEM(AR) \rightarrow DR, DR + 1 \rightarrow DR,$ $DR \rightarrow MEM(AR), DR - 1 \rightarrow DR,$ $DR \rightarrow AR; MEM(AR) \rightarrow DR$
1	0	1	1	ADD -(L)	Косвенная автодекрементная (прединкремент)		$SXT_CR(0..7) \rightarrow BR,$ $BR + IP \rightarrow AR, MEM(AR) \rightarrow DR, DR - 1 \rightarrow DR,$ $DR \rightarrow MEM(AR),$ $DR \rightarrow AR; MEM(AR) \rightarrow DR$
1	1	0	0	ADD &N ADD (SP+N)	Косвенная относительная, со смещением (SP)		$SXT_CR(0..7) \rightarrow BR,$ $BR + SP \rightarrow DR,$ $DR \rightarrow AR; MEM(AR) \rightarrow DR$
1	1	0	1		Резерв		
1	1	1	0	ADD L ADD (IP+N)	Прямая относительная		$SXT_CR(0..7) \rightarrow BR,$ $BR + IP \rightarrow DR,$ $DR \rightarrow AR; MEM(AR) \rightarrow DR$
1	1	1	1	ADD #N	Прямая загрузка		$SXT_CR(0..7) \rightarrow BR, BR \rightarrow DR$

Рисунок В.10 Циклы выборки адреса и операнда для различных режимов адресации

Исполнение. Последовательность действий, выполняемых в этом машинном цикле, определяется самой выполняемой командой. Для подробного изучения цикла исполнения каждой микрокоманды необходимо обратиться к микрокоду БЭВМ (табл. В.21).

Цикл исполнения для команды ADD 21 умещается в один такт (рис. В.9в): Содержимое DR подается на левый вход АЛУ, а содержимое аккумулятора на правый вход АЛУ и производит операцию сложения. Коммутатор передаст результат сложения на блок проверок неизменным.

Далее выход коммутатора (результат сложения) записывается в АС, устанавливая при этом признаки выполнения операции. Т.к. при сложении переноса и переполнения не было, то в С и V запишется 0, в признак отрицательного числа (N) запишется 0, т.к. результат сложения получился положительным, и признак Z установится так же значение 0, т.к. результат не равен нулю.

Команда выполнена, далее МПУ производит проверку на необходимость выполнить "ОСТАНОВ" и цикл прерывания, и управление передается на начало микрокода.

Машинный цикл прерывания будет рассмотрен в разделе 2.4. приложения В.

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

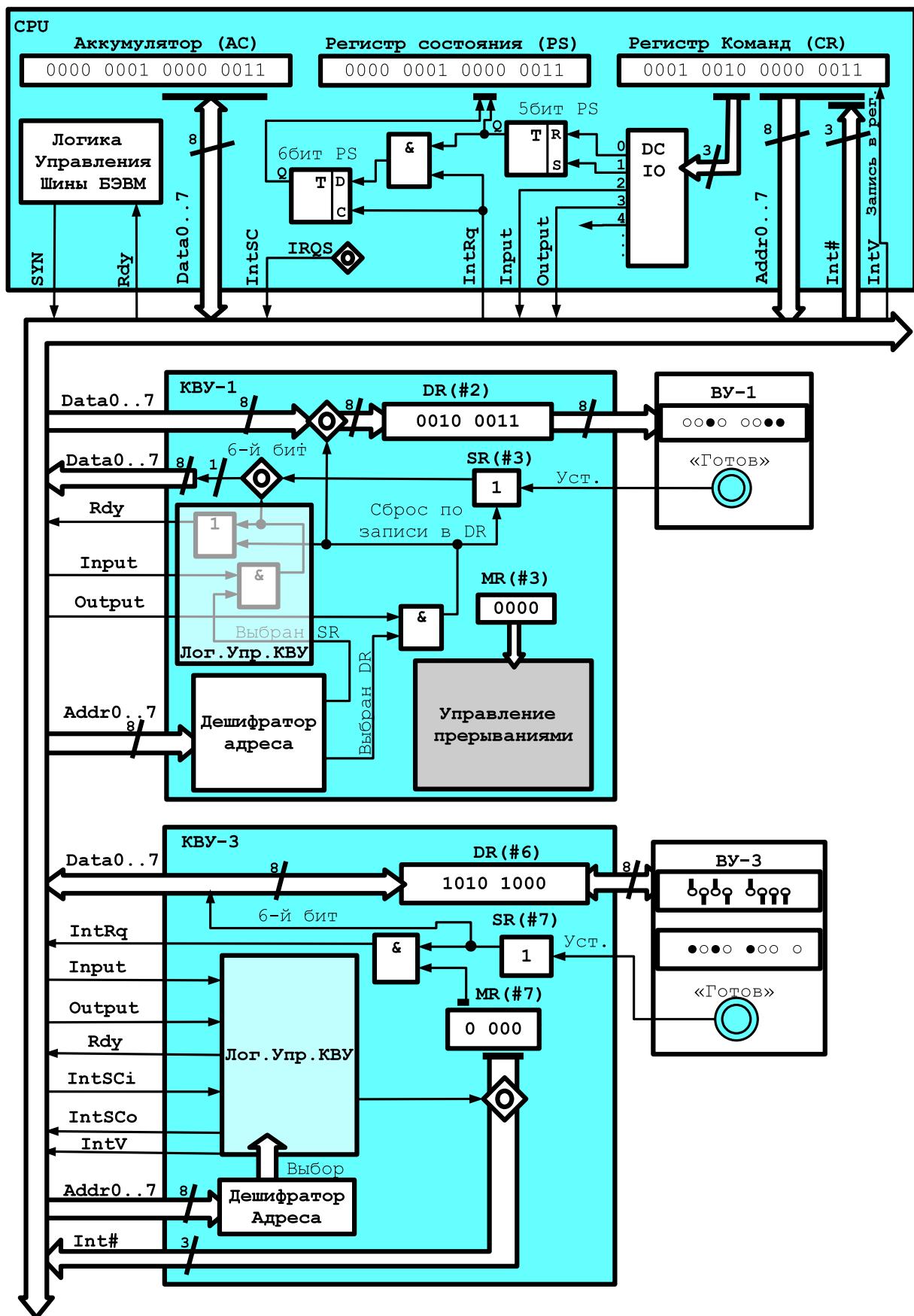


Рисунок В.11 Подсистема ввода-вывода базовой ЭВМ (показаны КВУ и ВУ 1,3)

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 последней версии БЭВМ реализован таймер (устройство ВУ-0), которое вызывает прерывание через заданное в его DR число секунд; устройство ввода-вывода ВУ-4, аналогичное с ВУ-3, за исключением того, что используются отдельные регистры для входных и выходных данных; текстовый принтер ВУ-5; бегущая строка ВУ-6; 8-ми разрядный 7-сегментный индикатор ВУ-7; клавиатура ВУ-8 и цифровая клавиатура ВУ-9.

