Федеральное агентство по образованию

Государственное образовательное учреждение
высшего профессионального образования

Владимирский государственный университет

Кафедра вычислительной техники

ВИРТУАЛЬНАЯ ЭВМ

Методические указания к лабораторным работам

Составитель
В.Б. БУЛАНКИН

Владимир 2006

УДК 004.3
ББК 32.972
В52

Рецензент
Кандидат технических наук, доцент
кафедры управления и информатики в технических
и экономических системах Владимирского
государственного университета
А.Б. Градусов

Печатается по решению редакционно-издательского совета
Владимирского государственного университета

Виртуальная ЭВМ : метод. указания к лаб. работам / Владим. гос. ун-т. ; сост. В. Б. Буланкин. – Владимир : Изд-во Владим. гос. ун-та, 2006. – 48 с.

В52

Посвящены исследованию выполнения цикла работы процессора в ЭВМ с микропрограммным управлением. Описываются операционный блок и блок микропрограммного управления. Рассматриваются принципы эмуляции ЭВМ на уровне микропрограммного и программного управления, микропрограммирование команд СМ ЭВМ.

Предназначены для студентов специальности 220100 – вычислительные машины, комплексы, системы и сети, а также могут быть полезны студентам других специальностей, изучающим архитектуру и детальную организацию ЭВМ.

Табл. 22. Ил. 6. Библиогр.: 4 назв.

УДК 004.3
ББК 32.972

ВВЕДЕНИЕ

Микроминиатюризация средств вычислительной техники, когда процессор и другие компоненты ЭВМ изготовлены на кристалле в форме интегральных схем с ограниченным числом выводов интерфейса и ограниченными возможностями доступа ко многим не всегда представленным внутренним ресурсам, не позволяет в полном объеме познакомиться с организацией и тонкостями работы всех узлов и блоков.

Подобные проблемы можно исключить, перейдя к аппаратным, а чаще программным моделям, которые с достаточной степенью детализации соответствуют реальным объектам. Организация модели дает возможность не только изучить работу моделируемого объекта и закрепить практические навыки правильной работы с ним, но и сосредоточить внимание на отдельных деталях, требующих более глубокого и многовариантного рассмотрения.

В представленных методических указаниях описана программная модель гипотетической ЭВМ с архитектурой операционного и управляющего автоматов, соответствующих архитектуре элементной базы серии 1804,

используемой в компьютерах PDP фирм DEC, СМ ЭВМ, «Мера», «Электроника» и др. Данная работа ставит задачу наряду с другими познакомить студентов с системой команд вычислительных машин данного класса.

Тематически методические указания разделены на четыре лабораторных работы, из которых первая посвящена описанию структурной схемы операционного блока ЭВМ, функций, реализуемых составляющими его устройствами, принципов управления выполнением этих функций, составлению и отладке микропрограмм; вторая – описанию структурной схемы и функций блока микропрограммного управления, составлению и отладке микропрограмм обработки данных, содержащихся в оперативной памяти с различными вариантами организации вычислительного процесса; третья – описанию принципов программной эмуляции цикла работы процессора ЭВМ с заданной системой команд; четвертая – описанию способов адресации данных и программирования задач на языке машинных команд, раскрываемых до уровня микроопераций.

Лабораторная работа № 1

ОПЕРАЦИОННЫЙ БЛОК МИКРОПРОГРАММИРУЕМОГО ПРОЦЕССОРА

Цель работы: изучение операционного блока на уровне структурной схемы, ознакомление с составом микрокоманд и порядком их выполнения, составление и отладка микропрограмм.

1. Порядок выполнения работы

1.1. Знакомство с моделью процессора и системой микрокоманд

По материалам пп. 2, 3 изучить общую структуру процессора, состав операционного блока, структуру микрокоманды (МК) и функции ее полей.

1.2. Ввод и выполнение микропрограммы

По материалам п. 4 ознакомиться с порядком ввода микропрограмм и исходных данных. Ввести описание микропрограммы нахождения наименьшего числа, приведенное в п. 5, и исходные данные в регистры *RG*1, *RG*2, *RG*3.

Выполнить микропрограмму в режиме МИКРОКОМАНДА, записывая состояния элементов модели по информации на экране. Можно ограничиться фиксацией состояний счетчика микрокоманд (СМК), регистрового запоминающего устройства (ЗУ) и регистров флажков. При засветке указателя СТОП восстановить исходное состояние, установив СМК, указатель адреса МК *MUAD*, регистр результата *RG*4 и признак *STOP* в состояние 0. Установить режим АВТОМАТ. Повторить выполнение микропрограммы с разными вариантами исходных данных.

1.3. Микропрограммирование алгоритмов

Составить, ввести и выполнить микропрограмму для одной из задач п. 8. Решая задачу помикрокомандно, записать и проанализировать изменения состояний основных элементов процессора.

2. Структура процессора

2.1. Состав процессора

В процессор входят операционный блок, блок микропрограммного управления, оперативная память (ОП) и микропрограммная память МП. Он может использоваться для выполнения алгоритмов в форме микропрограмм, а также для эмуляции операций в системе команд СМ ЭВМ. В последнем случае для выполнения каждой команды вызывается соответствующая микропрограмма.

ОП содержит 8192 16-разрядных слов (16384 байта) с шестнадцатиричными адресами 0-*DFFF*. МП содержит 64 МК по 64 бита с адресами 0-3*F*. ОП связана с операционным блоком через регистры чтения *RGR*, записи *RGW* и адресный регистр *AOP*. С выхода *RGR* возможны передачи в регистр команд RGK, адресный регистр, на шину *DB*, как код одного из операндов, и на шину Y для записи в регистровое ЗУ.

Блок микропрограммного управления содержит 64-разрядный регистр микрокоманд *RG*МК и схему управления следующим адресом. Поле константы МК является одним из возможных источников операндов и подключается к шине *DA* операционного блока.

2.2. Операционный блок

Схема операционного блока приведена на рис. 1. Блок может быть построен на четырех микропроцессорных секциях МПС 1804ВС2 с некоторыми дополнительными узлами. В него входят блоки: внутренней памяти (БВП), арифметико-логический (БАЛ), рабочего регистра (БР), управления БУ.

1. БВП включает в себя регистровое запоминающее устройство (ЗУ), предназначенное для хранения операндов и результатов выполнения операций мультиплексор адреса МАРЗУ, регистры *RGA* и *RGB*. 16 регистров РЗУ обозначаются *RGO*, *RG*1,..., *RGF*. Регистры *RG*6 и *RG*7 в режиме эмуляции СМ ЭВМ используются как указатель стека *SP* и программный

счетчик *PC*. Чтение РЗУ происходит одновременно по адресам *A* и *B* в регистры *RGA* и *RGB* соответственно. Запись выполняется по адресу *B*.

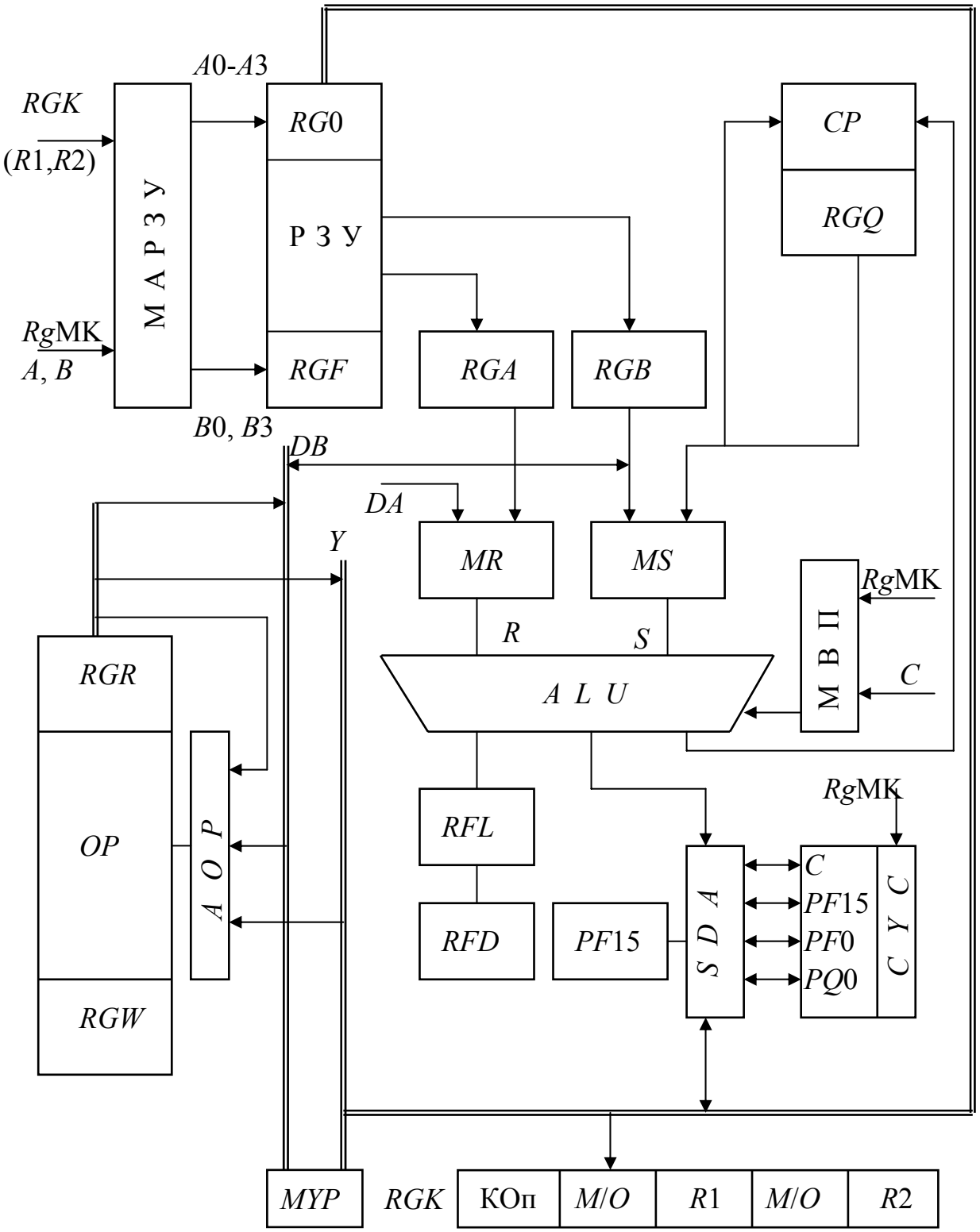

Рис. 1. Схема операционного автомата

Адреса *A* и *B* поступают на РЗУ из одноименных полей МК или (в режиме эмуляции) из полей *R*l, *R*2 *RGK*. Выбор источника адреса осуществляется схемой МАРЗУ под управлением полей *MA*, *MB* микрокоманды.

2. БАЛ включает в себя мультиплексоры операндов *MR* и *MS*, управляемые полем *SRC* МК, арифметико-логическое устройство *ALU*, сдвигатель *SDA* и регистры флажков *RFL*, *RFD*. *ALU* и *SDA* управляются соответственно полями *ALU* и *SH* МК.

