Федеральное агентство по образованию
Государственное образовательное учреждение высшего профессионального образования
Ульяновский государственный технический университет

МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ УПРАВЛЕНИЯ ЭЛЕКТРОПРИВОДОМ

Методические указания к лабораторным работам

Составитель В. М. Иванов

Ульяновск 2007

УДК 681.31 (076)
ББК 32.973.2.я73
М59

Рецензент д-р техн. наук, профессор А. В. Кузнецов.
Ответственный за выпуск зав. кафедрой «Электропривод и АПУ» д-р техн. наук, профессор В. Н. Дмитриев.
Одобрено секцией методических пособий научно-методического совета университета.

М59 **Микропроцессорные** системы управления электроприводом **:** методические указания к лабораторным работам/ сост. В. М. Иванов. – Ульяновск : УлГТУ, 2007. – 36 с.

Рассматриваются вопросы алгоритмического и программного обеспечения микропроцессорных систем управления. Первоначальные цели указаний: закрепление начальных знаний в микропроцессорной технике с последующим их углублением в области алгоритмического и программного обеспечения производственных систем, включая такие сложные вопросы, как прямое цифровое управление электроприводом. Данные указания является базовыми и могут быть использованы в дальнейшем при более глубоком изучении устройств числового программного управления и их программировании, рассматриваемых в курсе «Электроприводы с системами числового программного управления». Содержательная постановка данных работ может развиваться и уточняться. Основное внимание уделено принципам реализации цифровых регуляторов и программированию на языке ассемблера.

Методические указания составлены в соответствии с учебной программой курса «Микропроцессорные системы управления электроприводом» и могут быть использованы студентами заочного и дневного факультетов специальности 140604 «Электропривод и автоматизация промышленных установок и технологических комплексов». Данные указания могут использоваться при подготовке и выполнении лабораторных работ, а также в качестве дополнительного пособия по указанным курсам, в том числе при выполнении курсовых работ.

Работа подготовлена на кафедре ЭП и АПУ.

УДК 681.31(076)
ББК 32.973.2.я73

СОДЕРЖАНИЕ

Указания по технике безопасности …… …… 3
Введение ..…… 4
Лабораторная работа № 1
Изучение методов адресации микропроцессора «К1801ВМ2» ……... 6
Лабораторная работа № 2
Разработка программ на языке ассемблер ……..……. 12
Лабораторная работа № 3
Организация ввода-вывода информации в микроЭВМ …….. 18
Лабораторная работа № 4
Цифровые регуляторы …… 22
Лабораторная работа № 5
Алгоритмы электроавтоматики ……. 26
Приложения А – Г ……..…… 31
Библиографический список …….. 36

ПРАВИЛА ПО ТЕХНИКЕ БЕЗОПАСНОСТИ

При работе на компьютере и микроЭВМ, встроенных в системы числового программного управления, необходимо знать общие правила техники безопасности:

1. К работе допускаются студенты, прошедшие инструктаж по технике безопасности и охране труда.
2. Перед началом работы необходимо убедится, что выключатели «Сеть» на общем щитке и на стенде находятся в положении «Отключено», а штепсельные разъемы находятся в рабочем положении.
3. Производить замену блоков, делать перекоммутацию только при отключенном питании сети.
4. Запрещается включать стенд без разрешения преподавателя.
5. Во время эксплуатации двери устройства числового программного управления (УЧПУ) должны быть закрыты с помощью специального ключа. Ключ должен находится у обслуживающего персонала или у преподавателя.
6. При проведении работ с использованием осциллографа необходимо остерегаться одновременного касания руками испытательной панели и корпуса осциллографа.
7. Запрещается самостоятельно устранять неисправности, необходимо отключить напряжение и поставить в известность преподавателя. При работе и ремонте установки особое внимание следует обратить на надежное заземление. К болту заземления УЧПУ прокладывается медная шина или провод сечением не менее 2,5 мм.
8. Помещение лаборатории должно быть оборудовано системой пожарной сигнализации и первичными средствами пожаротушения (огнетушителями), а также средствами оказания первой медицинской помощи.

ВВЕДЕНИЕ

Прогресс в области электронных и информационных технологий существенным образом изменяет как подходы к проектированию систем управления производственными механизмами и установками, так и их элементную базу. За последние десятилетия в практике регулируемого электропривода все шире находят средства прямого цифрового управления. Область микропроцессорных систем управления, предназначенная для управления двигателями, получила специальное название Motor Control. Появление мощных, полностью управляемых полевых транзисторов MOSFET и биполярных транзисторов с изолированным затвором IGBT привело к развитию преобразовательной техники и расширению сферы применения синхронных и асинхронных электроприводов с преобразователями частоты. Другим фактором, обусловившим совершенствование регулируемого электропривода и расширение областей его применения, было создание микропроцессоров (МП) и однокристальных микроконтроллеров (МК) достаточной вычислительной мощности.

Существующая широкая номенклатура МК предназначена для разнообразных сфер применения и удовлетворяет самые разнообразные требованиям к параметрам локальных цифровых регуляторов. Усилия разработчиков в области электропривода и электронных компонентов привели к созданию интеллектуальных драйверов, интегрированных модулей, включающих в свой состав силовой преобразователь, вычислительное ядро, преобразователи информации и датчики физических параметров. Новые решения в МП технике, в том числе в архитектуре МК, исходят из обобщения аппаратных принципов систем управления. Именно совместные усилия специалистов различного профиля привели к созданию семейств МК, предназначенных для управления электродвигателями.

Потребительские свойства МП систем управления заложены в их архитектуре. Причем, с точки зрения потребителя, в ряде случаев различные семейства микроконтроллеров имеют близкие характеристики, и основные отличия их заключаются в архитектуре вычислительного ядра и системе команд.

Архитектура микропроцессора – совокупность аппаратных и программных средств, определяющая возможности создания конкретной микропроцессорной системы. Работу электронной вычислительной машины (ЭВМ) невозможно познать, не постигнув правил и тонкостей ее языка. Язык ЭВМ – это ее машинный код. Все процессы в машине на самом низком аппаратном уровне приводятся в действие только командами (инструкциями) машинного языка. На этом языке программист непосредственно взаимодействует с ЭВМ. Считается, что программировать на этом языке наиболее сложно, поскольку программист должен мыслить в терминах машинных функций. Однако программирование на машинном языке обеспечивает программисту полный контроль и управление каждой

последовательностью шагов, которые выполняет машина. Такое управление позволяет оптимизировать программу с точки зрения ее выполнения и требуемого для ее размещения объема памяти. Это особенно важно для систем работающих в реальном времени, к которым относятся цифровые системы управления электроприводом.

Для облегчения написания программ используется мнемоническая запись команд. В этом случае запись программы осуществляется на языке ассемблера, представляющего собой символическое представление машинного языка. При использовании языка ассемблера программиста отделяет от машины программа ассемблер – совокупность программных средств, предназначенных для ввода программы ее компоновки и трансляции (компиляции) с языка ассемблера на машинный язык. Ограниченные ресурсы памяти МК и их многообразие ставят под сомнение целесообразность разработки специализированных микроЭВМ, предназначенных для подготовки программ на языке ассемблера конкретного семейства. В настоящее время, в связи с широким распространением персональных компьютеров, более предпочтительным является использование кросс-сред программирования. Создание программ на персональной ЭВМ с использованием кросс-ассемблера значительно облегчается, так как в этом случае можно использовать мощные визуальные среды проектирования, позволяющие оптимизировать процесс написания программы и осуществить первичную проверку и отладку программы. Ведущие фирмы, производители микропроцессорной техники, особое внимание уделяют интегрированным средам разработки и отладки МП и МК. Данные среды включают в себя оценочные платы, средства подготовки и загрузки программ, эмуляторы и симуляторы, позволяющие осуществить прогонку программы и ее отладку.

Необходимость развития и проблемной ориентации цифровой аппаратуры требует объединения знаний и усилий специалистов различных областей техники, преодоления расхождений в концепциях и методологии. Характерным примером в области специализированной микропроцессорной техники является разработка программируемых контроллеров, ввод информации в которые осуществляется с точки зрения пользователя на языке релейно-контактных символов и логических функций.

Таким образом, использование микроконтроллеров приводит к изменению характера проектной среды разработки. Во многих случаях процесс проектирования систем управления электроприводом неразрывно связан с разработкой алгоритмического и программного обеспечения микроконтроллеров и микроЭВМ.

Необходимо отметить, что знания в предметной области во многих случаях являются тем решающим фактором, который гарантирует эффективность и конкурентоспособность разработки МП систем управления.

ЛАБОРАТОРНАЯ РАБОТА № 1
ИЗУЧЕНИЕ МЕТОДОВ АДРЕСАЦИИ МИКРОПРОЦЕССОРА «К1801ВМ2»

1. Цель работы

Получение навыков работы, изучение методов адресации процессора.

2. Методические указания по подготовке к работе

Данные в микроЭВМ, реализованных на базе процессора К1801ВМ2, представляются байтом или словом, состоящим из двух байт. Доступное программисту адресное пространство включает в себя: регистр состояния программы RS (РСП), регистры общего назначения (РОН), оперативное (ОЗУ) или постоянное (ПЗУ) запоминающее устройство, регистры периферийных устройств.

RS	РОН	ОЗУ, ПЗУ		Регистры внешних устройств
	R0 R1 R2 R3 R4 R5 R6 R7	Векторы прерывания	Программы, данные	
Центральный процессор		000 - 377	400 - 157777	160000 - 177777

Центральный процессор содержит 9 доступных программисту регистров. В регистре состояния программы хранятся признаки выполнения текущей команды (N,Z,V,C) и приоритета программных прерываний. Регистры общего назначения РОН служат для промежуточного хранения данных, указателей адресов и счетчиков циклов. Два РОН имеют специальное назначение: регистр R6 (SP – stack pointer) используется в качестве указателя стека УС, регистр R7 (PC – program counter) используется как счетчик команд СК.

Максимальное адресное пространство памяти равно 64К байт или 32К слов, что определено 16-разрядной шиной адресов-данных. Использование аппаратных средств (диспетчера памяти) позволяет расширить адресное пространство за счет страничной организации памяти. Первые 28К слов отведены для оперативного или постоянного запоминающего устройства. Начальные 256 адресов (000-377) являются системной областью памяти и предназначены для записи векторов прерывания. Последние 4К слов адресного пространства отведены под адреса внешних устройств.

Базовую систему команд микроЭВМ типа «Электроника 60» (см. приложение Б) можно разделить по их функциям: арифметические, логические, пересылки, условных и безусловных переходов. С точки зрения записи их в памяти и особенностей выполнения принято выделять команды безадресные, одноадресные и двухадресные.

Форматы команд. Формат команды представлен 16-разрядным словом. При записи команды различают две основные части: код операции (КОП), определяющий выполняемое действие, и адресную часть, позволяющую

определить операнды (данные, необходимые для выполнения команды). В системе команд используются четыре основных формата (рис.1).