Между ВУ и процессором включены простейшие контроллеры внешнего устройства (КВУ), каждый из которых содержит:

- десифратор адреса, позволяющий выделить обращение к данному ВУ среди всех обращений к устройствам ввода-вывода, подключенных к процессору;
- логику управления КВУ, набор логических схем, позволяющий реагировать и формировать сигналы шины БЭВМ. Данные схемы не представлены подробно для КВУ-3, в качестве примера реализации необходимо обратится к схеме КВУ-1 на рис. В.11;
- регистр данных (DR - Data Register), через который происходит обмен данными между процессором и внешним устройством;
- регистр состояния (SR - State Register), в котором хранится информация о готовности ВУ к обмену данными с процессором. В контроллерах простейших ВУ используются однобитовые регистры готовности, которые часто называют флагом.
- регистр управления (MR - Management Register), регистр, в котором используются 4 младших разряда, разряд 3 - для разрешения прерывания от контроллера и разряды с 0 по 2, которые содержат номер вектора прерывания. Если прерывания разрешены командой EI и установлено разрешение прерывания от контроллера (разряд 3), то контроллер будет генерировать сигнал IntRq и выставлять номер вектора прерывания на шину адреса.

Контроллеры ВУ связаны с процессором при помощи системной шины БЭВМ, в сегменты (или, физически, разъемы для установки контроллеров) которой подсоединяются различные шины со стороны процессора и контроллера:

- шина данных (Data0..7), по которым происходит передача данных в процессор или из процессора;
- шина адреса (Addr0..7), по которой передается адрес внешнего устройства от процессора к КВУ и номер вектора запроса на прерывание (Int#) от КВУ к процессору, подтверждаемый сигналом выдачи вектора (IntV);
- сигнал запроса прерывания (IntRq), по которому выставляется требование прерывания от внешнего устройства;
- сигнал ввода (Input) - для передачи приказа на ввод (IN #reg);
- сигнал вывода (Output) - для передачи приказа на вывод (OUT #reg);
- начальный сигнал предоставления прерывания (IntSC), который управляется программно микрокодом в цикле прерывания по микрокоманде INTS.
- входящий цепочный сигнал предоставления прерывания (IntSCI# - Interrupt Supply Chain input), по которому контроллер, в соответствии с порядком подключения к шине, проверяет возможность предоставления ему прерывания в соответствии с очередью подключения, и генерирует прерывание, если они разрешены глобально командой EI и в регистре MR контроллера;
- исходящий цепочный сигнал предоставления прерывания (IntSCo# - Interrupt Supply Chain output), который передает контроллер по шине далее следующему КВУ, если нет необходимости вызвать прерывание вычислительного процесса;
- сигнал готовности (Rdy), подтверждающий завершение операции ввода-

вывода внутри цикла обмена между АС и DR соответствующего КВУ. В случае операции ввода Rdy подтверждает данные, передаваемы по шине данных, и в обоих случаях операции ввода-вывода сигнализирует о том, что цикл обмена с регистром данных DR контроллера завершен;

- сигнал синхронизации (Syn) от тактового генератора БЭВМ, который задает временной слот единичного обмена по шине БЭВМ.

Со стороны процессора к системнойшине БЭВМ подключены:

- десифратор приказа (DC IO), который преобразует приказ в КОП команды ввода-вывода в набор управляющих сигналов нашине. 0-й выход десифратора активен после команды DI; 1 - для EI; 2 - для команды IN, формируя нашине БЭВМ сигнал Input; 3 — для команды OUT и сигнала Output. Выходы, начиная с 4-го не используются;
- регистр разрешения прерывания, который входит в состав PS 5-ым разрядом, и показывает глобальный статус разрешений прерывания в процессоре;
- регистр прерывания от КВУ, по которому выполняется цикл прерывания, если прерывания разрешены глобально и в настоящее время нашине БЭВМ есть запрос на прерывание от одного из КВУ. Данный регистр входит в состав PS в 6-ым разрядом;
- логика управления шины БЭВМ предназначена для подключения и отключения приемо-передатчиков сигналов КВУ и процессора для осуществления обмена CPU с одним из контроллеров в один момент времени.

Назначение регистров контроллеров внешних устройств представлены в таблице В.12. R - означает, что регистр доступен только для чтения, W - только для записи. R/W - доступен для обоих операций. Направление обмена закодированы мнемоникой Data In для ввода данных из КВУ, Data Out - для вывода данных в КВУ, а Data i/o - для двунаправленного обмена.

Таблица В.12
Назначение регистров нашине ввода-вывода БЭВМ.

Адрес рег. КВУ	Наименование ВУ	Наименование регистра КВУ	Адрес рег. КВУ	Наименование ВУ	Наименование регистра КВУ
#0	ВУ-0 Таймер	DR (R/W)	#10	ВУ-6 Бегущая строка	DR (R/W) Data out
#1		MR	#11		
#2	ВУ-1 У-во вывода	DR (W)	#12	ВУ-7 7-ми сегментный индикатор	SR (R/W)
#3		SR (R) MR (W)	#13		MR (R/W)
#4	ВУ-2 У-во ввода	DR (R)	#14	ВУ-8 Клавиатура	DR (R/W) Data out
#5		SR (R) MR (W)	#15		
#6	ВУ-3 У-во ввода-вывода	DR (R/W) Data i/o	#16	ВУ-9 Цифровая клавиатура	SR (R/W)
#7		SR (R) MR (W)	#17		MR (R/W)
#8	ВУ-4 У-во ввода-вывода с регистрами, доступным RW	DR (R/W) Data in	#18	ВУ-9 Цифровая клавиатура	DR (R/W) Data in
#9		DR (R/W) Data out	#19		
#A		SR (R/W)	#1A		SR (R/W)
#B		MR (R/W)	#1B		MR (R/W)
#C	ВУ-5 Текстовый принтер	DR (R/W) Data in	#1C	ВУ-9 Цифровая клавиатура	DR (R/W) Data in
#D			#1D		
#E		SR (R/W)	#1E		SR (R/W)
#F		MR (R/W)	#1F		MR (R/W)

2.2 Команды ввода-вывода

Команды ввода-вывода приведены на рисунке В.13. Указан формат команды, в котором код операции представлен значением 0x1 в разрядах с 12 по 15. В разрядах 8..11 располагается приказ на ввод-вывод, его три младших разряда декодируются аппаратно на дешифраторе приказов (DC IO), а биты с 0 по 7 кодируют адрес регистра контроллера ввода вывода, с которым осуществляется обмен. К командам добавлена операция возврата из прерывания IRET, которая осуществляет восстановление из стека значений регистра состояния и счетчика команд.

Команда DI запрещает прерывания, помещая в 5 бит регистра состояния '0'.

Команда EI разрешает прерывания, помещая в 5 бит регистра состояния '1'.

Команда IN #reg осуществляет чтение из регистра ВУ по адресу в аккумулятор.

Команда OUT #reg осуществляет запись из аккумулятора по адресу в регистр ВУ.

Команда INT #num вызывает программное прерывание с вектором num.

Команда IRET возврат из программы обработки прерывания.

Формат команд ввода-вывода

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	Приказ				Регистр устройства							

Команды, связанные с вводом-выводом

Наименование	Мнемоника	Код	Описание
Запрет прерываний	DI	1000	
Разрешение прерываний	EI	1100	
Ввод	IN #reg	12XX	REG → младший байт АС
Выход	OUT #reg	13XX	младший байт АС → REG
Прерывание	INT #num	18XX	Программное прерывание с вектором num
Возврат из прерывания	IRET	0B00	(SP)+ → PS, (SP)+ → IP

Рисунок В.13 Ввода-вывод: Команды и формат.