Флажки (признаки результата) формируются и хранятся только на время исполнения текущей МК. При необходимости они переписываются в регистр длительного хранения *RFD*. Обозначения флажков: *N* – знак минуса (старший бит результата), *Z* – признак нуля, *V* – признак переполнения, *C* – перенос из старшего бита, одноразрядные регистры *PF*0, *PF*15 служат для фиксации спадающих битов при сдвигах. В *PF*0 может формироваться также признак паритета.

3. Блок результата (БР) состоит из рабочего регистра *RGQ* и сдвигателя *CP* и используется в операциях умножения, деления, двойного сдвига, а также в качестве аккумулятора. *RGQ* служит одним из источников операнда *S*. Запись на него может производиться со сдвигом вправо, влево или без сдвига.

4. Блок управления (БУ) включает в себя мультиплексор входного переноса (МВП), схему управления сдвигом (СУС) и мультиплексор управления результатом (МУР), управляемые соответственно полями *CC*, *CSH* и *DST* микрокоманды.

3. Структура микрокоманды и описание полей

3.1. Структура МК

МК содержит 64 бита и включает в себя 21 поле (рис. 2).

A	*B*	*MA*	*MB*	*R*	*W*	*SRC*	*SH*	*ALU*	*CCX*	*CSH*	*WQ*	*DBA*	*F*	*DST*	
0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	

	JFI	*CC*	*CHA*	*WA*	*C*0	*CONST*
	0	0	E	1	1	0000

Рис. 2. Структура микрокоманды:
верхняя строка – мнемоническое обозначение поля;
нижняя строка – значение по умолчанию

Для каждого поля указаны его мнемоника, порядковый номер и значение по умолчанию в шестнадцатиричном поле. Значение поля по умол-

чанию устанавливается автоматически и соответствует определенной функции. Если эта функция устраивает микропрограммиста, он может не описывать поле в микропрограмме.

3.2. Описание полей

Поля *A* и *B* задают адреса регистров РЗУ. По умолчанию *A* = *B* = 0. Поля *MA*, *MB* управляют выбором адресов РЗУ из МК или регистра команд в соответствии с табл. 1. По умолчанию *MA* = *MB* = 0 (выбор адресов из *МК).*

Таблица 1

Управление адресом РЗУ

Поле *MA*/*MB*	Источник адреса
0	Поле *A*/*B* МК
1	*R*1
2	*R*2
3	*R*3

Поля *R* и *W* задают соответственно чтение из ОП в регистр чтения *RGR* и запись в ОП из регистра записи *RGW*. По умолчанию *R* = *W* = 0 – обращение к памяти отсутствует.

Поле *SRC* управляет выбором источников операндов в соответствии с табл. 2. По умолчанию *SRC* = 0 (операнды выбираются из *RGA* и *RGB*).

Таблица 2

Выбор источников операндов

Поле *SRC*	Операнд *R*	Операнд *S*
0	*RGA*	*PGB*
1	*RGA*	*RGQ*
2	*RGA*	*DB*
3	*RGA*	*RGQ*
4	*DA*	*RGB*
5	*DA*	*RGQ*
6	*DA*	*DB*
7	*DA*	*RGQ*

Поле *SH* управляет работой сдвигателей *SDA* и *CP* в соответствии с табл. 3 при ненулевом значении поля *ALU* или младшего бита поля *SRC*. В противном случае поле управляет специальными функциями *ALU*, которые

в данной работе не используются. По умолчанию *SH* = 6, что соответствует передаче результата с выхода *ALU* на шину *Y* без сдвига.

Поле *ALU* управляет простыми операциями *ALU* (табл. 4). По умолчанию *ALU* = 6 – сложение операнда *R* с входным переносом *C*0. Значение *C*0 определяется полем *CCX* и выбирается из МК или из бита *C RFD* согласно табл. 5.

Таблица.3

Управление с двигателем

Поле *SH*	Операция	*PF*15	*PF*0
0	АС *ALU* вправо	Вход	*F*0
1	ЛС *ALU* вправо	Вход	*F*0
2	АС *ALU*, ЛС *RGQ* вправо	Вход	*F*0
3	ЛС *ALU*, ЛС *RGQ* вправо	Вход	*F*0
4	Без сдвига	Вход	*P*
5	ЛС *RGQ* вправо	Вход	*P*
6	*RGQ*:=*ALU*	Вход	*P*
7	*RGQ*:=*ALO*	Вход	*P*
8	АС *ALU* влево	*F*14	Вход
9	ЛС *ALU* влево	*F*15	Вход
A	АС *ALU*, ЛС *RGQ* влево	*F*14	Вход
B	ЛС *ALU*, ЛС *RGQ* влево	*F*15	Вход
C	Без сдвига	*F*15	–
D	ЛС *RGQ* влево	*F*15	–
E	Расширение знака (см. прим. 2)	–	Вход
F	Без сдвига	*F*15	–

Примечания: 1. ЛС, АС – логический и арифметический сдвиги соответственно, *P* – паритет, *F*0, *F*14, *F*15 – выходы разрядов *ALU*. 2. Биты 8 – 15 (старший байт) устанавливаются равными биту 7 (знаковый бит младшего байта).

Таблица 4

Управление входным переносом

Поле *CCX*	*CO*
0	0
1	1
2	*C*
3	*C*

Поле *CSH* обеспечивает операции сдвига, которые не могут быть полностью заданы полем *SH*. К ним относятся кольцевые сдвиги вправо и влево через бит переноса *C* (*ROR*, *ROL*), а также арифметические сдвиги с записью спадающего бита в бит *C*

(*ASR*, *ASL*, *ASH*, *ASHC*). При этом в поле *SH* должен бить закодирован логический или арифметический сдвиг в нужном направлении. Функции поля *CSH* описываются в табл. 6.

Таблица 5

Простые операции ALU

Поле *ALU*	Операция *ALU*	Флажки				Поле *ALU*	Операция *ALU*	Флажки			
		N	*Z*	*V*	*C*			*N*	*Z*	*V*	*C*
0	На всех выходах "1"	1	0	0	0	8	На всех выходах "0"	0	1	0	0
1	*S*-*R*-1+*C*0	+	+	+	+	9	*R* 8 *S*	+	+	0	0
2	*R*-*S*-1-*C*0	+	+	+	+	*A*	*R* + *S*	+	+	0	0
3	*R*+S+C0	+	+	+	+	*B*	*R* + *S*	+	+	0	0
4	*S*+*C*0	+	+	+	+	*C*	*R* & *S*	+	+	0	0
5	*S*+*C*0	+	+	+	+	*D*	*R* v *S*	+	+	0	0
6	*R*+*C*0	+	+	+	+	*E*	*R* & *S*	+	+	0	0
7	*R*+*C*0	+	+	+	+	*F*	*R* v *S*	+	+	0	0

Примечание. Символ "+" в поле флажка означает установку этого флажка в соответствии с результатом операции.

Таблица 6

Управление сдвигом

Поле *CSH*	Сдвиг	Примечание
0	–	В управлении сдвигом не участвует
1	*ROR*	Кольцевой сдвиг вправо
2	*ASR, ASH*	Арифметический сдвиг вправо
3	*ASHC*	Двойной сдвиг вправо
4	–	В управлении сдвигом не участвует
5	*ROL*	Кольцевой сдвиг влево
6	*ASL, ASH*	Арифметический сдвиг влево
7	*ASHC*	Двойной сдвиг влево

Поле *WQ* управляет записью в рабочий регистр *RGQ*. По умолчанию поле равно нулю – запись разрешена.

Поле *DBA*. При *DBA* = 1 содержимое *RGB* через шину *DB* передается в адресный регистр оперативной памяти *AOP*.

Поле фиксации флажков *F*. При *F* = 1 текущее значение флажков запоминается в *RFD*.

Поле *DST* управляет записью результата операции. С выхода *SDA* результат может записываться в РЗУ по адресу *B*, в регистр записи *RGW*, в адресный регистр *AOP*. Возможны передачи в РЗУ и *AOP* из *RGR*. Функции поля описаны в табл. 7. По умолчанию *DST* = 0 – запись в РЗУ и *RGW*.

Таблица 7

Управление записью результата

Поле *DST*	Источник	Приемники
0	*SDA*	РЗУ, *RGW*
1	*SDA*	РЗУ, *AOP*
2	*SDA*	*RGW*
3	*SDA*	*AOP RGW*
4	*RGR*	РЗУ
5	*RGR*	РЗУ, *AOP*
6	–	–
7	*RGR*	*AOP*

Поле *JFI* участвует совместно с полем *CC* в формировании условий перехода. Старший бит *J* = 1 – признак безусловного перехода, бит *F* = 1 указывает, что условие перехода определяется флажками из *RFD*, бит *I* = 1 означает инверсию формируемого условия. При *JFI* = 101 фиксируется останов по команде СТОП. По умолчанию *JFI* = 0.

Поле *CC* управляет формированием условий перехода в соответствии с табл. 8.

Таблица 8

Формирование условий перехода

Поле *CC*	Команды ветвления	Условие перехода
0	Переход по паритету	$PF0 = 1$
1	*BEQ*, *BNE* *	$Z = 1$
2	*BMI*, *BPL* *	$N = 1$
3	*BVS*, *BVC* *	$V = 1$
4	*BCS*, *BCC* *, *BHIS* *, *BLO*	$C = 1$
5	*BLT*, *BGE* *	$N+V = 1$
6	*BLE*, *BGT* *	Z v $(N+V) = 1$
7	*BLOS*, *BHI**	C v $Z = 1$

Обозначения переходов соответствуют командам ветвления СМ ЭВМ. При микропрограммировании переходов, отмеченных звездочкой, устанавливается бит инверсии *I* = 1 в поле *JFI*.

Поле СНА обеспечивает формирование адреса следующей МК. В данной работе используются только две функции этого поля – "Продолжить" *CONT* с кодом *E*, выполняемая по умолчанию, и "Условный переход по адресу из МК" *CJP* с кодом 3. Для кодирования безусловного перехода записывается код 3 в СНА и код 4 в *JFI*.

Поле *WA* управляет записью в регистр счетчик адреса (*RACT*) из поля *CONST* МК, которая происходит при *WA* = 1. По умолчанию *WA* = 0, т.е. записи нет.

Поле *CONST* содержит 16-битовую константу, подключаемую к шине *DA* операционного блока, или 6-разрядный адрес перехода, подаваемый на шину *DA* блока микропрограммного управления. По умолчанию *CONST* = 0.

4. Диалог пользователя с моделью

При запуске модели на исполнение на экране появляется основное меню (табл. 9). Требуемая функция выполняется после установки указателя на соответствующую функцию и нажатия клавиши <*Enter*>. При этом для функций 1, 2, 3, 4, 5 вызывается внутреннее меню.