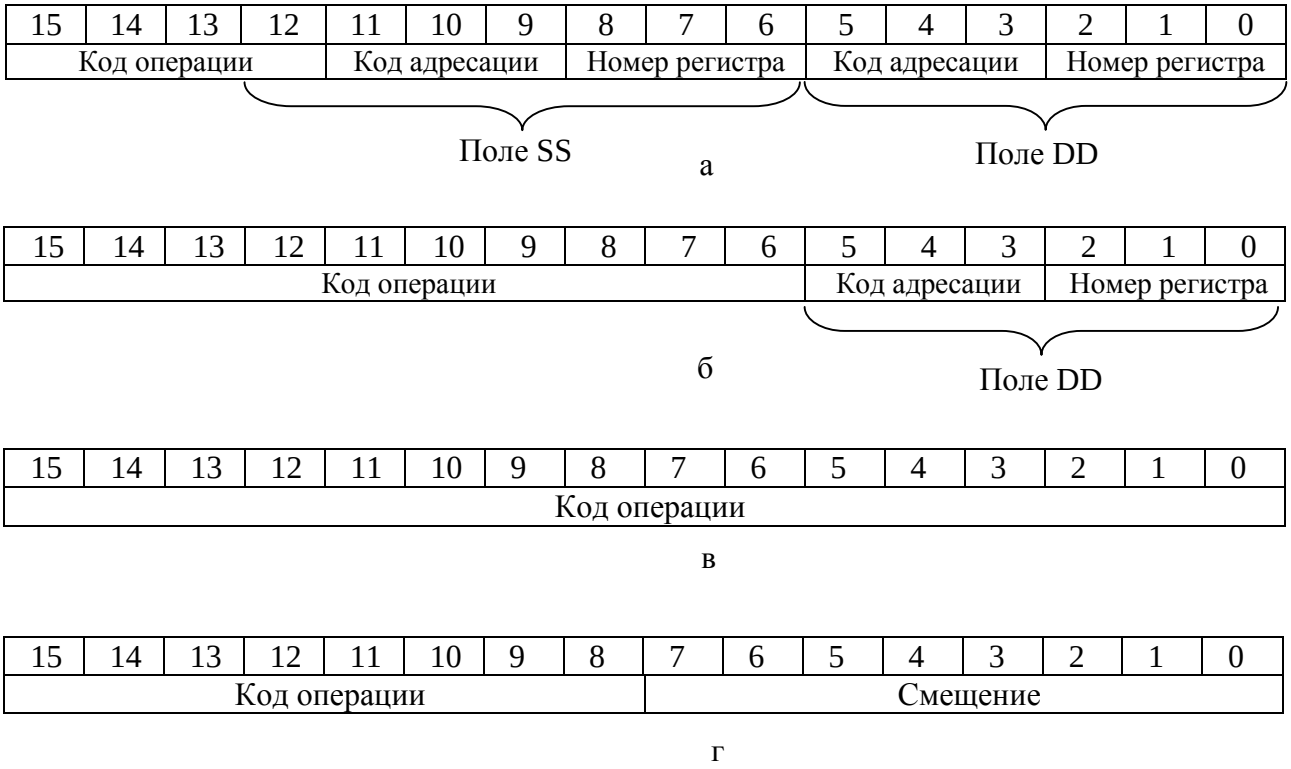

Рис.1. Основные форматы команд: а – двухадресные; б – одноадресные; в – безадресные; г – ветвления

Поля SS и DD предназначены для указания первого операнда (src – источника) и второго операнда (dst – приемника).

Доступ к операнду задается 6-разрядным полем, содержащимся в команде и включающим в себя 3-разрядный номер регистра и 3-разрядный указатель способа адресации (рис.1).

В ЭВМ используется двоичный способ представления информации. Для удобства представления двоичного содержимого слова используется восьмеричная система счисления. Для перехода от двоичной системы к восьмеричной достаточно разбить исходное двоичное число по три разряда справа налево и каждую триаду представить восьмеричной цифрой. В дальнейшем будет использоваться восьмеричная система счисления. Для представления 16-разрядного адреса или слова требуется 6 восьмеричных цифр, причем цифра старшего разряда может быть 0 или 1. Например, машинный код команды пересылки (MOV R1,@R2) в двоичной и восьмеричной системах счисления будет представлен следующими числами:

0001000001001011_2;

010113_8.

По этой команде содержимое регистра R1 будет пересылаться в ячейку памяти, адрес которой равен содержимому регистра R3.

Три бита, отведенные под код адресации, позволяют реализовать 8 способов адресации. Краткое их описание дано в приложении А.

Многообразие методов адресации позволяет обеспечить эффективную работу с данными и адресами. Ниже приведены примеры записи команд в сочетании с различными методами адресации:

0. Регистровый (Rn)

400/010102 MOV R1,R2 ; перегрузка содержимое регистра R1 в регистр R2.

До операции	После операции
R1/5 R2/1000	R1/5 R2/5

l. Косвенный регистровый (@Rn или (Rn))

400/011102 MOV @R1,R2 ; перегрузить содержимое ячейки памяти, адрес которой находится в регистре R1, в регистре R2.

До операции	После операции
R1/1000 1000/3 R2/2000	R1/1000 1000/3 R2/3

2. Автоинкрементный ((Rn)+)

400/012112 MOV (R1)+,(R2) ; перегрузить содержимое ячейки памяти, адрес которой указан в R1, в ячейку памяти, адрес которой указан в R2. После операции увеличить указатель R1 на 2.

До операции	После операции
R1/1000 1000/7 R2/2000 2000/3000	R1/1002 1000/7 R2/2002 2000/7

3. Косвенный автоинкрементный (@(Rn)+)

400/013122 MOV @(R1)+,(R2)+ ; перегрузить содержимое ячейки памяти, адрес адреса которой находится в R1, в ячейку памяти, адрес которой находится в R2. После операции содержимое R1 и R2 увеличивается на 2.

До операции	После операции
R1/1000 1000/2000 2000/12 R2/3000 3000/4000	R1/1002 1000/2000 2000/12 R2/3002 3000/12

4. Автодекрементный (-(Rn))

400/014132 MOV -(R1),@(R2)+ ; содержимое ячейки памяти, адрес которой находится как содержимое R1, уменьшенное на 2, перегрузить в ячейку памяти, адрес адреса которой указан в R2. После операции указатель R2 увеличивается на 2.

До операции	После операции
R1/1000	R1/776
776/16	776/16
R2/2000	R2/2002
2000/3000	2000/3000
3000/5	3000/16

5. Косвенный автодекрементный (@-(Rn))

400/015132 MOV @-(R1),R2 ;перегрузить содержимое ячейки памяти, адрес адреса которой равен содержимому R1, уменьшенному на 2, в регистр R2. Указатель R1 уменьшается на 2 до выполнения операции.

До операции	После операции
R1/1000	R1/776
776/3000	776/3000
3000/10	3000/10
R2/4000	R2/10

6. Индексный (X(Rn))

400/016132 MOV 1000(R1),@(R2)+ ; перегрузить содержимое ячейки
402/ 1000 ;памяти, адрес которой равен сумме содержимого R1 и смещения 1000, в ячейку памяти, адрес адреса которой указан в R2. После операции указатель R2 увеличивается на 2.

До операции	После операции
402/1000	402/1000
R1/1000	R1/1000
2000/15	2000/15
R2/4000	R2/4002
4000/5000	4000/5000
5000/10	5000/15

7. Косвенный индексный (@X(Rn))

400/017132 MOV @100(R1),R2 ; перегрузить содержимое ячейки
402/100 ; памяти, адрес адреса которой равен сумме содержимого R1 и смещения 100, в регистр R2.

До операции	После операции
402/100	402/100
R1/1000	R1/1000
1100/2000	1100/2000
2000/10	2000/10
R2/0	R2/10

Методы адресации через счетчик команд (R7).

2. Непосредственный (#n)

400/012731 MOV #10,R1 ; загрузить число 10 в регистр R1
402/000010

До операции	После операции
402/10 R1/1000	402/10 R1/10

3. Абсолютный (@#A)

400/013721 MOV @#1000,(R1)+ ; перегрузить содержимое ячейки
402/1000 ;памяти с адресом 1000 в ячейку памяти, адрес которой указан в регистре R1. После выполнения операции указатель R1 увеличить на 2.

До операции	После операции
402/1000 1000/7 R1/2000 2000/0	402/10 1000/7 R1/2002 2000/7

6. Относительный (X(R7) или A)

400/016711 MOV 1000(R7),@R1 ; перегрузить содержимое ячейки,
402/1000 ; адрес которой определен по
404/0 ; отношению к следующей команде со смещением на 1000, в ячейку памяти, адрес которой находится в R1.

До операции	После операции
402/1000 1404/5 R1/3000 3000/4000	402/1000 1404/5 R1/3000 3000/5

7. Косвенный относительный (@X(R7) или @A)

400/017701 MOV @100(R7),R1 ; перегрузить содержимое ячейки
402/100 ; памяти, адрес адреса которой определен по отношению к следующей команде со смещением на 100, в регистр R1.

До операции	После операции
402/100 1404/5000 5000/1 R1/0	402/1000 1404/5000 5000/1 R1/1

3. Задание для домашней подготовки

3.1. Ознакомится с организацией микроЭВМ с точки зрения программиста [1,2,3].

3.2. Изучить особенности режимов адресации на примерах выполнения отдельных команд [1,с.361-370].

4. Рабочее задание

Для заданных команд и сочетаний методов адресации составить программы их выполнения в машинных кодах, осуществить их ввод и проверку.

5. Порядок работы

Ввод данных осуществляется в среде симулятора 1801. После запуска программы в меню «Режим» выбрать пункт “1801ВМ2”. Ввод программы в машинных кодах производится с помощью следующих функциональных клавиш:

«/» – открыть ячейку. По этой команде на экран дисплея выводится содержимое ячейки памяти, РОН, RS. Пример: 1000/000000 11524, где 1000 – восьмеричный адрес ячейки памяти, которая должна быть открыта; 011524 – новое содержимое ячейки с адресом 1000.

«Enter» – закрыть ячейку и открыть следующую.

«Backspace» – закрыть ячейку и выйти на исходный режим связи с пультом.

«G» – запуск. По данной команде осуществляет запуск программы с адреса, который был введен до команды G. Пример: *1000G.

6. Содержание отчета

6.1. Титульный лист.

6.2. Цель работы.

6.3. Отчет по заданию, выводы по работе, анализ изменений, внесенных в первоначальный текст программы.

7. Контрольные вопросы

7.1. Укажите отличия в формате и выполнении безадресных, одноадресных и двухадресных команд.

7.2. Какие методы адресации через РОН эквивалентны абсолютному методу адресации через СК?

7.3. Какие особенности вычисления исполнительных адресов имеются при использовании относительного и косвенного относительного методов адресации через СК?

7.4. Есть ли эквивалент непосредственному методу адресации?

7.5. Какие методы адресации целесообразно использовать при работе с массивами информации?