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 в 6-м разряде регистре SR.

Пример. С помощью устройства ввода ВУ-2 (DR#4, SR#5) записать в ячейку в памяти коды символов слова "ДА" в кодировке KOI8-R. Пример программы представлен в табл. В.14.

В начале программа висит в ожидании готовности: считывается статусный (с номером 5) регистр ВУ-2, который передается через шину данных в 6 разряде числа, сравнивается с 0x40 (01000000₂), и пока устройство не готово (SR равен 0). Такие циклы с неопределенным временем завершения, называются циклами "spin-loop", они постоянно проверяют готовность устройства или переменной в программе, загружая процессор. Как только 6 разряд SR принимает значение 1 (устройство готово), в регистр данных ВУ считывается в младшие 8 разрядов аккумулятора, старшие разряды АС при этом остаются не изменными. Для того, чтобы подготовить АС к приему второго символа, происходит обмен байтов аккумулятора при помощи

команды SWAB, и сохранение этого символа в старших разрядах в ячейку результата.

Далее снова происходит ожидание готовности устройства, после получения готовности загружается ячейка с результатом и считывается 2 символ в младшую часть аккумулятора. Готовое слово с двумя символами в аккумуляторе сохраняется в ячейку результата.

Необходимо еще раз подчеркнуть, что ввод-вывод происходит только с младшей частью аккумулятора, старшая часть во время выполнения команд IN ли OUT не изменяется. Также важно, что при данной реализации асинхронного обмена ЭВМ тратит время на ожидание (неопределенно долгое!) момента готовности циклически опрашивая флаг (spin-loop) и не может выполнять никакой другой работы. Разумная организация процедур ввода-вывода позволяет избегать такого зацикливания, например через периодический опрос флага при выполнении основной программы или через прерывания.

Таблица В.14

Ввод данных с использованием ВУ-2 Ассемблерный листинг.

Листинг примера: Ввод двух символов с устройства ввода ВУ-2 (DR#4, SR#5)

```
ORG      0x10
START:  CLA
S1:     IN   5      ; Ожидание ввода первого символа
        AND  #0x40 ; Бит 6 SR == 0 («Готов» нажата?)
        BEQ  S1      ; Нет - «Спин-луп»
        IN   4      ; Ввод первого символа
        SWAB          ; Перемещение первого символа в старшую часть АС
        ST   RES    ; Сохранение его в ячейке RES
S2:     IN   5      ; Ожидание ввода второго символа
        AND  #0x40 ; Бит 6 SR == 0 («Готов» нажата?)
        BEQ  S2      ; Нет - «Спин-луп»
        LD   RES    ; Ввод второго в младшие 8 разрядов А
        ST   RES
        HLT
RES:    WORD ?      ; Ячейка для записи слова "ДА"
```

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).

Когда будет нажата кнопка готовности, то в случае, если в контроллере прерывания разрешены (например, реализуется через схему „И“ регистра состояния SR и старшего бита регистра управления MR, рис. В.11, контроллер КВУ-1) поступает сигнал "Запрос прерывания" по линии IntRq. После чего этот сигнал поступит на схему „И“ центрального процессора В.11 с 5 битом регистра состояния (EI – разрешение прерываний) проверяющий разрешены ли прерывания в ЭВМ и записывающий результат схемы в 6 бит регистра состояния (INT – прерывание). После цикла исполнения очередной инструкции основной программы в ЭВМ будет запущен комплекс программ, проверяющий, было ли требование прерывания, и если было, то формирует управляющий сигнал "Предоставление прерывания" IntSC, который последовательно проходит через все контроллеры (сигналы IntSCo – выходной и IntSCI – входной) и проверяет было ли в этом контроллере прерывание. Когда сигнал доходит до нужного контроллера, то он выставляет на шину адреса свой номер вектора прерывания (см. рис. В.15). После чего процессор по номеру прерывания вызывает программу обработки прерывания.

Ячейки памяти с адресами 0-0xF отведены для инициализации векторов прерывания, по 2 ячейки на каждый вектор. В начале программы все контроллеры ВУ, в которых должны вызывать прерывания должны быть проинициализированы.

Если прерывания разрешены, то после выполнения всех команд кроме HLT, а

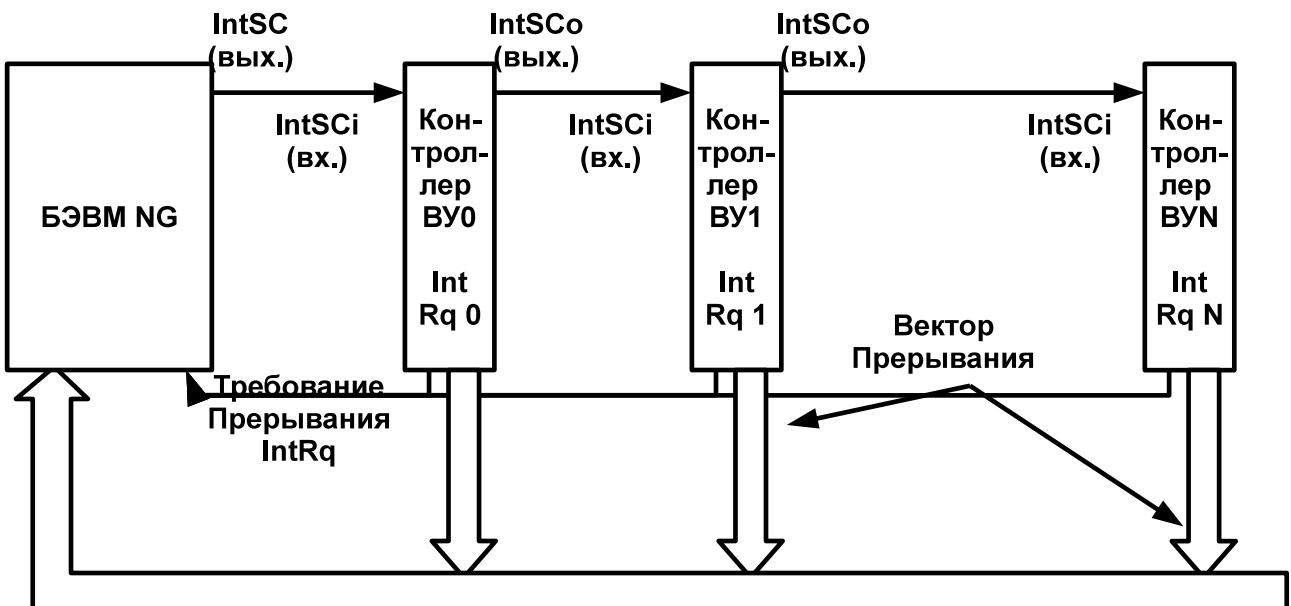


Рисунок В.15 Распространение сигналов предоставления прерывания

также если режим выполнения БЭВМ установлен в значение "РАБОТА", выполняются следующие действия:

Шаг 1. По завершению цикла исполнения текущей команды происходит переход на цикл прерывания. Если в этот момент б 6 бит регистра состояния INT (прерывание) не равен 1, то происходит переход к следующей команде. При наличии требования прерывания БЭВМ формирует сигнал "Предоставление прерывания" (IntSC) через выполнение микрокоманды INTS.