Таблица 9

Основное меню модели

Номер функции	Функция
1	Справка
2	Режим исполнения
3	Режим трассировки
4	Микропрограммная память
5	Чтение-запись данных
6	Таблица преобразования адресов
7	Исполнение
8	Выход

Режим исполнения можно установить в трех вариантах: МИКРОКОМАНДА (код 1), КОМАНДА (код 2), АВТОМАТ (код 3). По умолчанию устанавливается режим МИКРОКОМАНДА. РЕЖИМ ТРАССИРОВКИ (индикация состояния после заданного шага моделирования при работе

в режиме АВТОМАТ) возможен в вариантах БЕЗ ТРАССИРОВКИ (0), КОМАНДА (1), МИКРОКОМАНДА (2). В последних двух случаях пользователь задает также величину задержки после каждого шага. По умолчанию устанавливается режим БЕЗ ТРАССИРОВКИ.

Функция ИСПОЛНЕНИЕ запускает модель на исполнение программы. Если программа выполняется в режиме МИКРОКОМАНДА (КОМАНДА), то после каждой микрокоманды (команды) задача будет останавливаться и для продолжения нужно нажать любую клавишу, а для выхода в основное меню – клавишу <ESC>.

Функция ВЫХОД позволяет закончить работу.

Подробнее рассмотрим функцию ЧТЕНИЕ-ЗАПИСЬ ДАННЫХ. Ее внутреннее меню содержит функции ЧТЕНИЕ-ЗАПИСЬ РЕГИСТРОВ (1), ЧТЕНИЕ-ЗАПИСЬ ПАМЯТИ (2), ЧТЕНИЕ МАССИВА ЯЧЕЕК (3), ВЫХОД (4).После выполнения функций 1, 2, 3 происходит возврат в это же меню, после функции 4 – возврат в основное меню. При задании функции ЧТЕНИЕ-ЗАПИСЬ РЕГИСТРОВ выводится таблица, в которой указаны все регистры с соответствующими номерами, а также содержимое каждого регистра в шестнадцатеричном коде. Для ввода нового значения указатель устанавливается на значение соответствующего регистра, затем вводится само значение и нажимается <Enter>. Для выхода из режима нужно нажать клавишу <ESC>.

В режиме ЧТЕНИЕ МАССИВА ЯЧЕЕК вводятся основание системы счисления представления адреса, начальный адрес и длина массива. На экране отображается таблица с заданным количеством слов с их адресами и значениями соответственно в шестнадцатеричном и восьмеричном кодах.

В режиме ЧТЕНИЕ-ЗАПИСЬ ПАМЯТИ вводятся основание системы счисления адреса, начальный адрес и длина пассива. На экране появляются адрес и значение первого слова массива. Для изменения этого значения нужно ввести новое. Переход к следующему слову – по нажатию <Enter>. По окончании чтения-записи заданного числа слов запрашиваются параметры следующего массива. Для выхода из режима набирается длина массива 0.

Функция МИКРОПРОГРАММНАЯ ПАМЯТЬ содержит следующее внутреннее меню: ЧТЕНИЕ-ЗАПИСЬ МП (1), ПРОСМОТР МП (2), ЧТЕНИЕ ИЗ ФАЙЛА (3), ЗАПИСЬ В ФАЙЛ (4), ВЫХОД (5).

Для функции ЧТЕНИЕ-ЗАПИСЬ МП вводятся начальный адрес МП (шестнадцатеричный код) и длина МП (десятичный код). На экране появляется таблица, строки которой – микрокоманды, а столбцы – адрес и поля микрокоманды. Значения всех полей первоначально установлены по умолчанию. Для изменения значения поля нужно ввести номер поля и нажать клавишу <Enter>, а затем новое значение нуля (в шестнадцатеричном коде) и <Enter>. Если значение поля некорректно, то выдается соответствующее сообщение и ввод необходимо повторить. Для перехода к следующей микрокоманде ввести код 55. По окончании чтения-записи заданного числа МК задается вопрос: "Продолжить? [*Y*/*N*]". Если "*Y*", то вводятся новые параметры, если "*N*" – выход в промежуточное меню.

Функция ПРОСМОТР МП подобна предыдущей, но здесь можно только посмотреть состояние *MPM*.

Функция ЗАПИСЬ В ФАЙЛ МП используется для сохранения содержимого МП. Для этого на запрос "Введите имя файла" вводится имя файла вида: *BRnn.xxx*, где *nn* – номер бригады, а *xxx* – номер группы.

Функция ЧТЕНИЕ ИЗ ФАЙЛА МП используется для восстановления ранее сохраненного состояния МП из заранее подготовленного файла.

ТАБЛИЦА ПРЕОБРАЗОВАНИЯ АДРЕСОВ в данной работе не используется и будет описана ниже.

5. Микропрограммирование алгоритмов

5.1. Общий порядок микропрограммирования

Микропрограммирование начинается с составления содержательной схемы алгоритма. Затем распределяется оперативная и регистровая память для хранения и преобразования данных. Далее при наличии языка микропрограммирования составляется микропрограмма на этом языке и транслируется в машинный код.

В нашем случае язык и средства трансляции отсутствуют, в связи с чем рекомендуется такой порядок микропрограммирования:

– написать программу выполнения алгоритма в терминах языка ассемблера СМ ЭВМ или другого известного вам языка;

– произвести интерпретацию команд программы микрокомандами процессора. При этом интерпретация одной команды может потребовать нескольких МК и, наоборот, одна МК может интерпретировать более чем одну команду. Интерпретирующие МК сводятся в таблицу, в которой указываются их адреса, интерпретируемые команды, символические обозначения и номера полей МК и их значения. Описания полей, значения которых устанавливаются по умолчанию, можно опустить. Адреса МК и значения полей записываются в шестнадцатеричном коде.

5.2. Пример микропрограммирования

На рис. 3 приведена схема алгоритма поиска наименьшего элемента *M* из трех чисел *A*, *B*, *C*.

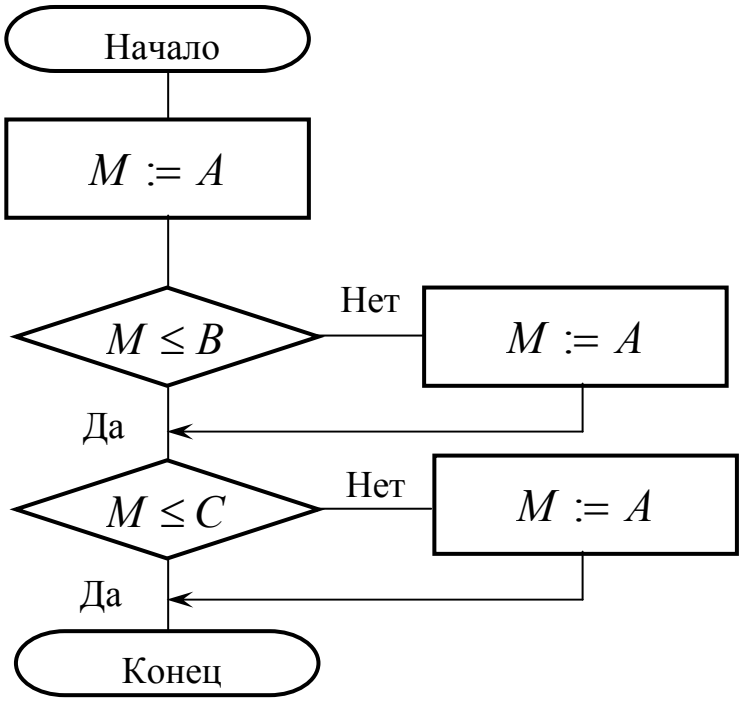

Рис. 3. Схема алгоритма поиска наименьшего элемента M

Программа на языке ассемблера и ее табличное представление приведены в табл. 10. Предполагается, что исходные данные загружены в регистры *R*l, *R*2, *R*3, а значение *M* фиксируется в *R*4. Заметьте, что пары микрокоманд сравнения и перехода интерпретируются одной микрокомандой.

Обратите также внимание на то, что для определения условия перехода в МК 2 использован дополнительный регистр флажков *RFD*.

Таблица 10

Микропрограмма поиска наименьшего числа

Адрес МК	Операция	Поле	Номер	Значение	Функция
0	*Minel* :	*A*	1	1	*R*l
	*MOV R*l, *R*4	*B*	2	4	*R*4
1	*CMP R*4, *R*2	*A*	1	4	*R*4
	*BLE Min*1	*B*	2	2	*R*2
		ALU	9	2	*R* – *S* – 1 + *C*0
		CCX	10	1	*C*0 = 1
		DST	15	6	Без записи
		CC	17	6	*BLE*
		CHA	18	3	Усл. переход
		CONST	21	3	Адрес перехода
2	*MOV R*2, *R*4	*A*	1	2	*R*2
		8	2	4	*R*4
3	*Min*1 :	*A*	1	4	*R*4
	*CMP R*4, *R*3	*B*	2	3	*R*3
	*BLE Min*2	*ALU*	9	2	*R* – *S* – 1 + *C*0
		CCX	10	1	*C*0 = 1
		DST	15	6	Без записи
		CHA	18	3	Усл. переход
		CC	17	6	*BLE*
		CONST	21	5	Адрес перехода
4	*MOV R*3, *R*4	*A*	1	3	*R*3
		B	2	4	*R*4
5	*Min*2: *HALT*	*JFI*	16	5	*STOP*

6. Содержание отчета

В отчет входят схема алгоритма, программа и микропрограмма выполняемого алгоритма по форме табл. 10, значения исходных данных и ре-

зультатов, протокол изменения состояния основных элементов модели выводимых на экран в режиме МИКРОКОМАНДА.

7. Варианты заданий на составление микропрограмм

1. Найти число Фибоначчи с номером К. Первые два числа Фибоначчи равны 1. Каждое последующее число равно сумме двух предыдущих. Найти сумму первых К чисел ряда.

2. Задано целое число М. Определить, является ли М числом Фибоначчи. Если да, то результатом будет его порядковый номер в ряду таких чисел. В противном случае результат равен нулю.

3. Найти сумму квадратов натуральных чисел от 1 до К. При вычислении квадрата очередного числа использовать формулу

$$(x+1)^2 = x^2+2x+1.$$

4. Составить программу умножения двух положительных чисел.

5. Найти наибольший общий делитель двух чисел по алгоритму Евклида.

6. Преобразовать 4-разрядное положительное десятичное число в двоичное. Преобразование производится по формуле

$$B = ((D1 \cdot 10 + D2) \cdot 10 + D3) \cdot 10 + D4,$$

где $D1$ – $D4$ – тетрады числа, начиная со старшей. Для выделений тетрад использовать операцию двойного сдвига, для умножения на 10 – сдвиг и сложение сдвинутых кодов.

7. Составить программу нормализации числа со знаком: арифметический сдвиг влево до появления единицы (для отрицательных чисел – до появления нуля) в старшем бите, подсчет количества сдвигов и запись мантиссы и порядка в соседние регистры общего назначения (РОНы).

8. Определить количество единиц в коде числа.