ЛАБОРАТОРНАЯ РАБОТА № 2
РАЗРАБОТКА ПРОГРАММ НА ЯЗЫКЕ АССЕМБЛЕРА

1. Цель работы

Освоение методики разработки программ на языке ассемблера.

2. Методические указания по подготовке к работе

Программное обеспечение для систем управления, как правило, создается с использованием машинно-ориентированного языка ассемблера. Язык ассемблера – это символический язык, являющийся программным инструментом. Для перевода программ, написанных на языке ассемблера, на машинный язык используется системная программа, называемая ассемблером. В результате трансляции исходной программы формируется объектный модуль, содержащий последовательность байт (бит), которая с помощью программы загрузчика записывается в память вычислительной машины. Ассемблер является машинно-ориентированным языком и позволяет наиболее продуктивно использовать все возможности микропроцессорной системы.

Запись программы на ассемблере. Запись программы на языке ассемблера в основном сводится к записи в символьной форме команд, их операндов и меток (символических адресов, определяющих адреса переходов, подпрограмм и данных). В каждой строке программы на ассемблере должна быть записана только одна команда с соответствующими операндами. Общий формат записи определен следующими полями:

```
Метка:          Операция Операнд 1, Операнд 2      ; Комментарий
1               9        17                        33
```

Позиции полей, указанные в нижней строке, в общем случае не фиксированные, однако для большей наглядности и единообразия их рекомендуется соблюдать.

Поле метки. Метка представляет определяемое пользователем символьное имя, которое при трансляции определяет физический адрес ячейки памяти. Все метки заканчиваются двоеточием, по которому производится распознавание метки. Метки должны иметь уникальные имена, иначе при трансляции возникают ошибки.

Двойное двоеточие (::) определяет метку как глобальную. На такую метку может ссылаться другой, независимо транслируемый программный модуль. Ссылки в этой метке из других модулей будут вычислены компоновщиком при связывании объектных модулей и формировании загрузочного модуля.

Поле операции и операндов. Поле операции содержит символьное обозначение машинной команды, директиву ассемблера или макрокоманду.

Если оператор является мнемоническим обозначением инструкции, то генерируется код машинной инструкции. Затем ассемблер вычисляет адреса операндов. Если оператор является директивой, то ассемблер выполняет

соответствующие этой директиве действия. Если оператор является вызовом макрокоманды, то транслятор вставляет коды, сгенерированные как расширение вызываемой макрокоманды. Ограничителями оператора могут быть пробел или знак табуляции.

Операнды – это данные, над которыми совершается операция. Они могут быть представлены непосредственными данными, выражениями или заданы через адреса переменных (физические адреса), определяющих их нахождение. Если в поле операндов указываются несколько выражений, они должны быть разделены запятыми.

Поле комментария. Поле комментария позволяет делать необходимые пояснения, облегчающие понимание программы.

Директивы – это псевдокоманды управляющие работой программы ассемблер при трансляции программы. Директивы используются для облегчения ввода данных и компоновки программы. Список основных директив приведен в приложении В. Знание основных директив и возможностей транслятора позволяет достаточно эффективно писать программы на языке ассемблера [3,4].

Пример оператора:

M1: MOV #1000, R0 ;Загрузка указателя в R0

Здесь M1-метка, MOV-операция, #1000 и R0-операнды, после точки с запятой следует комментарий.

Перевод программы с языка ассемблера на машинный код выполняется с помощью специальной программы-транслятора. Символические имена ячеек памяти и данных вводятся с помощью директив ассемблера. Другая возможность работы с адресами ячеек памяти и данными заключается в прямом вводе их значений. В дальнейшем при разработке программ будем руководствоваться системой команд центрального процессора (ЦП) микроЭВМ «Электроника-60» и аналогичной системой команд ЦП К1801 ВМ1, ВМ2 (см. приложение Б). Специфические же особенности использования абсолютной и относительной адресации будем использовать в виде, принятом в данных процессорах. Заметим, что расширение кросс-ассемблера позволяет при работе с переменными изменять метод адресации, принятый по умолчанию. Следующий фрагмент программы показывает различия в трансляции при использовании относительного и абсолютного методов адресации.

PC	Машинный код	Команды с символическими адресами	Команды с абсолютными адресами
600 602	016700 000134	MOV SZAD,R0	MOV 134(R7),R0
604 606	013701 137000	MOV @#STEK,R6	MOV @#137000,R6

Здесь относительный адрес определяется относительно счетчика команд (адреса следующей команды) SZAD=604+134=740, а абсолютный адрес (STEK=160002) – задан словом в ячейке памяти, следующей за командой.

Способы записи алгоритмов. Этапу разработки программы обычно предшествует этап разработки алгоритма. Любой алгоритм записывается в виде некоторой последовательности действий (операторов) над входной информацией. К наиболее распространенным способам записи алгоритмов относятся: графический, операторный, словесный.

Для упрощения разработки алгоритмов рекомендуется соблюдать следующие правила:

1. Первоначальную разработку алгоритмов производить укрупнено с последующей поэтапной детализацией.

2. Строить алгоритм по модульному принципу, принимая в качестве модуля отдельные завершенные фрагменты, отладку которых можно производить автономно.

3. Использовать ограниченное число типовых структур.

Специфические особенности системы команд микроЭВМ накладывают в основном отпечаток на детализацию модулей, особенно, при первоначальном ознакомлении с микроЭВМ. Рассмотрим пример (см. рис. 2) разработки алгоритма упорядочивания массива произвольных положительных чисел в порядке их убывания.

Размещение чисел в заданном порядке осуществляется путем формирования промежуточного массива. При просмотре исходного массива определяется наибольшее число, которое пересылается в промежуточный массив, а затем наибольший элемент исходного массива исключается путем его обнуления. Прежде чем приступить к составлению программы необходимо распределить память, присвоить абсолютные адреса именам переменным и определить константы, заданные в символьной форме. На этапе отладки для размещения программы целесообразно использовать оперативную память. И хотя подобных ограничений при использовании средств эмуляции не возникает, будем придерживаться стандарта микроЭВМ «МС 1201.02», согласно которому под ОЗУ отведено адресное пространство в диапазоне адресов 400-40000. Конкретное назначение массивов информации здесь не рассматривается. Пусть исходный массив состоит из 10(8) чисел и определен с помощью директивы «.WORD» (см. программу 1). Под промежуточный массив отведем область памяти с помощью директивы «.BLKW». Введем символические адреса (Adm1 и Adm2), определяющие начало 1-го и 2-го массивов. Учитывая, что язык ассемблера поддерживает обработку арифметических выражений, конечные адреса массивов могут быть определены непосредственно в полях операндов как Adm1+10 и Adm2+10. Подобное определение операндов в принципе не обязательно. Эквивалентность обработки может быть достигнута, например, путем ввода размерности массива и использованием счетчика циклов.

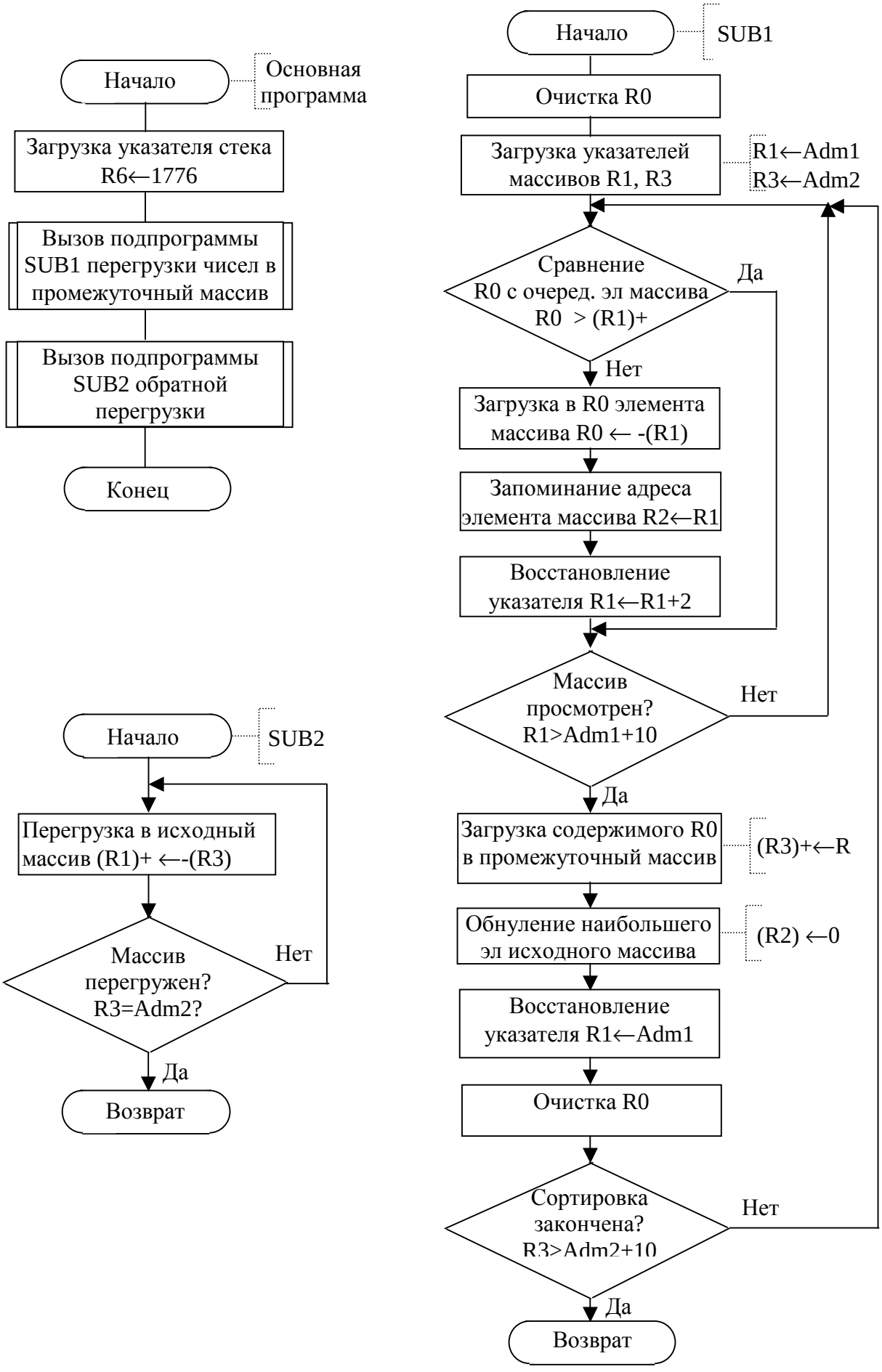

Рис. 2. Блок-схема алгоритма упорядочивания массива

Программа 1

```
        .
        TITLE SORT
        .ENABL LC
; *** ПРОГРАММА СОРТИРОВКИ ЧИСЕЛ ***
        .=120000                ;Установка счетчика программы
Adm1:       .WORD 20,1,17,2,16,3,15,4,5,7
Adm2:       .BLKW 10
        .=2000
        MOV #1776,R6
        JSR R7,@#SUB1
        JSR R7,@#SUB2
        HALT
;Поиск наибольшего числа и его загрузка в промежуточный массив
SUB1: CLR R0                    ;Очистка R0
        MOV #Adm1,R1            ;Загрузка указателя исходного массива
        MOV #ADM2,R3            ;Загрузка указателя промежуточного массива
M1:     CMP (R1)+,R0            ;Если очередной элемент массива
        BLE M2                  ;меньше предыдущего, то идти на M2
        MOV -(R1),R0            ;иначе загрузить элемент в R0
        MOV R1,R2               ;запомнить его адрес
        TST (R1)+               ;восстановить указатель R1
M2:     CMP R1,#Adm1+10         ;Массив просмотрен
        BLE M1                  ;если нет, то повторить цикл
        MOV R0,(R3)+            ;Перегрузка элемента в промежуточный массив
        CLR @R2                 ;Очистка наибольшего элемента
        MOV #Adm1,R1            ;Восстановить указатель
        CLR R0                  ;и буфер
        CMP R3,#Adm2+10         ;Конец сортировки
        BNE M1                  ;если нет, то повторить цикл
        RTS R7
;Перегрузка в порядке возрастания значений
SUB2: MOV -(R3),(R1)+
        CMP R3,#Adm2
        BLE SUB2
        RTS PC
```

3. Задание для домашней подготовки

3.1. Изучить систему команд микроЭВМ "Электроника-60" [2,3].