Шаг 2. Сохраняется счетчик команд и регистр состояния БЭВМ в стеке.

Шаг 3. Младшие 8 разрядов (номер вектора прерывания) записываются в буферный регистр и вычисляется адрес с переходом на подпрограмму обработки прерывания, как номер вектора * 2, после чего тот записывается в DR, а после в IP.

Шаг 4. Далее к младшим 8 разрядам буферного регистра прибавляется 1, чтобы выбрать адрес следующей ячейки вектора прерывания (новый регистр состояния PS), ограничивая результат 8-ю разрядами. После чего по этому адресу содержимое в памяти записывается в DR, а после в PS.

Шаг 5. 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 стеке, также восстанавливается сохраненный регистр состояния из стека.

Пример. Составить программу, которая постоянно наращивает на 1 содержимое аккумулятора. Восемь младших разрядов удвоенного значения аккумулятора должны выводиться на ВУ-1 по его запросу, а по запросу ВУ-3 содержимое регистра данных ВУ-3 должно записаться в ячейку 3F.

Таблица В.16

Ввод данных с использованием прерываний. Ассемблерный листинг.

Листинг примера: Готовность ВУ1: 2*A→РДВУ1, Готовность ВУ3: РДВУ3→ яч. 3F

```

ORG 0x0 ; Инициализация векторов прерывания
V0: WORD $DEFAULT, 0x180 ; Вектор прерывания #0
V1: WORD $INT1, 0x180 ; Вектор прерывания #1
V2: WORD $DEFAULT, 0x180 ; Вектор прерывания #2
V3: WORD $INT3, 0x180 ; Вектор прерывания #3
...
V7: WORD $DEFAULT, 0x180 ; Вектор прерывания #7
DEFAULT:IRET ; Просто возврат
ORG 0x020 ; Заргзка начальных векторов прерывания
START: DI
       CLA
       OUT 1      ; MR КВУ-0 на вектор 0
       OUT 5      ; MR КВУ-2 на вектор 0

```

Листинг примера: Готовность ВУ1: 2*А→РДВУ1, Готовность ВУЗ: РДВУЗ→ яч. ЗЕ

```

LD #9      ; разрешить прерывания и вектор №1
OUT 3      ; (1000|0001=1001) в МР КВУ-1
LD #0xB    ; разрешить прерывания и вектор №3
OUT 7      ; (1000|0011=1011) в МР КВУ-3
...
JUMP $PROG

PROG:
EI          ; Установка состояния разреяшени прерывания
CLA         ; Первоначальная очистка аккумулятора
INCLP: INC   ; Цикл для наращивания
           BR INCLP ; содержимого аккумулятора
ORG 0x03F

IO3: WORD ?  ; Ячейка для хранения кодов, поступающих с ВУ-3
ORG 0x040

INT1:        ; Прерывание сохранило содержимое PS
NOP         ; отладочная точка останова (NOP/HLT)
PUSH        ; Сохранили АС
ASL          ; Умножили АС на 2
OUT 2       ; Записали в РДВУ-1 (DR#2)
POP          ; Вернули АС назад

PROG:
EI          ; Установка состояния разр. прерывания
CLA         ; Первоначальная очистка аккумулятора
INCLP: INC   ; Цикл для наращивания
           BR INCLP ; содержимого аккумулятора
ORG 0x040

INT3:        ; Прерывание сохранило содержимое PS
NOP         ; отладочная точка останова (NOP/HLT)
PUSH        ; АС кладем в стек
CLA
IN 6        ; РДВУ-3
ST $IO3     ; сохранение
NOP         ; отладочная точка останова (NOP/HLT)
POP          ; АС вынимаем из стека
IRET        ; Возврат из обработки прерывания

```

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 кнопок готовности ВУ, то будет выполнен переход к подпрограмме обработки прерываний.

Для отладки программы (табл. В.8) в ней используются точки останова. Если в ячейке памяти точки останова расположена команда NOP, то программа выполняется в обычном, не отладочном режиме. Если вместо NOP поместить команду HLT, то программа во время выполнения остановится и будет возможно проконтролировать содержимое регистров и ячеек памяти. После завершения контроля необходимо продолжить дальнейшее исполнение программы. В приведенном примере точки останова позволяют исследовать содержимое аккумулятора в момент возникновения прерывания, а так же правильность подсчета и вывода на ВУ-3 удвоенного его значения.