9. Определить количество нулей в коде числа.

10. Дополнить число контрольным битом по четности. Контрольный бит – старший.

11. Дополнить число контрольным битом по нечетности. Контрольный бит – младший.

12. Определить остаток от деления:

– двоичного числа на 3;
– двоичного числа на 7;
– двоичного числа на 5;
– десятичного числа на 9;
– десятичного числа на 11.

8. Контрольные вопросы

1. Опишите структуру микрокоманды и функции ее полей.
2. Определите двоичный код МК по описанию ее полей.
3. Какие действия выполняются по МК, полностью сформированной по умолчанию?
4. Опишите выполнение операций сдвига и функции полей *SH*, *CSH*.
5. Как задаются в МК условные и безусловные переходы?

Лабораторная работа № 2

БЛОК МИКРОПРОГРАММНОГО УПРАВЛЕНИЯ. РАБОТА С ОПЕРАТИВНОЙ ПАМЯТЬЮ

Цель работы: изучение структуры и функций блока микропрограммного управления (БМУ), составление и отладка микропрограмм обработки данных, записанных в ОП, с использованием циклов и подпрограмм.

1. Порядок выполнения работы

1. По материалам п. 2 изучить структуру и функции БМУ и поля МК, управляющие его работой.

2. По материалам п. 3 ознакомиться со способами адресации данных.

3. Используя как образец микропрограмму, приведенную в п. 4, разработать и закодировать микропрограмму решения заданной задачи. Ввести микропрограмму и исходные данные. Отладить микропрограмму и решить задачу, проследив за порядком функционирования процессора при выполнении операций.

2. Блок микропрограммного управления

2.1. Структура блока

Схема блока микропрограммного управления (БМУ) представлена на рис. 4. В его состав входят:

– схема управления последовательностью микрокоманд (УПМ);

– микропрограммная память *МРМ* емкостью 64 64-разрядных МК;

– регистр микрокоманд *Rg*МК;
– мультиплексор кода условия (МКУ);
– инвертор кода условия (ИКУ);
– схема формирования признака *STOP*.

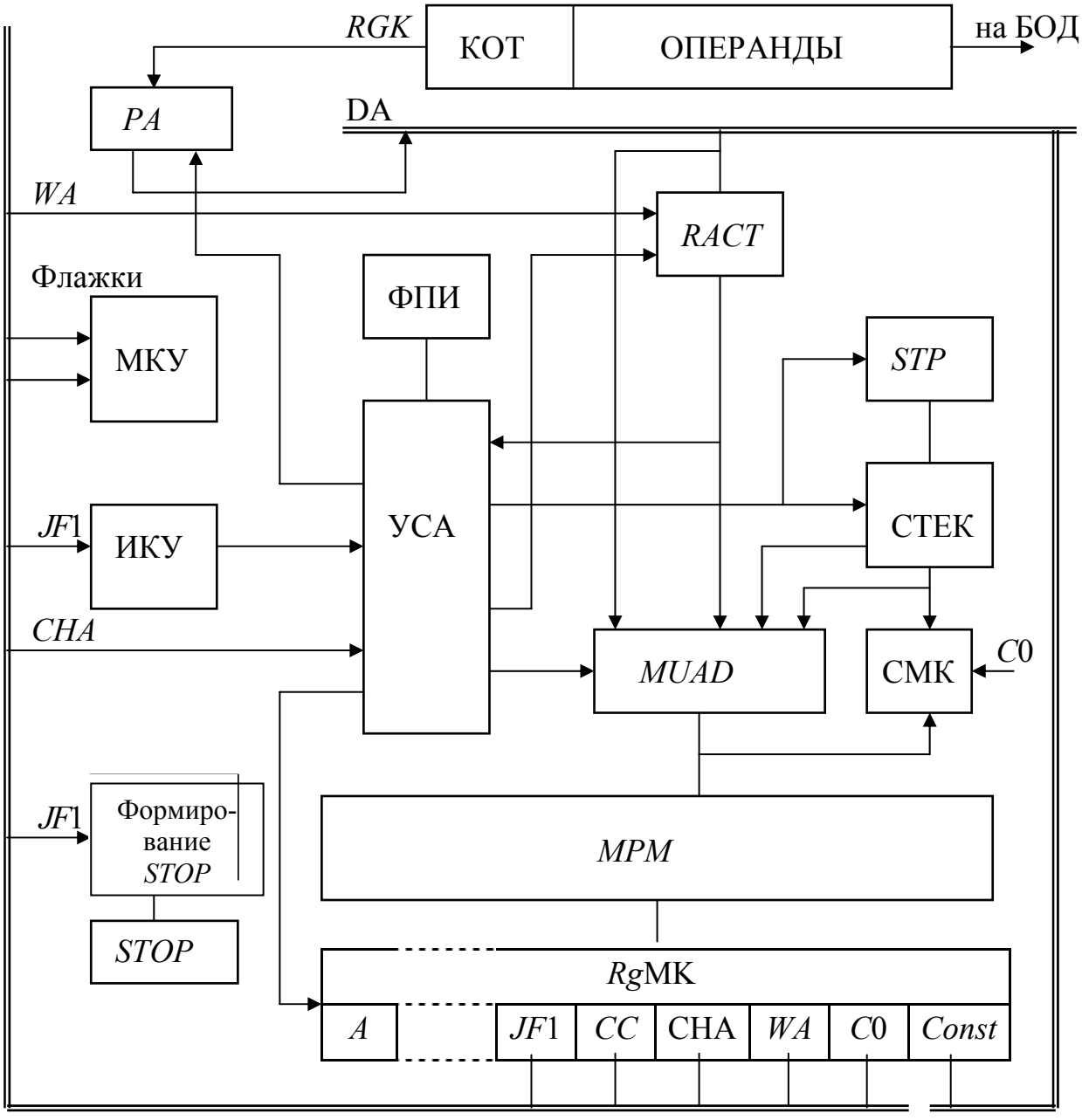

Рис. 4. Схема блока микропрограммного управления

УПМ имеет структуру микросхемы 1804ВУ4. Ее основная функция – формирование последовательности адресов МК и организация циклических вычислительных процессов. В УПМ можно выделить шесть основных

блоков: мультиплексор адреса *MUAD*, счетчик микрокоманд СМК, регистр счетчик-адреса *RACT*, аппаратный стек емкостью четыре слова с указателем стека *STP*, формирователь признака нуля ФПН и схему управления следующим адресом (УСА). Все элементы схемы имеют разрядность 12, что позволяет адресовать до 4 096 слов (хотя в работе используются только 64) или задавать до 4 096 циклов.

Мультиплексор адреса *MUAD* выбирает в качестве адреса содержимое СМК, стека, *RACT* или прямого входа адреса с шины *DA*. Адрес на эту шину может поступать из поля константы МК или с преобразователя начального адреса *PA*. Выход *MUAD* соединен с адресным входом *MPM* и входом СМК. СМК увеличивает показание на 1 при входном сигнале $C0 = 1$. При $C0 = 0$ одна и та же МК может быть выполнена любое число раз. Значение $C0$ задается в МК и по умолчанию равно 1.

Регистр счетчик-адреса *RACT* используется как буфер для записи и хранения адреса или числа циклов, поступающих с шины *DA*.

Аппаратный стек содержит четыре регистра и предназначен для хранения адресов возврата из подпрограмм и циклов. Возможны следующие режимы работы стека:

– очистка стека путем установки в нуль указателя стека *STP*;

– хранение – содержимое стека не изменяется, может выполняться чтение из регистра-накопителя стека, адресуемого *STP*;

– загрузка – инкремент *STP* и запись в стек;

– выталкивание – считывание из вершины стека и декремент *STP*.

Формирователь признака нуля вырабатывает внутренний управляющий сигнал *ZR*, если $RACT = 0$.

Схема управления следующим адресом преобразует внешние сигналы из поля СНА МК, МКУ и ФПН в управляющие сигналы, обеспечивающие формирование следующего адреса. Преобразователь *PA* применяется при эмуляции команд. В данной работе не используется.

Мультиплексор кода условия связан по входу с регистрами флажков *RFL*, *RFD* и триггером $FF0$ операционного блока. Выбор анализируемых флажков при формировании кода условия *X*, а также признака безусловного перехода выполняется полем *CC* МК и битами *J* и *F* поля *JFI*. ИКУ инвертирует сигнал *X*, если бит $I = 1$.

Схема формирования признака *STOP* устанавливает его при $JFI = 5$.

2.2. Набор инструкций схемы УПМ

Работа УПМ описывается табл. 11. Столбец *Y* соответствует адресу следующей МК, столбец *X* – значению проверяемого логического условия (*X* = 1, если условие выполняется, *X* = 0, если оно не выполняется). Рассмотрим подробнее выполняемые инструкции.

Таблица 11

Инструкции к схеме УПМ

Поле СНА	Мнемоника	*ZR*	При *X* = 0		При *X* = 1		*RACT*
			Y	Стек	*Y*	Стек	
0	*JZ*	-	0	Очистка	0	Очистка	Хранение
1	*CJS*	-	СМК	Хранение	*CONST*	Загрузка	Хранение
2	*JMAP*	-	*PA*	Хранение	*PA*	Хранение	Хранение
3	*CJP*	-	СМК	Хранение	*CONST*	Хранение	Хранение
4	*PUSH*	-	СМК	Загрузка	СМК	Загрузка	См. примечание
5	*JSPR*	-	*RACT*	Загрузка	*CONST*	Загрузка	Хранение
Б	*CJV*	-	Не используется				Хранение
7	*JSP*	-	*RACT*	Хранение	*CONST*	Хранение	Хранение
8	*RFCT*	0	Стек	Хранение	Стек	Хранение	Декремент
	RFCT	1	СМК	Выталкивание	СМК	Выталкивание	Хранение
9	*RPCT*	0	Стек	Хранение	*CONST*	Хранение	Декремент
	RPCT	1	СМК	Хранение	СМК	Хранение	Хранение
A	*CRTN*	-	СМК	Хранение	Стек	Выталкивание	Хранение
B	*CJPP*	-	СМК	Хранение	*CONST*	Выталкивание	Хранение
C	*LDCT*	-	СМК	Хранение	СМК	Хранение	Загрузка
D	*LOOP*	-	Стек	Хранение	СМК	Выталкивание	Хранение
E	*CONT*	-	СМК	Хранение	СМК	Хранение	Хранение
F	*TWB*		См. табл. 12				

Примечание. Если *X* = 0, то хранение, если *X* = 1, то загрузка.

JZ – управление передается МК с адресом 0, и происходит очистка стека путем установки *STP* в состояние 0.

CJS 0 – условный переход к подпрограмме. При *X* = 1 продвинутый адрес МК запоминается в стеке и происходит переход по адресу подпрограммы, заданному в поле *CONST*.

JMAP – переход по адресу из *PA*.

CJP – условный переход по адресу из поля *CONST* при *X* = 1.

PUSH – засылка в стек и условная загрузка счетчика. Продвинутый адрес МК запоминается в стеке. При *X* = 1 происходит также загрузка в *RACT* из поля *CONST*.