3.2. Изучить особенности разработки блок-схем алгоритмов и графического изображения операторов [5].

4. Рабочее задание

4.1. Получить задание у преподавателя в виде блок-схемы алгоритма и произвести при необходимости его детализацию.

4.2. По заданным исходным данным составить программу, осуществить ее трансляцию в машинных кодах и отладить ее.

5. Порядок работы

В среде «Windows Commander» перейти в каталог, заданный преподавателем. Путем нажатия клавиш «Shift+F4» создать файл (xxx.mac.). Осуществить ввод программы и ее трансляцию. Запуск транслятора (kross1.exe) осуществляется автоматически при нажатии на клавишу «Enter» на имени исходного файла (xxx.mac). В результате трансляции создается файл листинга (xxx.lst) и объектный файл (xxx.bin). Нажать на клавишу «F3» на имени файла (xxx.lst) и осуществить просмотр листинга. При наличии ошибок отредактировать исходный файл и повторить трансляцию. После устранения явных ошибок запустить программу имитатора процессора 1801ВМ2 «Simulator 1801ВМ2». После запуска программы в меню режим выбрать пункт “1801ВМ2”. Затем в меню файл выбрать пункт «Открыть файл», ввести нулевую страницу, начальный адрес загрузки и нажать кнопку «Открыть». Далее перейти в каталог и открыть файл (xxx.bin). Проверить работу программы. Для этого в меню режим выбрать пункт «Шаг». Набрать начальный адрес, нажать на клавишу «G», а затем на «P». При каждом нажатии на клавишу «P» будет выполняться очередная команда, осуществляться вывод содержимого регистров общего назначения и машинный код команды. Через клавишу «Backspace» можно выйти на просмотр любой ячейки памяти. Нажатие на клавишу «P» закрывает просмотр и продолжает выполнение программы. Для выполнения программы в целом необходимо отменить пункт «Шаг», и запустить программу через клавишу «G».

6. Содержание отчета

6.1. Титульный лист, цель работы.

6.2. Блок-схема алгоритма с подробным словесным описанием.

6.3. Программа с подробными комментариями, анализ ошибок, исходные данные, результат.

7. Контрольные вопросы

7.1. Доработайте алгоритм рис.4, приняв в качестве фактических данных начальные адреса массивов и их размерность.

7.2. При использовании, каких методов адресации можно ограничится одним указателем элементов массивов?

7.3. Назовите основные операции при выполнении команд "Вызов подпрограммы", "Возврат из подпрограммы".

7.4. Каким образом используются разряды 04 и 07 РСП?

7.5. Каким образом просчитывается смещение при ветвлении назад и вперед?

7.6. Напишите последовательность команд, эквивалентных команде SOB.

7.7. Что может располагаться в поле операции?

7.8. Как могут быть определены операнды?

7.9. Может ли предложение ассемблера содержать одно поле?

7.10. Какие директивы используются для ввода табличных данных?

ЛАБОРАТОРНАЯ РАБОТА №3
ОРГАНИЗАЦИЯ ВВОДА-ВЫВОДА ИНФОРМАЦИИ В МИКРОЭВМ

1. Цель работы

Ознакомление с методами обмена информацией с внешними устройствами.

2. Методические указания по подготовке к работе

Обмен информацией между центральным процессором и внешними устройствами можно осуществлять с использованием четырех способов: синхронного, асинхронного, по прерыванию, с использованием прямого доступа к памяти. Первые два способа называют программно управляемыми, так как обмен происходит полностью под управлением программы [3,4].

С точки зрения программиста внешнее устройство представлено набором регистров (регистры данных, регистры состояния и т. д.), каждый из которых имеет свой собственный адрес в поле адресов ЭВМ.

Регистры данных являются обычными накопительными регистрами, и их формат определяются только требованиями пользователя. В регистрах состояния рекомендуется придерживаться формата, показанного на рис.3.

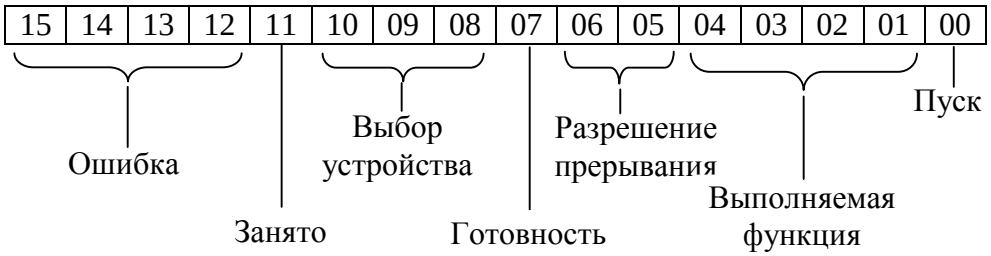

Рис. 3. Рекомендуемый формат регистров состояния ВУ

Рассмотрим два режима обмена с периферийными устройствами (ПУ): по программному опросу готовности ПУ и по прерыванию от ПУ. В качестве ПУ будем использовать терминал, состоящий из двух независимых устройств (клавиатуры и устройства отображения на экране дисплея).

Каждое из этих устройств имеет регистр состояния РС и регистр данных РД.

Терминал	Адреса		
	РС	РД	Вектор
Клавиатура	177560	177562	60
Дисплей	177564	177566	64

Алгоритм ввода символов с клавиатуры с опросом готовности показан на рис. 4,а. Ввод символов осуществляется с клавиатуры дисплея в буферную область памяти, начинающуюся с 1000-го адреса. В качестве указателя таблицы символов используется регистр R1. Ввод символов в память сопровождается их отображением на экран дисплея (режим «Эхо»). При

нажатии на клавишу одновременно с установкой бита готовности осуществляется запись кода нажатой клавиши в младший байт РД клавиатуры.

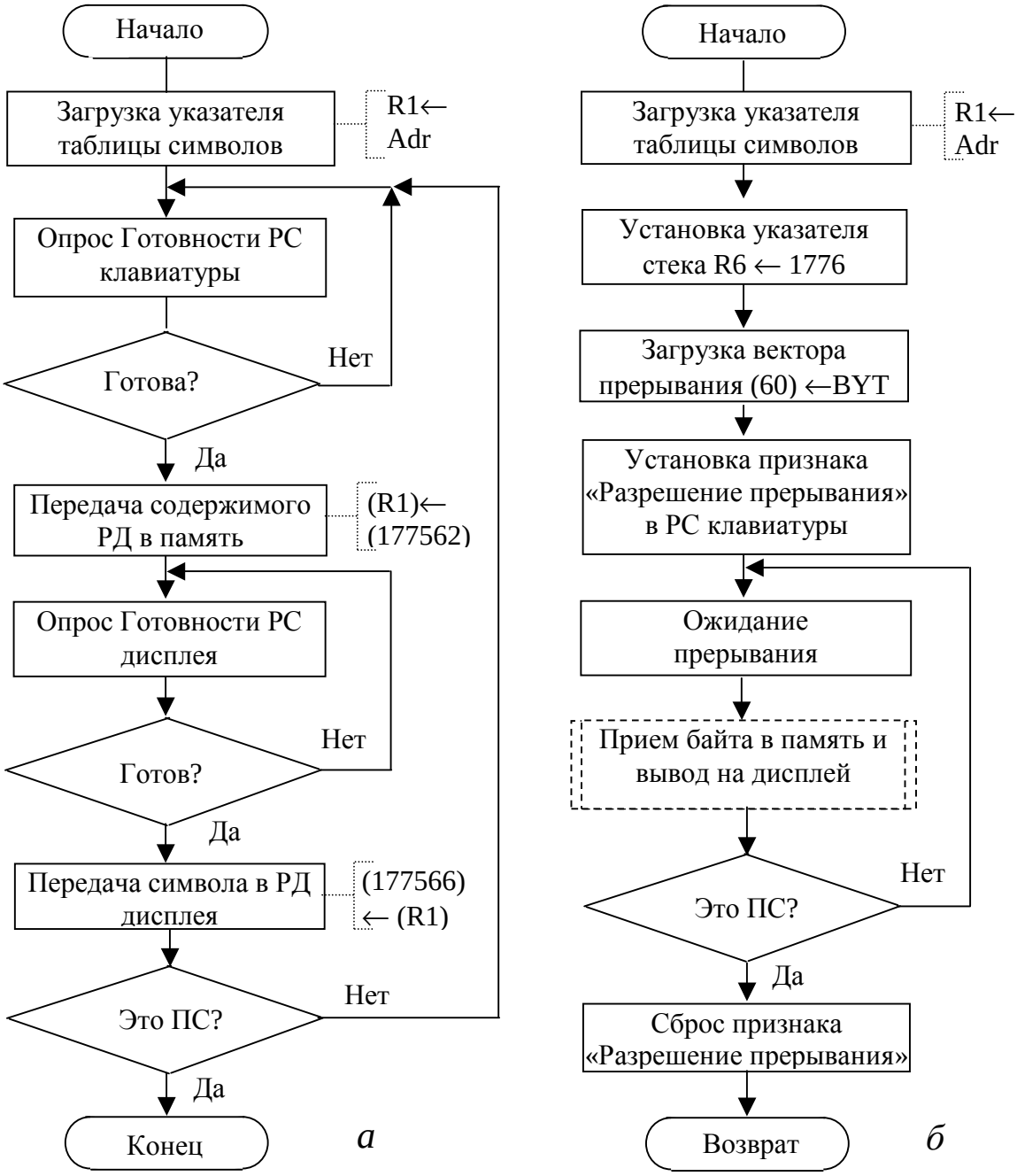

Рис. 4. Алгоритм ввода символов с клавиатуры в память:
а – с опросом готовности, *б* – по прерываниям

Информация кодируется в коде КОИ-7 [4, с.346]. Заканчивается ввод символов нажатием клавиши ПС. Так как ввод с клавиатуры медленнее, чем вывод символов на экран, то опрос готовности экрана в данном случае не обязателен. Программа выглядит следующим образом:

Программа 2

```
        MOV  #1000, R1          ;Загрузка указателя таблицы
M1:     TSTB @#177560           ;Опрос готовности клавиатуры
        BPL  M1                 ;Если не готова, то переход на M1
        MOVB @#177562, @R1      ;Перегрузка символа в память
M2:    TSTB @#177564            ;Опрос готовности экрана
        BPL  M2                 ;Если не готов, идти на M2
        MOVB @R1, @#177566      ;Передача символа на экран
        CMPB #12, (R1)+         ;Сравнение символа с ПС, если
        BNE M1                  ;не равны, ввод следующего символа
        HALT                    ;Останов
```

Специфической особенностью программы является использование команды BPL (ветвление, если плюс). Данная особенность связана с представлением чисел в ЭВМ. Признаком положительного числа является 0 в старшем разряде, а отрицательного – 1. Для байта – это 7-й разряд.