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

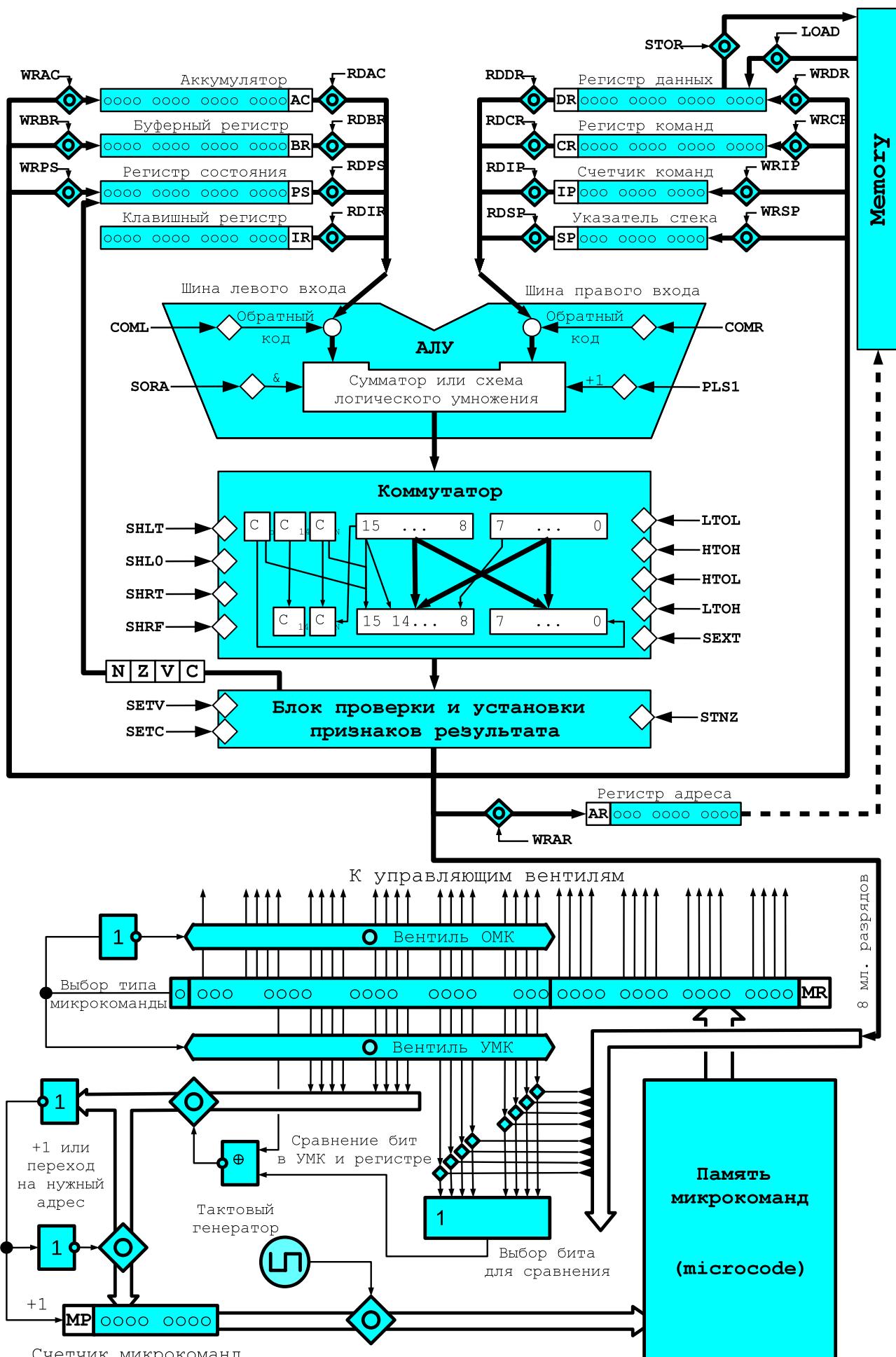


Рисунок В.17 Микропрограммное устройство управления БЭВМ с блоком регистров и АЛУ

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.17).

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.

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) пересыпается очередная микрокоманда, т.е. микрокоманда, на которую указывает счетчик микрокоманд (СЧМК), одновременно выполняющий функции регистра адреса микрокоманд.

Существует 2 типа микрокоманд (рис. В.18) - операционная (ОМК) и управляющая (УМК). Биты 0-15 обоих типов совпадают, логика остальных зависит от типа. В 39-ый бит РМК записывается код операции, исходя из которого происходит выбор типа микрокоманды (0 - для ОМК; 1 - для УМК). Этот сигнал подается напрямую на вентили УМК (16-32 биты РМК) и через инвертор на вентили ОМК (16-35 биты РМК), открывая/закрывая биты УМК/ОМК, тем самым производя выбор типа микрокоманды.

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.

Так как биты 0-15 обоих типов микрокоманд совпадают, то назначение у них одинаковое, биты 0-7 отвечают за сигналы чтения регистров, регистры 8-11 отвечают за действия в АЛУ, биты 12-15 отвечают за сигналы передачи в коммутаторе. Далее назначение битов в каждом типе отличается:

- ОМК: биты 16-23 отвечают за сигналы преобразований и установки флагов в коммутаторе;
- ОМК: биты 24-31 - запись в регистр результат преобразований;
- ОМК: биты 32-33 - взаимодействие с памятью БЭВМ
- ОМК: биты 34-35 - взаимодействие с внешними устройствами
- ОМК: бит 38 - HALT, бит останова.

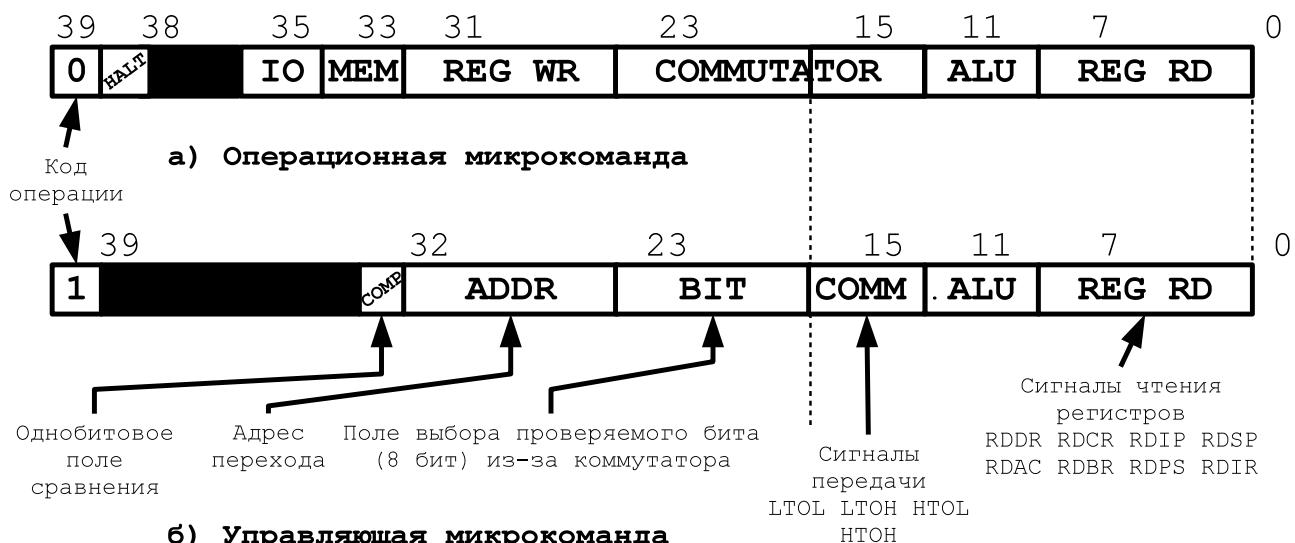


Рисунок В.18 Форматы микрокоманд

- УМК: биты 16-23 - выбор бита для сравнения, работающий по схеме „ИЛИ“ с данными поступающими из коммутатора;
- УМК: биты 24-31 задают адрес перехода, куда перейдет программа в случае, если схема инверсного „XOR“ даст положительный результат (бит сравнения и результат схемы „И“ выбора бита одинаковы);
- УМК: бит 32 - поле для сравнения с схеме инверсного „XOR“.

Справочная информация об отдельных операциях в микрокоманде приведена в табл. В.19.

Рассмотрим несколько примеров операционных микрокоманд.

1. Для вычитания содержимого DR из содержимого AC и записи результата в буферный регистр (AC-DR→BR) следует выполнить микрокоманду:

$$(0000\ 0000\ 0010\ 0000\ 0000\ 1001\ 0101\ 0001\ 0001)_2 = (0020009511)_{16},$$

т.е. одновременно подать единичные управляющие сигналы на чтение аккумулятора RDAC и регистра данных RDDR. Тогда к уменьшаемому прибавится обратный код вычитаемого и к этой сумме добавится единица, что эквивалентно AC + (-DR), а после записываем результат в буферный регистр.

2. Для сложения содержимого AC с содержимым DR и записью результата в буферный регистр с выставлением флагов (AC+DR→BR, NZVC) следует выполнить микрокоманду:

$$(0000\ 0000\ 0010\ 0000\ 1110\ 0000\ 1001\ 0000\ 0001\ 0001)_2 = (0020009511)_{16},$$

т.е. одновременно подать единичные управляющие сигналы на чтение аккумулятора RDAC и регистра данных RDDR. Тогда левый и правый вход АЛУ суммируются, выставляются флаги и результат записывается в буферный регистр.