JSPR – условный переход к одной из двух подпрограмм. Продвинутый адрес запоминается в стеке. Происходит переход к подпрограмме по адресу из поля *CONST* при *X* = 1 или из *RACT* при X = 0.

JSP – переход по адресу из *CONST* при *X* = 1 или из *RACT* при *X* = 0.

RFCT и *RPCT* – повторение цикла (переход по счетчику). Если содержимое *RACT* не равно нулю, из него вычитается 1 и выполняется переход по адресу из стека (*RFCT*) или из *CONST* (*RPCT*). Если *RACT* = 0, следующая МК выбирается в естественном порядке, а для инструкции *RFCT* производится еще выталкивание из стека – декремент *STP*.

CRTN – условный возврат из подпрограммы. При *X* = 1 происходит переход по адресу из стека и декремент *STP*.

LDCT – загрузка счетчика и продолжение. Из поля *CONST* в *RACT* загружается адрес или число циклов. Загрузка в *RACT* может также выполняться при любой другой инструкции при нулевом значении поля *WA* микрокоманды. По умолчанию *WA* = 1.

CJPP – условный переход по адресу поля *CONST* и декремент *STP*.

LOOP – контроль конца цикла. При *X* = 0 – переход по адресу из вершины стека. При *X* = 1 – декремент *STP* и переход к следующей по порядку МК.

CONT – продолжить выборку МК в естественном порядке. Этот код устанавливается по умолчанию.

TWB – разветвление на три направления. В зависимости от сигналов *ZR* и *X* источником следующего адреса являются СМК, поле *CONST* или стек в соответствии с табл. 12.

Таблица 12

Ветвление на три направления

ZR	*X*	Источник адреса	Операция стека	Операция *RACT*
0	0	Стек	Хранение	Декремент
0	1	Стек	Выталкивание	Хранение
1	0	*CONST*	Выталкивание	Декремент
1	1	СМК	Выталкивание	Хранение

3. Работа с оперативной памятью

Исходные данные в ОП вводятся, как было описано в работе 1. Возможны просмотр и корректировка данных в приостановках после выполнения очередного шага моделирования.

Адресация ОП может выполняться различными способами, которые рассматриваются ниже. При этом шестнадцатеричный адрес не должен превышать *DFFE*.

Прямая адресация. Адрес слова ОП задается в поле *CONST* МК. Поскольку непосредственная связь между регистром МК и *AOP* отсутствует, загрузка адреса происходит через процессор по цепочке "шина *DA* – мультиплексор *R – ALU – SDA – AOP*" и требует отдельной МК с кодом 7 в поле *DST* и кодом 4 в поле *SRC*.

Косвенная регистровая адресация. Адрес хранится в одном из регистров РЗУ, куда он может быть загружен предварительно из поля *CONST* или сформирован каким-либо другим образом. Адрес регистра задается полем *B*. Содержимое регистра передается в *AOP* через шину *DB*, для чего в поле *DBA* нужно записать 1. Параллельно в *ALU* могут выполняться различные преобразования информации.

Автоинкрементная адресация. Адрес также передается в *AOP* через шину *DB*. Одновременно к содержимому регистра прибавляется 2. Это можно сделать, записав число 2 в поле *CONST* и указав шину *DA* в качестве источника операнда *R*.

Автодекрементная адресация. Из содержимого регистра вычитается 2, после чего оно передается в *AOP* с выхода *SDA*.

Косвенная адресация. Адрес ОП выбирается из другой ячейки ОП, адресуемой полем *CONST* или регистром РЗУ. Для передачи адреса в *AOP* после его чтения из ОП в поле *DST* записывается код 6 – запись из *RGR* в *AOP* или код 5 – запись в *AOP* и РЗУ.

4. Микропрограммирование алгоритмов

Микропрограммирование задач с обработкой данных в ОП и с использованием циклов и подпрограмм рассмотрим на примере нахождения минимального элемента в массиве M чисел, записанных в ОП, начиная с адреса *NADDR*. Найденный элемент записывается в ОП по адресу *MIN*.

Программа *MINEL* нахождения минимального элемента в массиве представлена в табл. 13, а входящая в нее подпрограмма сравнения *SRAV* – в табл. 14.

Таблица 13

Программа нахождения минимального элемента

Текст программы				Пояснение
MINEL:	*MOV*	*#NADDR*,	*R*1;	Начальный адрес
	MOV	*@#NADDR*,	*R*2;	Первый элемент
	MOV	*#M*-1,	*R*0;	Число циклов
LMIN:	*JSR*	*RC*,	*SRAV*;	Сравнение элементов
	SOB	*R*0,	*LMIN*;	Переход по счетчику
	MOX	*R*2,	*@#MIN*;	Запись результата
	HALT		;	Останов

Таблица 14

Подпрограмма сравнения

Текст программы				Пояснение
SRAV:	*ADD*	#2,	*R*1;	Инкремент адреса
	CMP	(*R*1),	*R*2;	Сравнение
	BGE	*LSRAV*	;	Новый элемент не меньше
	MOV	(*R*1),	*R*2;	Запись меньшего элемента
LSRAV:	*RTS*	*PC*	;	Выход из подпрограммы

Минимальный элемент первоначально получается в *R*2, а затем передается в ОП по требуемому адресу.

При интерпретации программы микропрограммой примем:

– что микропрограмма начинается с адреса 0, подпрограмма – с адреса 10;

– массив из десяти исходных чисел записан, начиная с адреса 100, результат записывается по адресу 20;

– подсчет числа циклов происходит в регистре счетчика-адреса *RACT*; поскольку число выполняемых циклов на 1 больше числа, заносимого в счетчик, в него заносится значение 8.

Основная микропрограмма приведена в табл. 15, подпрограмма *SRAV* – в табл. 16.

Таблица 15

Микропрограмма нахождения минимального элемента

Адрес МК	Операция	Поле	Значение	Функция
0	*Minel* :		1	*R*l
	MOV #100, *R*1	*SRC*	4	*DA*
		DST	1	Запись РЗУ, *AOP*
		CONST	100	
1	*MOV* (*R*l), *R*2	*B*	2	*R*2
	MOV #8, *RACT*	*R*	1	Чтение ОП
		DST	4	Запись РЗУ
		WA	0	Запись в *RACT*
		CONST	8	
2	*LMIN*:	*JFI*	4	Б/У переход
	JSR PC, *SRAV*	*CHA*	1	*CJS*
		CONST	10	Адрес П/П
3	*SOB RACT*, *LMIN*	*CHA*	9	*RPCT*
		CONST	2	
4	*MOV* #20, *AOP*	*SRC*	4	*DA*
		DST	3	Запись в *AOP*
		CONST	20	
5	*MOV R*2, @#20	*A*	2	*R*2
	HALT	*DST*	2	Запись в *RGW*
		W	1	Запись ОП
		JFI	5	*STOP*

Таблица 16

Микроподпрограмма сравнения

Адрес МК	Операция	Поле	Значение	Функция
10	*SRAV*:	*B*	1	*R*l
	ADD #2, *R*1	*SRC*	4	*DA*, *RGB*
		ALU	3	*R* + *S*
		DST	1	Запись РЗУ, *AOP*
		CONST	2	

Окончание табл. 16

Адрес МК	Операция	Поле	Значение	Функция
11	*CMP*(*R*1), *R*2 *BGE RTS PC*	*A* *R* *SRC* *ALU* *CCX* *DST* *CC* *JFI* *CHA*	2 1 2 1 1 6 5 1 *A*	*R*2 Чтение ОП *RGA*, *DB* *S* – *R* – 1 – *C*0
12	*MOV* (*R*1), *R*2 *RTS PC*	*B* *R* *DST* *JFI* *CHA*	2 1 4 4 *A*	

Отметим особенности микропрограмм. По МК 0 начальный адрес массива заносится одновременно в *R*1 и *AOP*. По МК 1 первый элемент массива читается в *R*2, а в счетчик заносится число циклов из поля *CONST*. Таким образом три команды интерпретируются двумя микрокомандами. Команда записи результата интерпретируется: по МК 4 в *AOP* заносится адрес, по МК 5 происходит запись по этому адресу. Одновременно по этой МК устанавливается признак *STOP*. По МК 10 подпрограммы к содержимому *R*1 прибавляется 2, и полученная сумма засылается в *R*1 и *AOP* как адрес следующего элемента. В последующих двух МК по этому адресу выбирается число соответственно для сравнения и для записи в *R*2, если оно меньше того, что там находится. В МК 11 предусмотрен условный, а в МК 12 – безусловный выход из подпрограммы.

5. Содержание отчета

В отчет включаются блок-схема алгоритма, программа с подробными комментариями, микропрограмма, исходные данные и результаты с указанием их размещения в памяти.

6. Задания на микропрограммирование

1. Используя команды И, ИЛИ, ИСКЛЮЧАЮЩЕЕ ИЛИ, составить программу, реализующую четыре поразрядные двоичные функции над двумя кодами, записанными в оперативной и/или регистровой памяти. Реализуемый набор выбирается из числа нетривиальных функций с номерами из множества (2, 4, 8, 9, 11, 13, 14). Например, набор {2, 0, 11, 14) включает функции $a \,\&\, \bar{b}$, $\bar{a} \,\&\, \bar{b}$, a v $\bar{b}$, $\bar{a}$ v $\bar{b}$.

2. Составить аналогичную программу, в которой одним из операндов является код, записанный в оперативной памяти, а второй образуется из первого логическим сдвигом на заданное число бит в ту или иную сторону.

3. Составить программу, сравнивающую число *M*, записанное в оперативной памяти, с некоторым числом *C*. В случаях, когда $N = C$, $N > C$, $N < C$, в один из РОНов записываются соответственно 0, 1, –1.

4. Составить программу, вычисляющую номер 1 элемента массива M[1] = *C*, где *C* – некоторая константа, с записью полученного значения в один из РОНов. Если массив ив содержит заданного значении, в РОН записывается 0.

5. Составить программу вычисления квадрата положительного числа, записанного в одном из РОНов. Возведение в квадрат выполняется с помощью операций сложения и сдвига. Предполагается, что результат не выходит за пределы разрядной сетки.

6. Определить остаток и (или) частное от деления числа на константу, равную целой степени двойки.

7. Определить остаток *R* от деления числа *A*, рассматриваемого как целое без знака, на константу *P*, равную 3, 7 или 15.

Остаток вычисляется выделением четверичных, восьмеричных или шестнадцатиричных цифр *A* и суммированием их по модулю *P*. Так, алгоритм вычисления остатка по модулю 7 можно представить следующим образом:

```
        R:= 0;
M:      B:= A & 7;                  младшая цифра
        R:= R + B;
        если R > 6, то R: = R – 7;
        A:= SRA A, 3;               сдвиг на 3 бита вправо
        если A > 0, то перейти к метке M.
```

8. Сформировать четыре массива, содержащих поразрядные двоичные функции, аргументами которых являются одноименные элементы массивов *A* и *B*. Вычисление четверки функций для одной пары аргументов оформить как подпрограмму (см. задание 1).