Главный недостаток обмена с опросом бита готовности связан с тем, что процессор во время ожидания ввода ничем другим заниматься не может. Это снижает производительность при выполнении программ.

Совмещение автономной работы периферийных устройств с одновременной работой процессора достигается в режиме прерывания программы.

На рис. 4,б показан алгоритм ввода символов в память с использованием прерывания. Внутренний цикл имитирует основную программу, при этом она будет работать в так называемом фоновом режиме. Вектор прерывания записан по 60-му адресу. Следующая ячейка памяти отводится под установку приоритета процессора. Программа имеет вид:

Программа 3

```
        MOV #1000,  R1          ;Загрузка указателя таблицы символов
        MOV #17776, R6          ;Загрузка указателя стека
        MOV #SUB, @#60          ;Загрузка адреса обслуживаемой подпрограммы
        MOV #100, @#177560      ;Загрузка признака разрешения прерывания
M1:     WAIT                    ;Ожидание прерывания
        CMPB #12,(R1)+          ;Сравнение символа с ПС, если не
        BNE  M1                 ;конец ввода, то идти на метку M1
        BIC  #100,@#177560      ;Очистка признака разрешения прерывания
        HALT                    ;Останов
;*** Подпрограмма обслуживания прерывания ***
SUB     MOVB @#177562, @R1      ;Перегрузка символа в память
        MOVB @R1, @#177566      ;Передача символа на экран
        RTI                     ;Возврат из прерывания
```

При обслуживании по прерыванию нескольких внешних устройств во второе слово вектора прерывания необходимо записать число 200(8), устанавливающее РСП[7]=1 и запрещающее процессору во время работы по

подпрограмме обслуживать другие прерывания. В целом работа с прерываниями, особенно при работе в реальном времени, оказывается сложной. Наиболее оправдано использование прерывания таймера, устройств последовательной связи, при опросе датчиков аварийных сигналов в системах ЧПУ и кнопок пульта оператора.

3. Задание для домашней подготовки

3.1. Изучить организацию обмена с внешними устройствами [3,4].

3.2. Изучить работу интерфейса устройств сопряжения с объектом [6].

4. Рабочее задание

4.1. Написать и отладить программы с опросом готовности и по прерываниям.

4.2. Произвести доработку алгоритмов ввода-вывода в соответствии с заданием, полученным от преподавателя.

Пример 1: осуществить ввод команды (MOV R1, @R2) с клавиатуры, ее распознавание и запись машинного кода команды в ячейку памяти с адресом 4000; написать программу и осуществить ее отладку.

Пример 2: обработать содержимое ячейки памяти 4000/010112 , выдать его на экран в символьной форме с последующим выводом мнемоники команды.

5. Содержание отчета

5.1. Титульный лист, цель работы.

5.2. Блок-схема разработанного алгоритма, программа.

5.3. Выводы по работе, отслеживающие ход отладки программы

6. Контрольные вопросы

6.1. Напишите последовательность команд, которую выполняет процессор при получении запроса на прерывание и возврате из прерывания.

6.2. Два внешних устройства одновременно выставляют сигнал требования прерывания. От какого устройства прерывания будут обработаны раньше: обладающего более высоким или более низким приоритетом?

6.3. В ячейке памяти загружена символьная информация 030060(8). Каким двум символам она соответствует?

6.4. Каким образом можно обеспечить поочередную работу двух внешних устройств, если они работают по прерываниям?

6.5. Нужно ли производить очистку экрана и перезапись строки символов, если режим ввода символов предусматривает использование клавиши удаления символов «Delete»?

ЛАБОРАТОРНАЯ РАБОТА № 4
ЦИФРОВЫЕ РЕГУЛЯТОРЫ

1. Цель работы

Ознакомление с принцами построения цифровых регуляторов и их программной реализацией.

2. Методические указания по подготовке к работе

При синтезе систем управления электроприводом могут быть использованы подходы, принятые в инженерной практике проектирования непрерывных систем регулирования, конечным итогом которых является выбор регулятора. Для перехода к дискретному регулятору от аналогового прототипа могут быть использованы методы подстановки, численного интегрирования. Однако наиболее универсальный метод основан на использовании структуры (рис. 5) дискретной аппроксимации свойств непрерывного звена [8,9]. Свойства дискретного регулятора $W_p(z)$ приближены к непрерывному $W_p(p)$ за счет введения фиксирующего звена $W_o(p)$ нулевого порядка, восстанавливающего информацию между интервалами квантования. Передаточная функция дискретного регулятора в этом случае находится с использованием дискретного преобразования Лапласа или Z-преобразования.

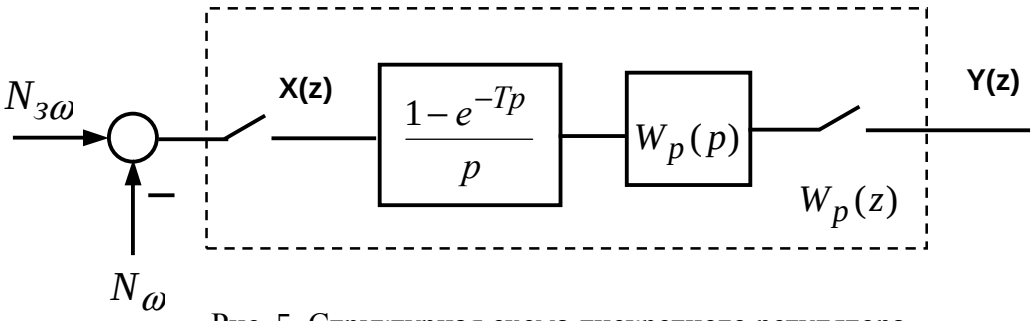

Рис. 5. Структурная схема дискретного регулятора

Выберем в качестве прототипа аналоговый ПИ-регулятор. Передаточную функцию дискретного ПИ-регулятора в соответствии со структурой (рис. 6) найдем в виде дискретного изображения последовательного соединения звеньев

$$W_p(z)=\frac{y(z)}{x(z)}=Z\left\{\frac{1-e^{-Tp}}{p}W_p(p)\right\}=Z\left\{\frac{1-e^{-Tp}}{p}\beta\frac{\tau p+1}{\tau p}\right\}=\frac{z-1}{z}Z\left\{\beta\frac{\tau_p+1}{\tau p^2}\right\}=$$

$$=\beta\frac{z-1}{z}Z\left\{\frac{1}{p}+\frac{1}{\tau p^2}\right\}=\beta\frac{z-1}{z}\left[\frac{z}{z-1}+\frac{Tz}{\tau(z-1)^2}\right]=\beta\frac{1-\left(1-T/\tau\right)z^{-1}}{1-z^{-1}}.$$

Раскроем передаточную функцию

$$\left(1 - z^{-1}\right) y(z) = \beta\left(1 - \left(1 - T/\tau\right) z^{-1}\right) x(z),$$

иначе

$$y(z) = \beta x(z) - \beta\left(1 - T/\tau\right) z^{-1} x(z) + z^{-1} y(z) \ .$$

Используя теорему запаздывания и упреждения, получим

$$y[n] = \beta x[n] - \beta\left(1 - T/\tau\right) x[n-1] + y[n-1].$$

Решение разностного уравнения можно реализовать на основе структур дискретных фильтров с многомерным входом или выходом или структур непосредственного программирования [8]. В первом случае может быть решена задача по минимизации ячеек памяти, используемых для размещения значений промежуточных переменных, во втором более просто решается задача ограничений значений переменных с целью избежания их переполнения. На рис. 6 показана одна из структур прямого (непосредственного) программирования полученного рекуррентного уравнения.

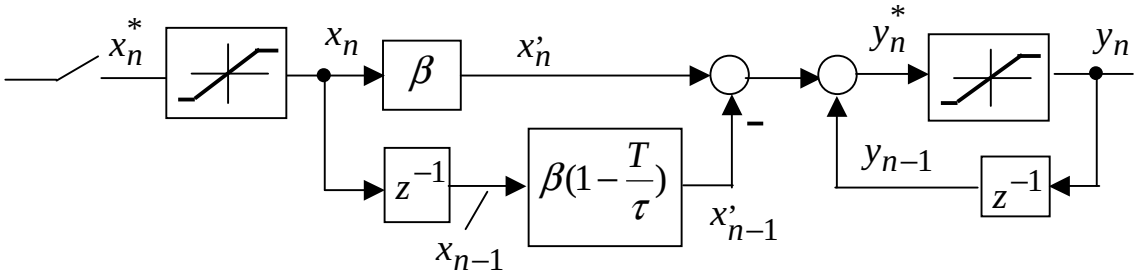

Рис. 6. Структура непосредственного программирования

Алгоритм регулятора скорости. Вхождение в алгоритм регулятора скорости (рис. 7) осуществляется по прерыванию от таймера. После вычисления ошибки между сигналом задания скорости $N_{3\omega}$ и обратной связи N_{ω} по скорости производится ее анализ на превышение значений ограничения.

Если ошибка меньше уровня ограничения, то алгоритм идет по обходной ветви, иначе значение ограничения присваивается вычисленной ошибке. После вычисления промежуточных величин происходит формирование выходной переменной.

Накопление интегральной составляющей производится также с учетом ограничений по входному сигналу. При обработке данного участка алгоритма с помощью промежуточных переменных y*[n], y'[n] определяют знак и величину модуля. При этом знак формируется в 15 разряде 16-разрядного числа выходной переменной путем принудительной его установки, а остальные разряды используются для формирования модуля. Далее идет вывод информации на ЦАП и переприсвоение значения переменных.