3. Для возведения в дополнительный код содержимого AC и записи результата в AC с выставлением флагов (~AC + 1→AC, NZVC) следует выполнить микрокоманду:

$$(0000\ 0000\ 0001\ 0000\ 1110\ 0000\ 1001\ 0110\ 0001\ 0000)_2 = (0010E09610)_{16},$$

т. е. подать сигнал на чтение аккумулятора RDAC. Тогда, к обратному коду левого входа прибавится единица, что эквивалентно -AC + 0, а после выставляются флаги и записывается результат в аккумулятор.

Рассмотрим две управляющих микрокоманды:

1. Переход к команде, расположенной по адресу, на которую указывает адресная часть команды, если аккумулятор содержит нечетное число.

$$(1000\ 0001\ 0101\ 1100\ 0000\ 0001\ 0001\ 0000\ 0001\ 0000)_2 = (815C011010)_{16}$$

Микрокоманда подает сигнал на чтение аккумулятора RDAC, прямую передачу младшего байта и выбор нулевого бита данных, поступивших с коммутатора для сравнения. Далее задается адрес перехода и поле сравнения равное 1.

Эта микрокоманда переходит на метку BR @ 5C, в которой происходит:

5C 0020011002 extend sign CR(0..7) ? BR - расширение знака

5D 0004009024 BR + IP ? IP - сложение буферного регистра BR с счетчиком команд IP и запись в счетчик IP

5E 80C4101040 GOTO INT @ C4 - переход к циклу прерывания (обязателен в конце каждой инструкции, обратите на это внимание при выполнении лабораторной работы).

2. Для организации безусловного перехода (например, по адресу 90) используется 4-й бит регистра состояний, содержащий константу 0 (см. табл. В.20). Сравнение этого разряда с нулем, записанным в 32-ый разряд УМК, всегда дает положительный результат и позволяет переслать в СЧМК нужный адрес перехода. Микрокоманда, реализующая эту операцию, имеет вид:

$$(1000\ 0000\ 0101\ 1100\ 0001\ 0000\ 0001\ 0000\ 0100\ 0000)_2 = (805C101040)_{16}.$$

Таблица В.19.

Операции в микрокоманде		
Бит	Мнемоника	Назначение
Чтение регистров		
00	RDDR	DR (РД) → правый вход АЛУ
01	RDCR	CR (РК) → правый вход АЛУ
02	RDIP	IP (СК) → правый вход АЛУ
03	RDSP	SP (УС) → правый вход АЛУ
04	RDAC	AC (А) → левый вход АЛУ
05	RDBR	BR (БР) → левый вход АЛУ
06	RDPS	PS (РС) → левый вход АЛУ
07	RDIR	KR (КР) → левый вход АЛУ
АЛУ		
08	COMR	Обратный код правого входа АЛУ
09	COML	Обратный код левого входа АЛУ
10	PLS1	Операция A + B + 1 (PLuS 1)
11	SORA	Операция И (Sum OR And) (да, мы наркоманы)
Управление коммутатором		
12	LTOL	Прямая передача младшего байта
13	LTON	Передача младшего байта в старший
14	HTOL	Передача старшего байта в младший
15	HTON	Прямая передача старшего байта
16	SEXT	Расширение знака младшего байта (sign extend)
17	SHLT	Сдвиг влево (арифметический) (SHift Left)
18	SHL0	Передача старого значения С в младший бит (для циклического сдвига влево)
19	SHRT	Сдвиг вправо (арифметический) (SHift Right)
20	SHRF	Переключатель сдвига вправо (для циклического сдвига 15 разряд)
21	SETC	Установка С
22	SETV	Установка V
23	STNZ	Установка N и Z
Запись в регистры		
24	WRDR	АЛУ → РД
25	WRCR	АЛУ → РК
26	WRIP	АЛУ → СК
27	WRSP	АЛУ → УС
28	WRAC	АЛУ → А
29	WRBR	АЛУ → БР
30	WRPS	АЛУ → РС
31	WRAR	АЛУ → РА
Работа с памятью		
32	LOAD	ОП → РД
33	STOR	РД → ОП
Организация ввода-вывода		
34	IO	Передача адреса и приказа на ВУ
35	INTS	Предоставление прерывания
36		Зарезервирован
37		Зарезервирован
Останов БЭВМ		
38	HALT	Останов
Тип команды		
39	TYPE	Бит выбора ОМК/УМК

Таблица В.20.

Содержимое регистра состояния

Бит	Мнемоника	Содержимое
0	C	Перенос
1	V	Переполнение
2	Z	Нуль
3	N	Знак
4	O	0 – используется для организации безусловных переходов в МПУ
5	EI	Разрешение прерываний
6	INT	Прерывание
7	W	Состояние тумблеров "РАБОТА/ОСТАНОВ" (1 – "РАБОТА")
8	P	Программа

3.2 Интерпретатор базовой ЭВМ

Полный текст микропрограммы (интерпретатора команд) приведен в табл. В.21. Интерпретатор состоит из нескольких блоков:

- Цикл выборки команд
- Цикл выборки адреса операнда и обработки режимов адресации
- Цикл выборки операнда
- Цикл исполнения
 - Декодирование и исполнение адресных команд
 - Декодирование и исполнение ветвлений
 - Декодирование и исполнение безадресных команд
- Декодирование и исполнение команд ввода-вывода
- Цикл прерывания
- Пультовые операции
- Свободные ячейки для:
 - Арифметической команды
 - Команды перехода
 - Безадресной команды

Память микрокоманд содержит 256 ячеек для хранения микрокоманд, включая резерв. Формат микрокоманд - горизонтальный.

Таблица В.21

Интерпретатор базовой ЭВМ (микропрограмма)