9. Сформировать четыре массива, содержащих поразрядные двоичные функции, аргументами которых являются элементы массива *A* и их сдвинутые значения. Вычисление четверки функций для одного элемента массива оформить как подпрограмму (см. задание 2).

10. Вычислить количество нулевых, положительных, отрицательных, неположительных и неотрицательных элементов массива *A* с использованием подпрограммы (см. задание 3).

11. Вычислить суммы положительных, отрицательных и количество ненулевых элементов массива *A* с использованием подпрограммы (см. задание 3).

12. Вычислить количество элементов массива *A*, равных некоторой константе, больших и меньших ее.

13. Вычислить количество элементов массива *A*, равных одноименным элементам массива *B*, больших и меньших этих элементов (см. задание 3).

14. Заданы массивы *A*, *B* и некоторые значения *M* и *N*. Составить программу, которая выполняет следующие вычисления:

– определяет номера H1, H2 элементов пассивов *A* и *B*, равных *M* и *N* соответственно;

– вычисляет квадраты номеров H1 и H2.

Определение номера элемента массива и возведение в квадрат оформить в виде вложенных подпрограмм (см. задания 4, 5).

15. Сформировать два массива, элементы которых равны остаткам и частным от деления одноименных элементов исходного массива *A* на целую степень двойки (см. задание 6).

16. Рассортировать исходный массив *A* на четыре массива, элементы которых при делении на 4 дают соответственно остатки 0, 1, 2 и 3 (см. задание 6).

17. Сформировать массив остатков от деления элементов исходного массива *A* на 3, 7 или 15 (см. задание 7).

18. Элементы исходного массива A дополнить остатками от деления этих элементов на 3, 7 или 15 (см. задание 7). При этом коды исходных чисел сдвигаются на соответствующее число битов влево, а остатки записываются на освободившееся место.

19. Подсчитать число элементов массива *A*, которые дают ненулевые остатки при делении на 3, 7 или 15 (см. задание 7).

7. Контрольные вопросы

1. Опишите структуру блока микропрограммного управления и функции его составляющих. Для каких целей в вашей микропрограмме используются стек, регистр счетчика-адреса, мультиплексор кода условия?

2. Какие способы формирования адреса следующей МК реализуются в БМУ?

3. Проанализируйте формирование условий перехода в зависимости от значений поля *CSH*. В каких случаях используется инверсия условия ? Когда следует использовать дополнительный регистр флажков?

4. Какие инструкции БМУ используются для организации циклов и подпрограмм? Можно ли в вашей микропрограмме организовать циклы и подпрограммы иначе, какими средствами вы это сделали? Как это повлияет на качество микропрограммы?

5. Опишите способы адресации ОП, используемые в процессоре. Какие из них имеются в вашей микропрограмме?

Лабораторная работа № 3

МИКРОПРОГРАММИРОВАНИЕ КОМАНД СМ ЭВМ

Цель работы: знакомство с принципами микропрограммной эмуляции ЭВМ с программным управлением, микропрограммирование машинных команд СМ ЭВМ.

1. Принципы микропрограммной эмуляции

Эмуляцией в общем случае называют метод приспособления одних ЭВМ для решения задач, подготовленных для других машин. Под микропрограммной эмуляцией понимается выполнение микропрограммируемым

процессором операций программно-управляемой ЭВМ (ПЭВМ). В нашем случае ПЭВМ эмулирует микропрограммируемый процессор, а тот, в свою очередь, эмулирует систему команд СМ ЭВМ.

В предыдущих работах для задач, предварительно запрограммированных на языке ассемблера, составлялись микропрограммы их решения в целом. Теперь речь идет о микропрограммировании отдельных операций ассемблера, решение же задачи в целом идет на программном, а не микропрограммном уровне.

Реализация ЭВМ с заданной системой команд в микропрограммируемых системах является часто встречающейся задачей проектирования, а микропрограммирование операций – один из наиболее важных этапов ее решения. Разработка микропрограммного обеспечении выполняется в следующей последовательности: составление и отладка микропрограмм отдельных операций с учетом наличия в них общих подпрограмм; размещение микропрограмм в микропрограммной памяти МП; кодирование преобразователя начального адреса, трансформирующего коды операций в адреса соответствующих микропрограмм.

2. Порядок выполнения работы

1. Составить список операций, входящих в программу решения задачи. Задачи берутся из заданий к работе 1.

2. По материалам п. 4 и рекомендуемой литературе ознакомиться с форматами команд и описанием операций.

3. Пользуясь рекомендациями п. 4, составить и отладить микропрограммы операций и разместить их в МРМ.

4. Составить и ввести в модель таблицу преобразования адресов.

5. Закодировать программу и ввести ее в оперативную память.

6. Ввести исходные данные и установить программный счетчик РС на начало программы.

7. Выполнить программу в режимах КОМАНДА и АВТОМАТ, зафиксировав изменения состояния модели после выполнения каждой команды.

3. Форматы команд и описание операций

Форматы команд представлены на рис. 5.

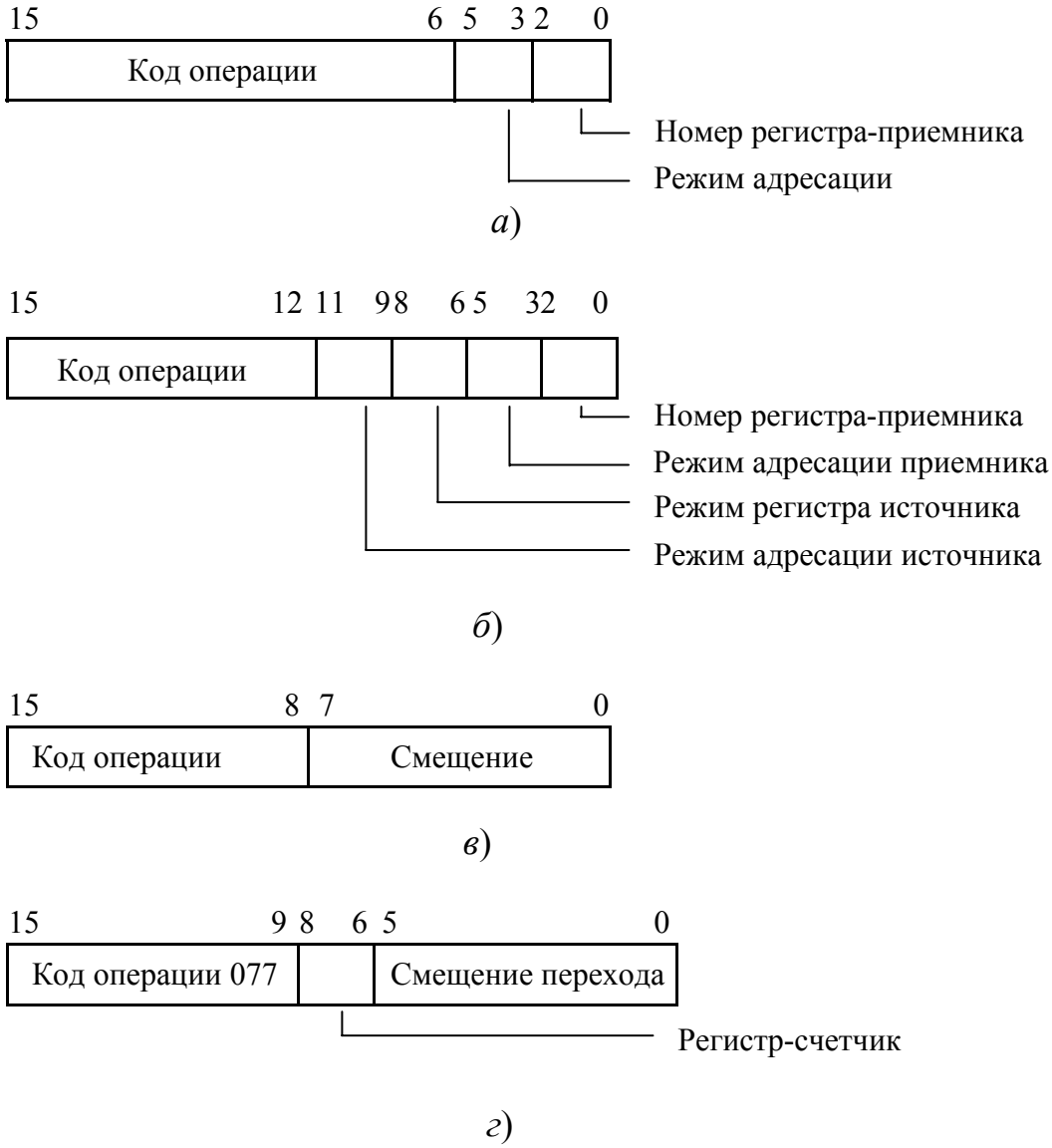

Рис. 5. Команды: а – одноадресные;
б – двухадресные; в – ветвления; г – SOB

3.1. Одноадресные команды

Одноадресная команда содержит один операнд, называемый приемником, и указание о способе его адресации. В этой работе для одноадресных команд используется только регистровая адресация (код 0).

В табл. 17 приведен список одноадресных команд и пояснения их работы.

Таблица 17

Одноадресные команды

Мнемоника	Код	Операция	Флажки			
			N	*Z*	*V*	*C*
CLR	0050	Очистка слова	0	1	0	0
COM	0051	Инверсия слова	+	+	0	1
INC	0052	Инкремент слова	+	+	+	–
DEC	0053	Декремент слова	+	+	+	–
NEG	0054	Арифметическое дополнение (смена знака)	+	+	+	Примечание 3
TST	0057	Проверка слова	+	+	0	0
ASR	0062	Арифметический сдвиг вправо	+	+	Примечание 3	Примечание 4
ASL	0063	Арифметический сдвиг влево	+	+	Примечание 3	Примечание 5
ROR	0060	Кольцевой сдвиг вправо через бит переноса *C*	+	+	Примечание 3	Примечание 4
ROL	0061	Кольцевой сдвиг влево через бит переноса *C*	+	+	Примечание 3	Примечание 5

Примечания: 1. Знак "+" – установка флажка по результату операции, знак "–" – флажок не меняет значения. 2. $C = 0$, если результат равен нулю, $C = 1$, если результат не равен нулю. 3. $V = C + N$. 4. *C* принимает значение разряда 0. 5. *C* принимает значение разряда 15.

3.2. Двухадресные команды

Команда включает в себя два операнда. Первый из них называется источником, второй – приемником. В работе для двухадресных команд используются регистровая (код 0) и непосредственная (код 2, номер ре-

гистра 7) адресация. Непосредственный операнд находится во втором слове команды. Список двухадресных команд с пояснениями приведен в табл. 18.