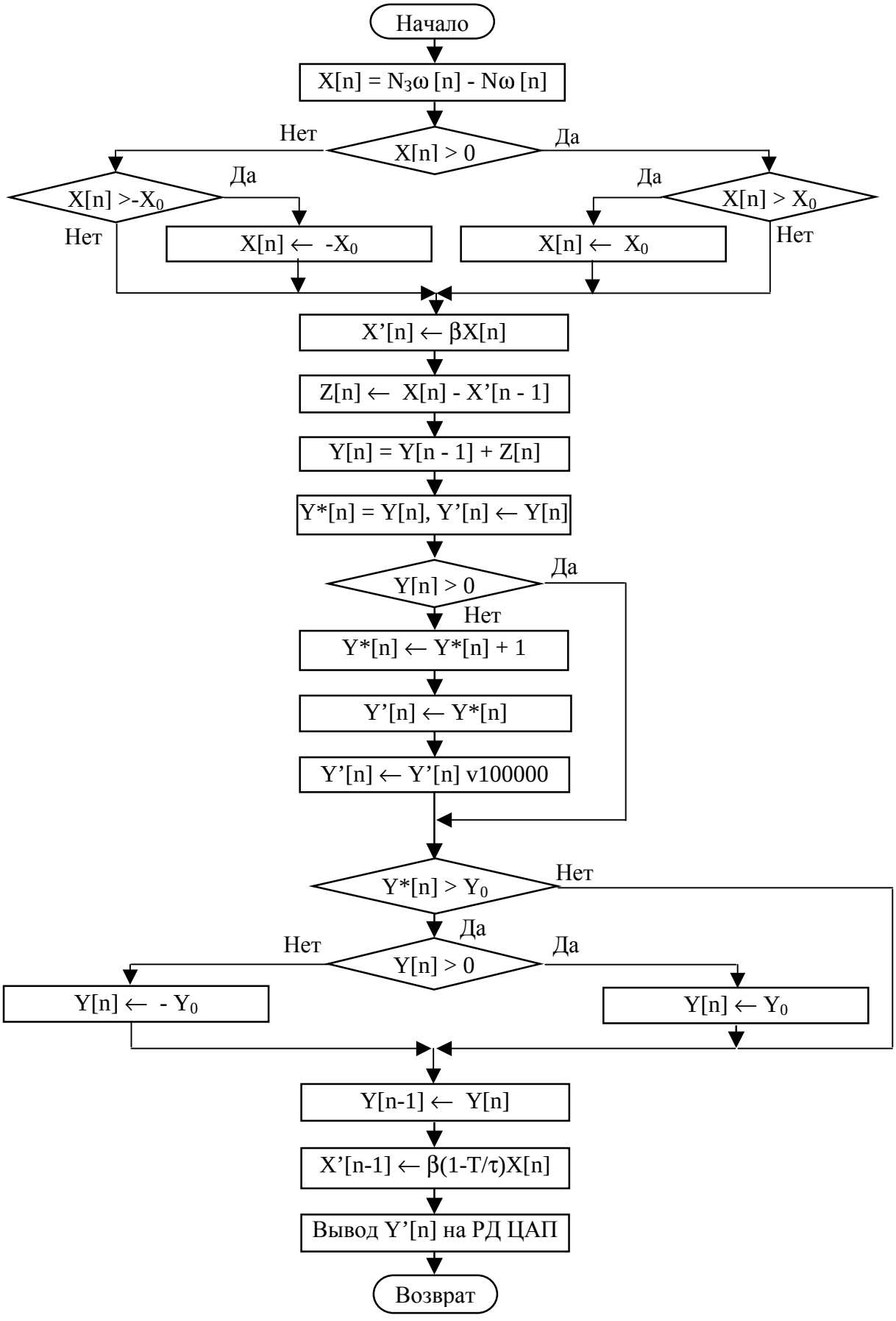

Рис. 7. Алгоритм регулятора скорости

3. Задание для домашней подготовки

3.1. Изучить теоретические основы D- и Z-преобразований и их использование при реализации дискретных регуляторов [7,8].

3.2. Ознакомится с алгоритмом (рис.7) и изучить особенности его реализации.

4. Рабочее задание

1. Получить задание у преподавателя в виде передаточной функции и типа цифрового фильтра.

2. В соответствии с указаниями к работе составить блок-схему алгоритма, написать программу и осуществить ее отладку.

5. Рекомендации по отладке программы

Отладку производить в среде симулятора путем прогона программы и просмотра значений переменных. Для просмотра переходной характеристики использовать пункт «График t» меню «Режим-Терминал». Для организации работы и обеспечения условий выхода дополнительно в основную программу ввести счетчик циклов.

6. Содержание отчета

6.1. Титульный лист, цель работы.

6.2. Блок-схема алгоритма с подробным словесным описанием.

6.3. Программа с комментариями и результаты ее выполнения.

7. Контрольные вопросы

7.1. Каким образом можно организовать учет ограничений в цифровых регуляторах?

7.2. Чем отличается структура дискретного фильтра с многомерным выходом от структуры непосредственного программирования.

7.3. Какие методы можно использовать при переходе от непрерывного регулятора к его дискретному аналогу?

7.4. Исходя из каких условий необходимо выбирать частоту прерываний от таймера?

7.5. Какие физические устройства могут быть использованы для реализации формирующей цепи, реакция которой на импульсное воздействие определяет импульс прямоугольной формы $[1(t) - 1(t - T)]$ функции?

7.6. Каким образом в МП системах обеспечивается заданный период квантования информации по времени?

ЛАБОРАТОРНАЯ РАБОТА № 5
АЛГОРИТМЫ ЭЛЕКТРОАВТОМАТИКИ

1. Цель работы

Ознакомление с методами программной реализации релейно-контактных схем управления.

2. Методические указания по подготовке к работе

Большинство задач управления технологическим оборудованием сводится к дискретному управлению. Электроавтоматика любой промышленной установки включает в себя две основные части: силовую и логическую. Силовая часть представляет собой группу исполнительных устройств, предназначенных для коммутации силовых электрических цепей. Логическая часть может быть представлена группой электромагнитных промежуточных реле или бесконтактных элементов, основным назначением которых является реализации логики управления в соответствии с информацией, поступающей с контактных и бесконтактных дискретных датчиков информации. Решения подобных задач в настоящее время осуществляется с помощью специализированных микропроцессорных систем, к которым относятся программируемые контроллеры и промышленные компьютеры. Специфической особенностью данных систем является то, что они комплектуются набором специализированных блоков ввода-вывода дискретной и аналоговой информации. В этом смысле устройства числового программного управления имеют те же специфические особенности. Более того, в настоящее время указанные группы устройств по набору блоков и функций управления имеют тенденции к сближению.

Программируемые контроллеры, как правило, имеют специализированное программное обеспечение и позволяют вводить программу с помощью ограниченного набора функциональных клавиш на языке релейно-контактных схем (РКС) или логических уравнений (Булевой алгебры логики). В отличие от них промышленные компьютеры позволяют свободное программирование. Для задания алгоритмов функционирования применяют специализированные языки программирования: языки символического кодирования, языки логических уравнений булевой алгебры и проблемно-ориентированные языки высокого уровня. Программируемые контроллеры, как правило, имеют специализированные программаторы. Однако в последнее время для программирования контроллеров обычно используются персональные компьютеры. Программное обеспечение в этом случае обеспечивает более наглядное визуальное проектирование, широкие возможности по редактированию программы и имитационному моделированию.

Программная реализация логических задач управления может быть осуществлена с помощью ряда методов: бинарного дерева, масок, табличного и таблично-алгоритмического [5] . Одним из самых простых в реализации

является метод бинарных программ, сущность которого раскроем на примере релейно-контактной схемы, показанной на рис. 8.

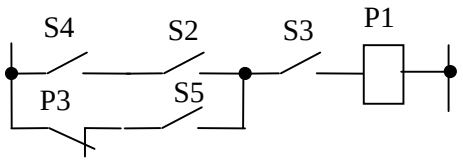

Рис. 8. Фрагмент релейно-контактной схемы

На схеме приведены символические обозначения: S – группа датчиков входной информации и P – группа выходных исполнительных устройств. В зависимости от числа блоков входных и выходных устройств конкретная привязка к номеру блока и его входу может быть осуществлена дополнительными цифровыми обозначениями, например, Snnmm, где nn – номер блока, mm – номер входа блока. Данная формализация позволяет упорядочить разработку и определить однозначное соответствие между обрабатываемой информацией и физическими адресами блоков.

При разработке алгоритма будем считать, что информация от датчиков Sn записывается в регистр состояния датчиков (RSS). Для формализации обработки информации примем следующий формат RSS:

Разряд	15	14			4	3	2	1	0
Датчик	S15	S14			S4	S3	S2	S1	S0

Для выходного регистра состояния исполнительных устройств (RSP) введем следующий формат:

Разряд	15	14			4	3	2	1	0
Датчик	P15	P14			P4	P3	P2	P1	P0

Схема алгоритма (рис. 9) представляет собой бинарное дерево, содержащее в худшем случае 2^n-1 вершин, соответствующих командам условного перехода. Конфигурация алгоритма, по сути, повторяет исходную электрическую схему с выделением основной цепи S4,S2,S3 и вспомогательной цепи P3,S5, которые сходятся в один узел S3. Каждый из разрядов проверяется его маской $MS_n = 2^n$, где n – номер разряда. Например, для S4 – маска MS4=20. Регистр R4 используется для установки признака включения соответствующего реле с помощью маски реле $MP_n = 2^n$. Для обработки информации введена промежуточная ячейка памяти состояния реле MSP и ее инверсия. Новое состояние разряда MSP формируется путем его предварительной очистки с последующим объединением с содержимым R4.