Адрес	Микрокоманда	Метка	Действие
01	00A0009004	INFETCH	IP → BR, AR
02	0104009420		BR + 1 → IP; MEM(AR) → DR
03	0002009001		DR → CR
04	8109804002		if CR(15) = 1 then GOTO CHKBR @ 09
05	810C404002		if CR(14) = 1 then GOTO CHKABS @ 0C
06	810C204002		if CR(13) = 1 then GOTO CHKABS @ 0C
07	8078104002		if CR(12) = 0 then GOTO ADDRESS @ 78
08	80C2101040		GOTO IO @ C2
09	800C404002	CHKBR	if CR(14) = 0 then GOTO CHKABS @ 0C
0A	800C204002		if CR(13) = 0 then GOTO CHKABS @ 0C
0B	8157104002		if CR(12) = 1 then GOTO BRANCHES @ 57
0C	8024084002	CHKABS	if CR(11) = 0 then GOTO OPFETCH @ 24
0D	0020011002	ADFETCH	extend sign CR(0..7) → BR
0E	811C044002		if CR(10) = 1 then GOTO T11XX @ 1C
0F	0080009024	T10XX	BR + IP → AR
10	0100000000		MEM(AR) → DR
11	8114024002		if CR(9) = 1 then GOTO T101X @ 14
12	81E0014002	T100X	if CR(8) = 1 then GOTO RESERVED @ E0
13	8024101040	T1000	GOTO OPFETCH @ 24
14	8119014002	T101X	if CR(8) = 1 then GOTO T1011 @ 19
15	0001009401	T1010	DR + 1 → DR
16	0200000000		DR → MEM(AR)
17	0001009201		~0 + DR → DR
18	8024101040		GOTO OPFETCH @ 24
19	0001009201	T1011	~0 + DR → DR
1A	0200000000		DR → MEM(AR)
1B	8024101040		GOTO OPFETCH @ 24
1C	8120024002	T11XX	if CR(9) = 1 then GOTO T111X @ 20
1D	81E0014002	T110X	if CR(8) = 1 then GOTO RESERVED @ E0
1E	0001009028	T1100	BR + SP → DR
1F	8024101040		GOTO OPFETCH @ 24
20	8023014002	T111X	if CR(8) = 0 then GOTO T1110 @ 23
21	0001009020	T1111	BR → DR
22	8028101040		GOTO EXEC @ 28
23	0001009024	T1110	BR + IP → DR
24	8026804002	OPFETCH	if CR(15) = 0 then GOTO RDVALUE @ 26
25	814A404002		if CR(14) = 1 then GOTO CMD11XX @ 4A
26	0080009001	RDVALUE	DR → AR
27	0100000000		MEM(AR) → DR
28	813C804002	EXEC	if CR(15) = 1 then GOTO CMD1XXX @ 3C
29	8130404002	CMD0XXX	if CR(14) = 1 then GOTO CMD01XX @ 30
2A	812D104002	CMD000X	if CR(12) = 1 then GOTO OR @ 2D
2B	0010C09811	AND	AC & DR → AC, N, Z, V
2C	80C4101040		GOTO INT @ C4
2D	0020009B11	OR	~AC & ~DR → BR
2E	0010C09220		~BR → AC, N, Z, V
2F	80C4101040		GOTO INT @ C4
30	8137204002	CMD01XX	if CR(13) = 1 then GOTO CMD011X @ 37
31	8134104002	CMD010X	if CR(12) = 1 then GOTO ADC @ 34
32	0010E09011	ADD	AC + DR → AC, N, Z, V, C
33	80C4101040		GOTO INT @ C4
34	8032011040	ADC	if PS(C) = 0 then GOTO ADD @ 32
35	0010E09411		AC + DR + 1 → AC, N, Z, V, C
36	80C4101040		GOTO INT @ C4
37	813A104002	CMD011X	if CR(12) = 1 then GOTO CMP @ 3A
38	0010E09511	SUB	AC + ~DR + 1 → AC, N, Z, V, C

Адрес	Микрокоманда	Метка	Действие
39	80C4101040		GOTO INT @ C4
3A	0000E09511	CMP	AC + ~DR + 1 → N, Z, V, C
3B	80C4101040		GOTO INT @ C4
3C	8143204002	CMD1XXX	if CR(13) = 1 then GOTO CMD101X @ 43
3D	81E0104002	CMD100X	if CR(12) = 1 then GOTO RESERVED @ E0
3E	0001009201	LOOP	~0 + DR → DR
3F	0220009201		~0 + DR → BR; DR → MEM(AR)
40	80C4804020		if BR(15) = 0 then GOTO INT @ C4
41	0004009404		IP + 1 → IP
42	80C4101040		GOTO INT @ C4
43	8146104002	CMD101X	if CR(12) = 1 then GOTO SWAM @ 46
44	0010C09001	LD	DR → AC, N, Z, V
45	80C4101040		GOTO INT @ C4
46	0020009001	SWAM	DR → BR
47	0001009010		AC → DR
48	0210C09020		BR → AC, N, Z, V; DR → MEM(AR)
49	80C4101040		GOTO INT @ C4
4A	8153204002	CMD11XX	if CR(13) = 1 then GOTO ST @ 53
4B	814E104002	CMD110X	if CR(12) = 1 then GOTO CALL @ 4E
4C	0004009001	JUMP	DR → IP
4D	80C4101040		GOTO INT @ C4
4E	0020009001	CALL	DR → BR
4F	0001009004		IP → DR
50	0004009020		BR → IP
51	0088009208	PUSHVAL	~0 + SP → SP, AR
52	8055101040		GOTO STORE @ 55
53	0080009001	ST	DR → AR
54	0001009010		AC → DR
55	0200000000	STORE	DR → MEM(AR)
56	80C4101040		GOTO INT @ C4
57	8171084002	BRANCHES	if CR(11) = 1 then GOTO BR1XXX @ 71
58	8166044002	BR0XXX	if CR(10) = 1 then GOTO BR01XX @ 66
59	8161024002	BR00XX	if CR(9) = 1 then GOTO BR001X @ 61
5A	815F014002	BR000X	if CR(8) = 1 then GOTO BNE @ 5F
5B	80C4041040	BEQ	if PS(Z) = 0 then GOTO INT @ C4
5C	0020011002	BR	extend sign CR(0..7) → BR
5D	0004009024		BR + IP → IP
5E	80C4101040		GOTO INT @ C4
5F	805C041040	BNE	if PS(Z) = 0 then GOTO BR @ 5C
60	80C4101040		GOTO INT @ C4
61	8164014002	BR001X	if CR(8) = 1 then GOTO BPL @ 64
62	815C081040	BMI	if PS(N) = 1 then GOTO BR @ 5C
63	80C4101040		GOTO INT @ C4
64	805C081040	BPL	if PS(N) = 0 then GOTO BR @ 5C
65	80C4101040		GOTO INT @ C4
66	816C024002	BR01XX	if CR(9) = 1 then GOTO BR011X @ 6C
67	816A014002	BR010X	if CR(8) = 1 then GOTO BCC @ 6A
68	815C011040	BCS	if PS(C) = 1 then GOTO BR @ 5C
69	80C4101040		GOTO INT @ C4
6A	805C011040	BCC	if PS(C) = 0 then GOTO BR @ 5C
6B	80C4101040		GOTO INT @ C4
6C	816F014002	BR011X	if CR(8) = 1 then GOTO BVC @ 6F
6D	815C021040	BVS	if PS(V) = 1 then GOTO BR @ 5C
6E	80C4101040		GOTO INT @ C4
6F	805C021040	BVC	if PS(V) = 0 then GOTO BR @ 5C
70	80C4101040		GOTO INT @ C4
71	81E0044002	BR1XXX	if CR(10) = 1 then GOTO RESERVED @ E0
72	81E0024002	BR10XX	if CR(9) = 1 then GOTO RESERVED @ E0