Таблица 18

Двухадресные команды

Мнемоника	Код	Операция	Флажки			
			H	*Z*	*V*	*C*
MOV	01	Пересылка	+	+	0	-
CMP	02	Сравнение: Второй операнд вычитается из первого, и устанавливаются флажки без записи результата	+	+	+	+
BIT	03	Операция "И" и установка флажков без записи результата	+	+	0	-
BIC	04	Очистка бит	+	+	0	-
BIS	05	Установка бит (операция ИЛИ)	+	+	0	-
ADD	06	Сложение	+	+	+	+
SUB	16	Вычитание	+	+	+	+

3.3. Команды ветвления

Код операции записывается в старшем байте. В младшем байте записано 7-битное смещение со знаком, которое определяет адрес перехода относительно *PC*. Отрицательные смещения записываются в дополнительном коде. Адрес перехода вычисляется так: знак смещения расширяется на биты с 8 по 15; результат умножается на 2 (смещение в байтах); полученное значение прибавляется к *PC*.

Смещение в словах рассчитывается по формуле $(A - PC)/2$ и усекается до одного байта. При этом нужно учитывать, что PC указывает на слово, следующее за командой ветвления, т.е. имеет значение на 2 больше, чем адрес этой команды.

Перечень команд ветвления приведен в табл. 19.

Таблица 19

Команды ветвления

Мне-моника	Код	Операция	Коды условий
BR	0004	Безусловный переход	
BNE	0010	Переход по неравенству нулю	$Z = 0$
BEQ	0014	Переход по равенству нулю	$Z = 1$
BPL	1000	Переход по плюсу	$N = 0$
BMI	1004	Переход по минусу	$N = 1$
BVC	1020	Переход по непереполнению	$V = 0$
BVS	1024	Переход по переполнению	$V = 1$
BCC	1030	Переход по отсутствию переноса	$C = 0$
BCS	1034	Переход по наличию переноса	$C = 1$
BGE	0020	Переход, если больше или равно	$N + V = 0$
BLT	0024	Переход, если меньше	$N + V = 1$
BGT	0030	Переход, если больше	Z v $(H + V) = 0$
BLE	0034	Переход, если меньше или равно	Z v $(H + V) = 1$
BHI	1010	Переход, если больше (без знака)	C v $Z = 0$
BLOS	1014	Переход, если меньше или равно (без знака)	C v $Z = 1$
BHIS	1030	Переход, если больше или равно (без знака)	$C = 0$
BLO	1034	Переход, если меньше (без знака)	$C = 1$

3.4. Дополнительные команды

В программах рассматриваемых задач используются также команды *SOB* (переход по счетчику) и *HALT* (останов). При выполнении команды *SOB* из регистра-счетчика вычитается 1. Если после этого регистр содержит число, отличное от нуля, осуществляется переход путем вычитания из *PC* удвоенного смещения. Если содержимое счетчика равно нулю, выполняется команда, следующая за *SOB*. Код операции команды *SOB* – 077 (восьмеричный).

Команда *HALT* (код 000000) вызывает останов работы процессора. *PC* указывает на следующую за *HALT* команду.

4. Рекомендации по микропрограммированию

4.1. Адресация регистров

Первые 8 регистров РЗУ имитируют регистры общего назначения СМ ЭВМ *RO*-R5, указатель стека *SP* и программный счетчик *PC*. Регистр *RF* используется для записи и обработки копии выполняемой команды. Остальные могут быть использованы по усмотрению микропрограммиста.

В программном режиме регистры могут адресоваться полями *A* и *B* микрокоманды или полями *R*1 и *R*2 команды. Для указания источника адреса используются поля *MA*, *MB* микрокоманды в соответствии с табл. 1.

4.2. Выборка команды

Процедура выборки команды одинакова для всех операций. Микропрограмма выборки команды записывается в *MPM*, начиная с адреса 0. Для выборки первой команды счетчик микрокоманд СМК устанавливается в состояние 0, а программный счетчик *PC* – на начало программы. После выполнения микропрограммы очередной команды СМК устанавливается в нуль инструкцией *JZ* (см. п. 3 (косвенная регистровая адресация) работы № 2).

Микропрограмма выборки команды содержит две МК. По первой из них адрес команды из *PC* передается через шину *DB* в *AOP*, а к *PC* прибавляется 2 для перехода к следующему слову команды. По второй МК происходит чтение *OP* и передача кода команды из регистра чтения *RGR* в регистр команд *PGK* и одновременно в регистр *RF* РЗУ. Для этого в поле *B* МК нужно записать адрес *F*, а в поле *DST* код 4 – передача из *RGR* в РЗУ. Код *F* в поле *B* обеспечивает также запись в *RGK*.

Для перехода к нужной микропрограмме в соответствии с кодом операции используется инструкция *JMAP* схемы УПМ, которая должна быть записана во второй МК выборки команды.

4.3. Команды пересылки и преобразования данных

Микропрограммы для одно- и двухадресных команд составляются с учетом способа адресации, который рассматривается в данном случае как часть кода операции. При микропрограммировании двухадресных команд с непосредственной адресацией одного из операндов необходимо помнить, что этот операнд записан во втором слове команды и адресуется *PC*. После засылки в *AOP* содержимого *PC* к последнему прибавляется 2.

Поскольку *RGR*, где находится непосредственный операнд после выборки, является источником операнда S, другой операнд, находящийся в РЗУ, выбирается по адресу *A* и на *ALU* поступает через мультиплексор *MR*. Результат операции записывается, если это нужно, по адресу *B*, который должен быть установлен равным адресу *A*.

Пусть, например, выполняется операция вида *ADD* #25,*R*5. Для ее выполнения требуется сначала выбрать непосредственный операнд 25 из *OP* в *RGR*. Адрес *A* = 5 выбирается из поля *R*2 команды. Тот же адрес устанавливается на адресном входе *B*. В поле *SRC* записывается код 2 – выборка операндов из *RGA* и с шины *DB*. В поле *ALU* задается операция сложения. Запись результата по адресу *B* = 5 выполняется по умолчанию.

В программном режиме преобразования и переходы разнесены по разным командам. Поэтому в микропрограммах преобразования необходимо предусмотреть фиксацию флажков в тех МК, где формируется результат операции, записав 1 в бит *F*.

4.4. Команды ветвления

В микропрограммах команд условного ветвления выполняется анализ флажков, зафиксированных в *RFD*, для чего записывается 1 в бит *F* поля *JFI*. Условие ветвления задается в поле *CC* в соответствии с табл. 8 и в бите *I* поля *JFI*. Для случая выполнения условия необходимо запрограммировать расширение знака младшего байта *RF*, где хранится код команды, удвоение путем сложения или сдвига влево полученного числа и прибавление его к *PC*. Для сокращения затрат времени можно анализ условия и расширение знака совместить в одной МК. Расширение знака задается кодом *E* в поле *SH*.

5. Кодирование и ввод программы

Команды программы кодируются в соответствии с их форматами и кодами операций, приведенными в п. 3. При кодировании команд ветвления и перехода по счетчику обратите особое внимание на правильное вычисление смещения.

Программа вводится в произвольную область *OP* при условии, что адреса команд не превышают *DFFE*. Порядок ввода данных в оперативную память описан в п. 4 работы 1. Программы могут корректироваться в приостановках.

6. Подготовка и ввод таблицы преобразования адресов

Таблица преобразования адресов устанавливает соответствие между кодами операций и начальными адресами соответствующих микропрограмм. Эта таблица является моделью преобразователя начальных адресов *PA*, который физически реализуется на ПЗУ или на программируемых логических матрицах ПЛМ.

В командах СМ ЭВМ длина кода операции меняется в зависимости от формата команды от 4 до 16 бит, что несколько затрудняет создание *PA* или его модели. В нашей модели принято, что код операции включает в себя биты, определяющие режим адресации операндов, и содержит 13 бит, совпадающих с битами 3 – 15 команд. При этом для ряда команд некоторые из этих бит не влияют на значение начального адреса. Иными словами, при любых значениях этих "безразличных" бит начальный адрес остается одним и тем же. Это относится, в частности, к двухадресным командам, где "безразличными" являются биты 6 – 8, определяющие номер регистра-источника, и к командам ветвления, где "безразличными" являются биты 3 – 7, к команде *SOB* с "безразличными" битами 3 – 8.

Обращение к таблице преобразования адресов осуществляется из основного меню. Таблица состоит из двух столбцов, в первый из которых записываются коды всех команд программы с попарно различными значениями пяти старших восьмеричных цифр. Шестой цифре присваивается

значение 0. Во второй столбец записываются адреса микропрограмм соответствующих команд. При этом некоторые адреса могут повторяться. От одного значения к другому переходят нажатием клавиши < ВВОД>. Для выхода в основное меню необходимо нажать клавишу <ВВОД> несколько раз, чтобы курсор переместился в последнюю позицию таблицы.

В табл. 20 приведен пример преобразования адресов для команд программы поиска наименьшего числа, описанной в работе 1. Адреса микропрограмм в правом столбце таблицы выбраны произвольно. Кодирование кодов и адресов – восьмеричное.

Таблица 20

Пример преобразования адресов

Команда	Код команды	Код операции	Адрес микропрограммы
*MOV R*1, *R*4	010104	010100	10
*MOV R*2, *R*4	010204	010200	10
*MOV R*3, *R*4	010304	010300	10
*CMP R*4, *R*2	020402	020400	14
*BLE MIN*1	003404	003400	20
*BLE MIN*2	003410	003400	20
HALT	000000	000000	24

7. Содержание отчета

В отчет входят программа решаемой задачи, микропрограммы операций, таблица преобразования адресов и результаты решения задачи в режиме "КОМАНДА".

8. Контрольные вопросы

1. В чем сущность микропрограммной эмуляции системы команд?

2. Каковы особенности микропрограммирования команд преобразования и команд ветвления?

3. Как моделируется преобразователь начального адреса? Постройте карту прошивки ПЛМ для реализации *РА* для вашего примера.

4. Сравните реализации вашей задачи на микропрограммном и программном уровнях по затратам времени и памяти. Изменится ли соотношение этих затрат при реализации не одной, а многих задач?

5. Составьте микропрограмму выполнения команды по указанию преподавателя.

Лабораторная работа № 4

МИКРОПРОГРАММИРОВАНИЕ ОПЕРАЦИЙ С РАЗЛИЧНЫМИ РЕЖИМАМИ АДРЕСАЦИИ

Цель работы: изучение способов адресации данных в СМ ЭВМ, приобретение навыков микропрограммирования команд с различными способами адресации и программирования с использованием этих команд.

1. Режимы адресации

Команды СМ ЭВМ могут содержать операнды, размещенные в регистрах общего назначения или в памяти. Операнд задается 6-разрядным полем, включающим в себя 3-разрядный номер регистра РЕГ и 3-разрядный указатель способа адресации *А*. Форматы одно- и двухадресных команд приведены на рис. 6.