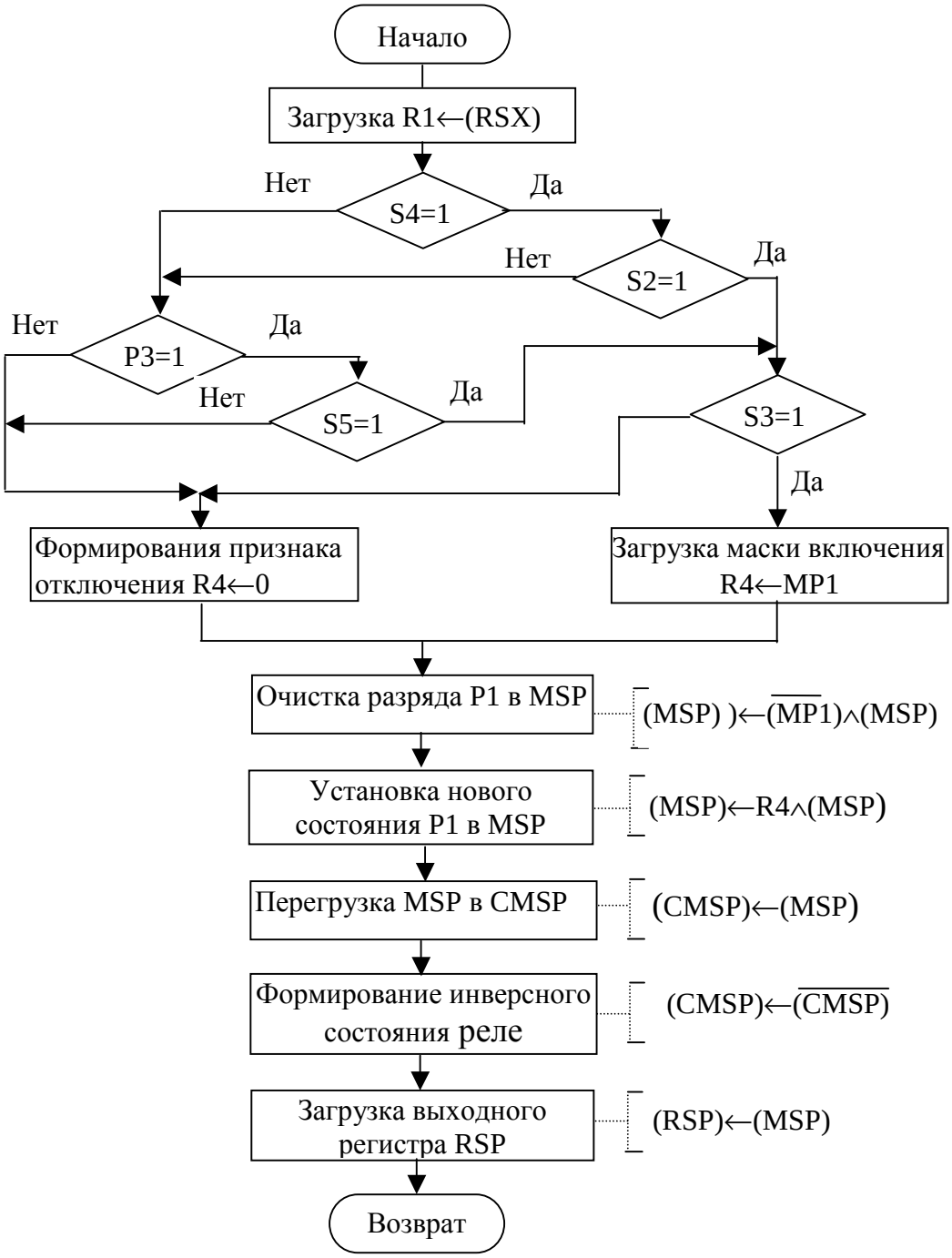

Рис. 9. Алгоритм обработки входных цепей реле P1

Программа реализации алгоритма, показанного на рис. 9, выглядит следующим образом:

Программа 4

```
.=2000                  ;Установка начального адреса
RSS=167606              ;Адрес регистра состояния датчиков
RSP=167636              ;Адрес регистра состояния реле
MS2=4                   ;Маска датчика S2
MS3=10                  ;Маска датчика S3
MS4=20                  ;Маска датчика S4
MS5=40                  ;Маска датчика S2
```

```
        MP1=2                   ;Маска реле P1
        MP3=10                  ;Маска реле P3
        MSP=4000                ;Промежуточная ячейка памяти состояния реле
        CMSP=4002               ;Буфер инверсных значений состояния реле
        MOV #1776,R6;
L1:     TST @#RSK;
        BMI L2;
        JSR R7,@#SUB1;
        BR L1;
L2:     HALT;
SUB1:   MOV  @#RSS, R1          ;Пересылка вектора состояния датчиков в R1
        BIT  #MS4, R1           ;Проверка 4-го разряда
        BEQ  M1                 ;Если S4=0, то переход на M1
        BIT  #MS2, R1           ;Проверка 2-го разряда
        BEQ  M1                 ;Если S2=0, то идти на M1
M2:     BIT  #MS3, R1           ;Проверка 3-го разряда
        BEQ  M5                 ;Если S3=0, то идти на M5
        MOV  #2, R4             ;Загрузка маски включения реле P1
        BR   M4                 ;Переход на метку M4
M1:     BIT  #MP3, @#CMSP       ;Проверка  P3
        BEQ  M5                 ;Если =0, то идти на M5
        BIT  #MS5, R1           ;Проверка 5-го разряда
        BNE  M2                 ;Если S5=1, то идти на M2
M5:     CLR  R4                 ;Отключить реле
M4:     BIC  #MP1, @#MSP        ;Очистка разряда P1 в MSP
        BIS  R4, @#MSP          ;Загрузка нового состояния P1
        MOV @#MSP, @#CMSP       ;Перегрузка состояния в CMSP
        COM @#CMSP              ;Формирование инв. значения состояний реле
        MOV @#MSP, @#RSP        ;Перегрузка в выход. регистр
        RTS R7;
.END;
```

3. Задание для домашней подготовки

1. Изучить основные методы программной реализации схем релейно-контакторного управления [5,10].

2. Ознакомится с алгоритмом (рис. 9) и изучить особенности его реализации.

4. Рабочее задание

1. Получить задание у преподавателя в виде электрической схемы.

2. В соответствии с указаниями к работе составить блок-схему алгоритма. Учесть, что при реализации схем с несколькими реле целесообразно цепи каждого реле обрабатывать автономно с использованием подпрограмм.

3. Разработать программу и произвести ее отладку.

5. Рекомендации по отладке программы

Составить таблицу состояний датчиков входных цепей реле, определяющую последовательность проверки состояний исполнительных

устройств (реле). Загрузить программу в симулятор и в меню режим выбрать пункт индикация. Осуществить проверку путем последовательного ввода восьмеричных данных в регистр RSS, соответствующих двоичным комбинациям состояния датчиков, и запуска программы. При необходимости перейти в режим трассировки и проследить прохождение программы. Окончательную проверку осуществить на УЧПУ 2С42-65. Для этого предварительно загрузить в УЧПУ следующую программу:

Программа 5

```
        .=500;
        MOV #2000,R1            ;Указатель начального адреса загрузки
M1:     TSTB @#177560           ;Выход по клавиатуре
        BMI L;
        TSTB @#177750           ; Опрос Готовности приемника
        BPL M1;
        MOVB @#177752,(R1)+     ; Прием байта
        BR M1                   ;Переход на следующий цикл
L:      HALT;
        .END
```

Выбрать в меню пункт «RS-232» и осуществить перегрузку объектного файла в УЧПУ. Запустить программу в УЧПУ и осуществить проверку. В качестве внешних датчиков информации и исполнительных устройств используются тумблеры и светодиоды, установленные на внешнем пульте.

6. Содержание отчета

1. Титульный лист, цель работы.
2. Принципиальная схема, блок-схема алгоритма.
3. Программа, выводы по целесообразной организации программы.

7. Контрольные вопросы

1. Дайте характеристику методов логической обработки информации с точки зрения унификации программ, быстродействия и затрат памяти.
2. Каким образом можно существенно сократить программу и повысить ее быстродействие, используя параллельную обработку информации в методе бинарных программ при реализации сложных электрических цепей?
3. Какое число запишется во внешний регистр, если датчики S1, S3, S6 находятся в замкнутом состоянии?
4. Какие изменения необходимо внести в программу загрузчика, если данные будут представлены в символьной форме?

ПРИЛОЖЕНИЯ А – Г

Приложение А.

Способы адресации процессора КР1801ВМ2.

Метод адресации			Описание
Обозначение	Код	Наименование	
Rn	0	Прямой	Операнд находится в регистре
¦@Rn или (Rn)	1	Косвенный	В регистре Rn находится исполнительный адрес операнда
(Rn)+	2	Автоинкрементный	В регистре Rn находится исполнительный адрес операнда. После выполнения команды содержимое Rn увеличивается на 2 для команд со словами или на 1 для байтовых операций
@(Rn)+	3	Косвенный автоинкрементный	В регистре Rn находится адрес адреса операнда. После выполнения команды содержимое Rn увеличивается на 2
-(Rn)	4	Автодекрементный	До выполнения команды содержимое Rn уменьшается на 2 (на 1 для байтовых команд) и используется как адрес операнда
@-(Rn)	5	Косвенный автодекрементный	До выполнения команды содержимое Rn уменьшается на 2 и используется как адрес адреса операнда
X(Rn)	6	Индексный	Адрес операнда находится как сумма индекса X и содержимого регистра Rn. Индекс X размещается в следующем слове команды
@X(Rn)	7	Косвенный индексный	Адрес адреса операнда находится как сумма индекса X и содержимого регистра Rn. Индекс X размещается в следующем слове команды
Методы адресации через счетчик команд (R7)			
#n	2	Непосредственный	Операнд n помещается в следующем слове команды
@#A	3	Абсолютный	Адрес (A) операнда помещается в следующее слово команды
A или X(R7)	6	Относительный	Значение выражения A определяет адрес операнда, который определяется как сумма смещения X и содержимого счетчика команд (адреса следующей команды). Смещение X размещается в следующем слове команды
@A или @X(R7	7	Косвенный относительный	Значение выражения A определяет адрес адреса операнда, который определяется как сумма смещения X и содержимого счетчика команд (адреса следующей команды). Смещение X размещается в следующем слове команды

Приложение Б.

Список команд процессора КР1801 ВМ2

Команда		Признаки	Результат операции	Описание
Мнем.	Код	N Z V C		
1	2	3	4	5
Одноадресные команды				
CLR(B)	*050DD		(d)←0	Очистка
COM(B)	*051DD		(d)←($\underline{d}$)	Инвертирование
INC(B)	*052DD		(d)←(d)+1	Прибавление единицы
DEC(B)	*053DD		(d) ←(d)-1	Вычитание единицы
NEG(B)	*054DD		(d)←($\underline{d}$)+1	Дополнительный код
TST(B)	*057DD	+ + 0 0	ССП←(d) ←(d)	Проверка
ASR(B)	*062DD	+ + + +	(d) ←(d)/2	Арифметический сдвиг вправо
ASL(B)	*063DD	+ + + +	(d) ←(d)*2	Арифметический сдвиг влево
ROR(B)	*060DD	+ + + +	(d_{n-1}) ←(d_n), (d15) ←C, C←(d0)	Циклический сдвиг вправо
ROL(B)	*061DD	+ + + +	(d_n) ←(d_{n-1}), (d0) ←C, C←(d15)	Циклический сдвиг влево
ADC(B)	*055DD	+ + + +	(d)←(d)+C	Прибавление переноса
SBC(B)¦	*056DD	+ + + +	(d)←(d)-C	Вычитание переноса
SXT	0067DD	- + 0 -	(d)←0, если N=0 (d)←1, если N=1	Расширение знака
SWAB	0003DD	+ + 0 0		Перестановка байтов
MFPS	1067DD	+ + 0 -	(d) ←ССП	Чтение ССП
MTPS	1064SS	+ + + +	ССП ←(d)	Запись ССП
Двухадресные команды				
MOV(B)	*1SSDD	+ + 0 –	(d)←(s)	Пересылка
CMP(B)	*2SSDD	+ + + +	ССП)←(s)-(d)	Сравнение
ADD	06SSDD	+ + + +	(d)←(d)+(s)	Сложение
SUB	16SSDD	+ + + +	(d)←(d)-(s)	Вычитание
BIT(B)	*3SSDD	+ + 0 –	ССП←(d)∧(s)	Проверка разрядов
BIC(B)	*4SSDD	+ + 0 –	(d)←(d)∧($\underline{s}$)	Очистка разрядов
BIS(B)	*5SSDD	+ + 0 -	(d)←(d)∨(s)	Логическое сложение
XOR	074RDD	+ + 0 –	(d)←(d)∀ (R)	Исключающее ИЛИ
Команды расширенной арифметики				
MUL	070RSS	+ + 0 +	R,R+1←R* (s)	Умножение
DIV	071RSS	+ + + +	R,R+1←R,R+1/(s)	Деление
ASH	072RSS	+ + + +	R↔R на NN, NN=(s5…s0)	Арифметический сдвиг

Продолжение приложения Б.