Адрес	Микрокоманда	Метка	Действие
73	8176014002	BR100X	if CR(8) = 1 then GOTO BGE @ 76
74	806D081040	BLT	if PS(N) = 0 then GOTO BVS @ 6D
75	806F101040		GOTO BVC @ 6F
76	806F081040	BGE	if PS(N) = 0 then GOTO BVC @ 6F
77	806D101040		GOTO BVS @ 6D
78	81A4084002	ADDRLESS	if CR(11) = 1 then GOTO AL1XXX @ A4
79	8189044002	AL0XXX	if CR(10) = 1 then GOTO AL01XX @ 89
7A	817D024002	AL00XX	if CR(9) = 1 then GOTO AL001X @ 7D
7B	80C4014002	AL000X	if CR(8) = 0 then GOTO INT @ C4
7C	80DE101040	HLT	GOTO STOP @ DE
7D	8183014002	AL001X	if CR(8) = 1 then GOTO AL0011 @ 83
7E	8181801002	AL0010	if CR(7) = 1 then GOTO NOT @ 81
7F	0010C00000	CLA	0 → AC, N, Z, V
80	80C4101040		GOTO INT @ C4
81	0010C09210	NOT	~AC → AC, N, Z, V
82	80C4101040		GOTO INT @ C4
83	8186801002	AL0011	if CR(7) = 1 then GOTO CMC @ 86
84	0000200000	CLC	0 → C
85	80C4101040		GOTO INT @ C4
86	8184011040	CMC	if PS(C) = 1 then GOTO CLC @ 84
87	0000208300		HTOH(~0 + ~0) → C
88	80C4101040		GOTO INT @ C4
89	8196024002	AL01XX	if CR(9) = 1 then GOTO AL011X @ 96
8A	8190014002	AL010X	if CR(8) = 1 then GOTO AL0101 @ 90
8B	818E801002	AL0100	if CR(7) = 1 then GOTO ROR @ 8E
8C	0010E60010	ROL	ROL(AC) → AC, N, Z, V, C
8D	80C4101040		GOTO INT @ C4
8E	0010F80010	ROR	ROR(AC) → AC, N, Z, V, C
8F	80C4101040		GOTO INT @ C4
90	8194801002	AL0101	if CR(7) = 1 then GOTO ASR @ 94
91	0001009010	ASL	AC → DR
92	0010E09011		AC + DR → AC, N, Z, V, C
93	80C4101040		GOTO INT @ C4
94	0010E80010	ASR	ASR(AC) → AC, N, Z, V, C
95	80C4101040		GOTO INT @ C4
96	819C014002	AL011X	if CR(8) = 1 then GOTO AL0111 @ 9C
97	819A801002	AL0110	if CR(7) = 1 then GOTO SWAB @ 9A
98	0010C11010	SXTB	extend sign AC(0..7) → AC, N, Z, V
99	80C4101040		GOTO INT @ C4
9A	0010C06010	SWAB	SWAB(AC) → AC, N, Z, V
9B	80C4101040		GOTO INT @ C4
9C	81A2801002	AL0111	if CR(7) = 1 then GOTO NEG @ A2
9D	81A0401002	AL01110	if CR(6) = 1 then GOTO DEC @ A0
9E	0010E09410	INC	AC + 1 → AC, N, Z, V, C
9F	80C4101040		GOTO INT @ C4
A0	0010E09110	DEC	AC + ~0 → AC, N, Z, V, C
A1	80C4101040		GOTO INT @ C4
A2	0010E09610	NEG	~AC + 1 → AC, N, Z, V, C
A3	80C4101040		GOTO INT @ C4
A4	81B5044002	AL1XXX	if CR(10) = 1 then GOTO AL11XX @ B5
A5	0080009008	AL10XX	SP → AR
A6	0100000000		MEM(AR) → DR
A7	81AE024002		if CR(9) = 1 then GOTO AL101X @ AE
A8	81AC014002	AL100X	if CR(8) = 1 then GOTO POPF @ AC
A9	0010C09001	POP	DR → AC, N, Z, V
AA	0008009408	INCSP	SP + 1 → SP
AB	80C4101040		GOTO INT @ C4
AC	0040009001	POPF	DR → PS

Адрес	Микрокоманда	Метка	Действие
AD	80AA101040		GOTO INCSP @ AA
AE	81B1014002	AL101X	if CR(8) = 1 then GOTO IRET @ B1
AF	0004009001	RET	DR → IP
B0	80AA101040		GOTO INCSP @ AA
B1	0040009001	IRET	DR → PS
B2	0088009408		SP + 1 → SP, AR
B3	0100000000		MEM(AR) → DR
B4	80AF101040		GOTO RET @ AF
B5	81BB024002	AL11XX	if CR(9) = 1 then GOTO AL111X @ BB
B6	81B9014002	AL110X	if CR(8) = 1 then GOTO PUSHF @ B9
B7	0001009010	PUSH	AC → DR
B8	8051101040		GOTO PUSHVAL @ 51
B9	0001009040	PUSHF	PS → DR
BA	8051101040		GOTO PUSHVAL @ 51
BB	81E0014002	AL111X	if CR(8) = 1 then GOTO RESERVED @ E0
BC	0080009008	SWAP	SP → AR
BD	0100000000		MEM(AR) → DR
BE	0020009001		DR → BR
BF	0001009010		AC → DR
C0	0210C09020		BR → AC, N, Z, V; DR → MEM(AR)
C1	80C4101040		GOTO INT @ C4
C2	81C7084002	IO	if PS(W) = 0 then GOTO IRQ @ C7
C3	0400000000	DOIO	IO
C4	80DE801040	INT	if PS(INT) = 0 then GOTO INFETCH @ 01
C5	8001401040		INTS
C6	0800000000		
C7	0088009208	IRQ	~0 + SP → SP, AR
C8	0001009004		IP → DR
C9	0200000000		DR → MEM(AR)
CA	0088009208		~0 + SP → SP, AR
CB	0001009040		PS → DR
CC	0220001002		LTOL(CR) → BR; DR → MEM(AR)
CD	00A0020020		SHL(BR) → BR, AR
CE	0100000000		MEM(AR) → DR
CF	0004009001		DR → IP
D0	0080001420		LTOL(BR + 1) → AR
D1	0100000000		MEM(AR) → DR
D2	0040009001		DR → PS
D3	8001101040		GOTO INFETCH @ 01
D4	00BBE00000	START	0 → DR, CR, SP, AC, BR, AR, N, Z, V, C
D5	80C3101040		GOTO DOIO @ C3
D6	0080009004	READ	IP → AR
D7	0104009404		IP + 1 → IP; MEM(AR) → DR
D8	80DE101040		GOTO STOP @ DE
D9	0080009004	WRITE	IP → AR
DA	0001009080		IR → DR
DB	0204009404		IP + 1 → IP; DR → MEM(AR)
DC	80DE101040		GOTO STOP @ DE
DD	0004009080	SETIP	IR → IP
DE	4000000000	STOP	Halt
DF	8001101040		GOTO INFETCH @ 01

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. Счетчик увеличивается на 1. |
| F6 | Чтение. Из ячейки памяти (по адресу расположенному в счетчике команд) информация читается в регистр данных. Счетчик увеличивается на 1. |
| 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 "РАБОТА" 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
Размещение в памяти	ORG адрес	ORG 0x10
Адресная команда	[метка:] МНЕМОНИКА АРГУМЕНТ	LD X ; прямая относительная ST \$Y ; прямая абсолютная LD -(X) JUMP (VALUES) SWAM (ARRAY) +
Безадресная команда	[метка:] МНЕМОНИКА	START: CLA
Команда ввода-вывода	[метка:] МНЕМОНИКА АДРЕСВУ	OUT 0x3
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.

Специальная метка START - указывает компилятору на первую выполняемую команду программы. Должна быть указана в любой программе.

Комментарии

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

Замеченные (и не исправленные) очепятки и логические несостыковки.

1. стр.35 входящий цепочный сигнал предоставления прерывания (*IntSCI# - Interrupt Supply Chain input*) — не нравится слово, предложить замену пока не могу