1	2	3	4	5
ASCH	073RSS	+ + + +	R,R+1↔R,R+1 на NN, NN=(s5…s0)	Арифметический сдвиг двойного слова
FADD	07500R	+ + 0 0	(R)+4,(R)+6) ←(R)+6,(R)+4 +(R),(R)+2	Сложение с плавающей запятой
FSAB	07501R	+ + 0 0	(R)+4,(R)+6) ←(R)+6,(R)+4 - (R),(R)+2	Вычитание с плавающей запятой
FMUL	07502R	+ + 0 0		Умножение с плавающей запятой
FDIV	07503R	+ + 0 0		Деление плавающей запятой
Команды управления программой				
BR	0004XXX		PC←PC+2*XXX,	Ветвление безусловное
BNE	0010XXX		если Z=0	Ветвление, если не равно нулю
BEQ	0014XXX		если Z=1	Ветвление, если равно нулю
BPL	1000XXX		если N=0	Ветвление, если плюс
BMI	1004XXX		если N=1	Ветвление, если минус
BVC	1020XXX		если V=0	Ветвление, если нет переполнения
BVS	1024XXX		если V=1	Ветвление, если переполнение
BCC	1030XXX		если C=0	Ветвление, если нет переноса
BCS	1034XXX		если C=1	Ветвление, если перенос
BGE	0020XXX		если N∀V=0	Ветвление, если > или = нулю
BLT	0024XXX		если N∀V=1	Ветвление, если < нуля
BGT	0030XXX		если N∀V=0	Ветвление, если > нуля
BLE	0034XXX		если Z∨(N∀V)=0	Ветвление, если < или = нулю
BHI	1010XXX		если C=0 и Z=0	Ветвление, если больше
BLOS	1014XXX		если C∨Z=1	Ветвление, если < или =
BHIS	1030XXX		Аналог BCC	Ветвление, если > или =
BLO	1034XXX		Аналог BCS	Ветвление, если меньше
SOB	077RNN		R←R-1, PC←PC – 2*NN, если Z=0	Вычитание единицы и ветвление
JMP	0001DD		PC←(d)	Безусловный переход
JSR	004RDD		↑(SP) ←R, R←PC, PC←(d)	Обращение к подпрограмме
RTS	00020		PC←R, R←(SP)↓	Возврат из подпрограммы
MARK	0064NN		(SP) ←PC+2*NN, PC←R5, R5←(SP)↑	Восстановление стека
Команды прерывания программы				
EMT	104000-104377	ССП←(32)	↑(SP) ←ССП, ↑(SP)←PC, PC←(30)	Командное прерывание для системных программ
TRAP	104400-104777	ССП←(36)	↑(SP) ←ССП, ↑(SP)←PC, PC←(34)	Командное прерывание
IOT	000004	ССП←(22)	↑(SP) ←ССП, ↑(SP)←PC, PC←(20)	Командное прерывание для ввода-вывода
BPT	000003	ССП←(16)	↑(SP) ←ССП, ↑(SP)←PC, PC←(14)	Командное прерывание для отладки

Окончание приложения Б.

1	2	3	4	5
RTI	000002	ССП ← (SP)↓	PC ← (SP)↓	Возврат из прерывания
RTT	000006	ССП ← (SP)↓	PC ← (SP)↓	Возврат из прерывания
Команды управления машиной				
HALT	000000			Останов
WAIT	000001			Ожидание прерывания
Команды изменения признаков				
CLN	000250	0 - - -		Очистка N
CLZ	000244	- 0 - -		Очистка Z
CLV	000242	- - 0 -		Очистка V
CLC	000241	- - - 0		Очистка C
CCC	000257	0 0 0 0		Очистка N,Z,V,C
SEN	000270	1 - - -		Установка N
SEZ	000264	- 1 - -		Установка Z
SEV	000262	- - 1 -		Установка V
SEC	000261	- - - 1		Установка C
SCC	000277	1 1 1 1		Установка N,Z,V,C

Примечание. R – регистр общего назначения; (s) – операнд источника; (d) – операнд приемника; XXX – смещение (8 разрядов); NN – смещение(6 разрядов); ( ) – содержимое ячейки; ← – становится равным; ↑ – запись в стек; ↓ – выборка из стека; B – байтовая команда; ∧ – логическое умножение ("И"); ∨ – логическое сложение ("ИЛИ"); ∀ – исключающее ИЛИ; ($\underline{d}$) – отрицание d; ↔ – сдвиг влево (вправо); n – номер разряда; ССП – слово состояния процессора.

Приложение В.

Основные директивы ассемблера

Директива	Аргументы	Назначение
.ASCII	/string/	Генерирует блок данных, содержащий последовательность кодов символов (байт) строки string
.ASCIZ	/string/	То же что и ASCII, но с добавлением нулевого байта в конце строки
.BLKB	exp	Резервирует память для байтового блока данных; длина блока определяется выражением exp, которое в простейшем случае определено числом
.BLKW	exp	Резервирует память для словного блока данных; длина блока определяется выражением exp
.BYTE	exp1,…expN	Записывает в память последовательность байт, определенных выражениями exp1,…expN или непосредственно записанных в виде числовых значений

Окончание приложения В.

.END	нет	Указывает конец исходной программы
.ENDM	нет	Указывает на конец блока повторений или макроопределения
.EVEN	нет	Обеспечивает выравнивание счетчика адреса программы до четного адреса
.GLOBL	name1,..nameN	Определяет имена name1,..nameN как глобальные
.LIST	нет	Вызывает распечатку исходной программы, объектного кода и таблицы символов (используется ассемблером по умолчанию)
.MACRO	name, list	Начинает определение макроиструкции
.NLIST		Запрещает генерацию листинга исходной программы
.ODD		Обеспечивает нечетность счетчика адреса программы
.REPT	n	Начинает блок повторений, который генерирует n копий кода программы следующей за директивой .REPT до директивы .ENDM
.TITLE	name	Задает имя name объектной программе, воспринимается транслятор как комментарий
.WORD	exp1,…expN	Записывает в память последовательность слов, определенных выражениями exp1,…expN или непосредственно записанных в виде числовых значений

Приложение Г.

Набор специальных символов языка ассемблера

Символ	Интерпретация ассемблера
:	Ограничитель метки
::	Ограничитель метки, определяющий ее как глобальный символ
=	Операция прямого присваивания
= =	Операция прямого присваивания, определяющая символ как глобальный
#	Индикатор непосредственной адресации
@#	Индикатор абсолютной адресации
@	Индикатор косвенной адресации
(	Индикатор начала терма регистра
)	Индикатор конца терма регистра
,	Разделитель поля операнда
;	Индикатор начала поля комментария
<	Индикатор начала аргумента или выражения
>	Индикатор конца аргумента или выражения
+	Знак операции арифметического сложения или автоувеличения в машинных инструкциях
-	Знак операции арифметического вычитания или автоуменьшения в машинных инструкциях
*	Знак операции арифметического умножения
/	Знак операции арифметического деления

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Микропроцессоры в 3 кн. Кн. 1. Архитектура и проектирование микроЭВМ. Организация вычислительных процессов : учеб. для вузов/ П. И. Нестерев, В. Ф. Шаньгин, В. Д. Горбунов и др.; под ред. Л. Н. Преснухина. – М. : Высш. шк., 1988. – 495 с.

2. Микропроцессоры в 3 кн. Кн. 2. Средства сопряжения. Контролирующие и информационные системы : учеб. для вузов/ В. Д. Вернер, В. В. Воробьев, А.В.Горячев и др.; под ред. Л.Н.Преснухина. – М. : Высш. шк., 1988. – 383 с.

3. Уокерли, Дж. Архитектура и программирование микроЭВМ: В 2 книгах. Пер. с англ./ Дж. Уокерли – М. : Мир, 1984. – Кн.2, 341 с.

4. Задков, В. Н. Компьютер в эксперементе : Архитектура и программные средства систем автоматизации/ В. Н. Задков, Ю. В. Пономарев. – .М. : Наука, 1988. – 376 с.

5. Микропроцессорные средства производственных систем/ В. Н. Алексеев, А. В. Коновалов, В. Г. Колосов и др.; под общ. ред. В. Г. Колосова. – Л. : Машиностроение, 1988. – 287 с.

6. Системы программного управления промышленными установками и робототехническими комплексами : учебн. пос. для вузов/ Б. Г. Коровьев, Г. И. Прокофьев, Л. Н. Рассудов. – Л. : Энергоатомиздат, 1990. – 352 с.

7. Микропроцессорные системы автоматического управления / В. А. Бессекерский, Н. Б. Ефимов, С. И. Зиатдинов и др.; под общ. ред. В. А. Бессекерского. – Л. : Машиностроение, 1988. – 365 с.

8. Иванов, В. М. Основы D- и Z-преобразований : конспект лекций/ В. М. Иванов. – Ульяновск, 2001. – 36 с.

9. Иванов, В.М. Микропроцессорные системы управления электроприводом : методические указания и задания к курсовой работе/ В. М. Иванов. – Ульяновск, 2001. – 36 с.

10. Управляющие вычислительные комплексы : учебное пособие / Н. Л. Прохоров, Г. А. Егоров, В. Е. Красовский и др. – М.: Финансы и статистика, 2003. – 352 с.

Учебное издание

МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ УПРАВЛЕНИЯ ЭЛЕКТРОПРИВОДОМ

Методические указания к лабораторным работам

Составитель ИВАНОВ Владимир Михайлович

Редактор *Н. А. Евдокимова*
Подписано в печать 21.06.2007. Формат 60×84/16. Бумага офсетная.
Печать трафаретная. Усл. печ. л. 2,10.
Тираж 100 экз. Заказ
Ульяновский государственный технический университет
432027, Ульяновск, Сев. Венец, 32.
Типография УлГТУ, 432027, Ульяновск, Сев. Венец, 32.