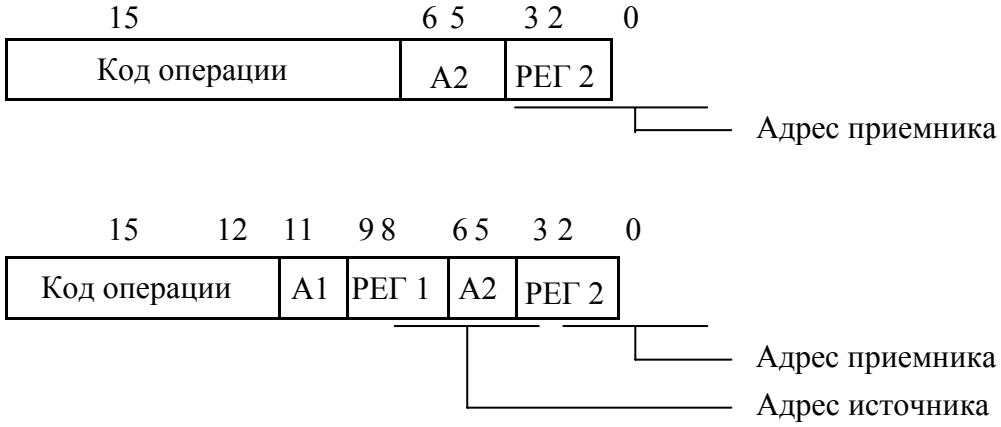

Рис. 6. Форматы одно- и двухадресных команд

1.1. Основные способы адресации

Существует семь режимов адресации, которые описываются ниже.

Регистровая (*А* = 0). Операндом является содержимое регистра. Исполнительный адрес для памяти отсутствует. Например:

*CLR R*0; очистка регистра 0.

Косвенная регистровая (*A* = 1). Исполнительный адрес определяется содержимым регистра. Например:

CLR (*R*0); очистка ячейки, адрес которой содержится в *R*0. Автоинкрементная (*A* = 2). Исполнительный адрес определяется содержимым регистра, затем это содержимое увеличивается на длину операнда. Так как в работе операции выполняются над словами, к регистру прибавляется 2. Пусть, например, в *R*0 записана величина 100. Тогда операция *CLR* (*R*0)+ очищает ячейку 100 и прибавляет 2 к содержимому регистра *R*0, который примет состояние 102.

Косвенная автоинкрементная (*A* = 3). Косвенный адрес определяется содержимым регистра, к которому затем прибавляется 2. Исполнительный адрес берется из ячейки по косвенному адресу. Если в *R*0 записана величина 100, а в ячейке 100 – значение 250, то команда *CLR* @(*R*0)+ очистит ячейку 250 и запишет 102 в *R*0.

Автодекрементная (*A* = 4). Содержимое регистра уменьшается на длину операнда, затем исполнительный адрес выбирается из регистра. Например, если в *R*0 записано число 104, команда *CLR* -(*R*0) записывает в регистр 102 и очищает ячейку 102.

Косвенная автодекрементная (*A* = 5). Содержимое регистра уменьшается на 2, затем косвенный адрес выбирается из регистра, а исполнительный – из ячейки по косвенному адресу. Так, если в *R*0 записано 104, а в ячейке 102 – значение 300, команда *CLR* @ – (*R*0) запишет в *R*0 значение 102 и очистит ячейку 300.

Индексная (*A* = 6). Содержимое регистра (индекс) и очередное слово в потоке команд (смещение) суммируются для получения исполнительного адреса. Если регистр не является программным счетчиком *PC*, его содержимое не изменяется. Адрес смещения (второе или третье слово команды) указывается *PC*. После выборки смещения к *PC* прибавляется 2. Например, если в *PC* после выборки очередной команды записано значе-

ние 62, а в *R*0 – число 100, то команда *CLR* 34(*R*0) очистит ячейку 134 и запишет 64 в *PC*.

Косвенно-индексная (*A* = 7). Содержимое регистра и очередное слово в потоке команд суммируются для получения косвенного адреса. Исполнительный адрес выбирается из ячейки по косвенному адресу. Пусть в *R*0 записано 100, в ячейке 134 – 170, а в *PC* – 62. Команда *CLR* @34(*R*0) очистит ячейку 170 и запишет 64 в *PC*.

1.2. Режимы адресации с использованием PC

Если поле РЕГ указывает программный счетчик *PC* (РЕГ = 7), то для режимов адресации 2, 3, 6, 7 получаем их особые случаи.

Непосредственная адресация (А = 2). Операндом является очередное слово в потоке команд. К *PC* прибавляется 2.

Абсолютная (*A* = 3). Очередное слово в потоке команд является исполнительным адресом. Содержимое *PC* увеличивается на 2.

Относительная (*A* = 6). Очередное слово в потоке команд является смещением относительно *PC*. После выборки смещения к *PC* прибавляется 2. Исполнительный адрес – сумма нового значения *PC* и смещения.

Косвенно-относительная (*A* = 7). Очередное слово потока команд суммируется с *PC*, после инкремента последнего, для получения косвенного адреса. Исполнительный адрес выбирается из ячейки памяти по косвенному адресу.

Обобщенные сведения по способам адресации и их представление на языке ассемблера сведены в табл. 21.

Таблица 21

Способы адресации

Код адресации	Номер регистра	Адресация	Обозначение	Описание режима
0	0 – 7	Регистровая	*Rn*	Операнд в регистре
1	0 – 7	Косвенная регистровая	*@Rn* или (*Rn*)	*EA* = *Rn*
2	0 – 7	Автоинкрементная	(*Rn*)+	*EA* = *Rn*; *Rn*: = *Rn*+2

Окончание табл. 21

Код адресации	Номер регистра	Адресация	Обозначение	Описание режима
3	0 – 7	Косвенная автоинкрементная	*@Rn*+	*IA* = *Rn*; *Rn*: = *Rn* + 2; *EA* = (*IA*)
4	0 – 7	Автодекрементная	-(*Rn*)	*Rn*: = *Rn* – 2; *EA* = *Rn*
5	0 – 7	Косвенная автодекрементная	*@-Rn*	*Rn*: = *Rn* – 2; *IA* = *Rn*; *EA* = (*IA*)
6	0 – 7	Индексная	*E*(*Rn*)	*E*: = (*PC*); *PC*: = *PC* + 2; *EA* = *E* + *Rn*
7	0 – 7	Косвенно-индексная	*@ERn*	*E*: = (*PC*); *PC*: = *PC* + 2; *IA* = *E* + *PC*; *EA* = (*IA*)
2	7	Непосредственная	#E	*E*: = (*PC*); *PC*: = *PC* + 2; *E* – операнд
3	7	Абсолютная	@#E	*EA* = (*PC*); *PC*: = *PC*+2
6	7	Относительная	*E*	*E*: = (*PC*); *PC*: = *PC* + 2; *EA* = *PC*
7	7	Косвенно-относительная	@E	*E*: = (*PC*); *PC*: = *PC* + 2; *IA* = *E* + *PC*; EA = (*IA*)

Примечание. *Rn* – регистр с номером n и одновременно его содержимое; *PC* – программный счетчик и его содержимое; (*Rn*), (*PC*) – содержимое ячейки памяти, адресуемой *Rn* или *PC*; *EA* – исполнительный адрес; *IA* – косвенный адрес; (*IA*) – содержимое ячейки с адресом *IA*.

2. Микропрограммирование формирования исполнительных адресов

Процедуру формирования адресов рассмотрим на примере косвенно-индексной адресации для одноадресной команды. Формирование *EA* можно описать в виде следующей последовательности микроопераций:

AOP: = *PC*; – адрес смещения *E*
PC: = *PC* + 2; – инкремент *PC*
ЧОп; – выборка *E* из ОП
AOP: = (РЕГ2) + *RGR*; – косвенный адрес *IA*
ЧОп; – выборка исполнительного адреса *EA*
AOP: = *RGR*; – запись *EA* в *AOP*

Здесь РЕГ2 – номер регистра-приемника, заданный в соответствующем поле команды, (РЕГ2) – содержимое этого регистра.

Микропрограмма формирования *EA* приведена в табл. 22. Аналогичным образом микропрограммируется формирование *EA* для других режимов адресации.

Таблица 22

Формирование исполнительного адреса

Содержание МК	Поле	Значение	Функция
AOP: = *PC* *PC*: = *PC* + 2	*B* *SRC* *ALU* *DBA* *CONST*	7 4 3 1 2	*PC* *DA*, *RGB* Сложение *AOP*: = *RGB* Приращение *PC*
ЧОп *AOP*: = (РЕГ2) + *RGR*	*MA* *SRC* *R* *ALU* *DST*	2 2 1 3 3	Адрес *A* из поля РЕГ2 *RGA*, *DB* Чтение ОП Сложение *AOP*: = *SHA*
ЧОп *AOP*: = *RGR*	*R* *DST*	1 7	ЧОп *AOP*: = *RGR*

Если в вашей программе встречается несколько разных команд с одним и тем же способом адресации, процедуру формирования *EA* целесообразно оформить в виде подпрограммы.

3. Порядок выполнения работы

1. Ознакомиться с режимами адресации и порядком микропрограммирования формирования исполнительных адресов.

2. Разработать программу на языке ассемблера в соответствии с заданием и закодировать ее в восьмеричном коде.

3. Разработать микропрограммы выполнения команд программы с учетом используемых способов адресации.

4. Ввести микропрограммы операций и произвести их отладку.

5. Ввести программу, числовые исходные данные и таблицу преобразования кодов операций в начальные адреса реализующих их микропрограмм (см. работу 3).

6. Отладить и выполнить программу в режимах КОМАНДА и АВТОМАТ.

4. Содержание отчета

В отчет входят программы, микропрограммы операций, таблица преобразования адресов, результаты вычислений, протокол выполнения программы в режиме КОМАНДА, протокол формирования адреса для наиболее сложного способа адресации в режиме МИКРОКОМАНДА.

5. Примеры заданий

1. Нахождение минимального (или максимального) из восьми чисел, записанных в произвольных ячейках памяти. Адреса чисел хранятся в таблице ТА. Результат записать в ОП по адресу *C*. В программе используются режимы непосредственной, абсолютной, косвенной автоинкрементной или косвенной автодекрементной адресации.

2. Выборка из восьми чисел, адреса которых заданы таблицей ТА, чисел больших или меньших некоторой константы К и запись их в массив *M*. В программе используются непосредственная, автоинкрементная или автодекрементная, косвенная автоинкрементная или косвенная автодекрементная адресации.

3. Поэлементное сравнение двух 10-элементных массивов, смещенных друг относительно друга на некоторое постоянное смещение, и подсчет количества совпадающих элементов. Возможные варианты – подсчет числа несовпадающих элементов, случаев, когда элемент первого массива больше, чем одноименный элемент второго и т.п. Используются режимы непосредственной, абсолютной, автоинкрементной и индексной адресации.

6. Контрольные вопросы

1. Опишите режимы адресации, используемые в СМ ЭВМ.

2. Составьте микропрограмму вычисления адреса для режима адресации, указанного преподавателем.

3. В чем состоят отличия между автоинкрементной и автодекрементной адресацией и как они отражаются в микропрограммах формирования адреса?

4. Составьте программы для заданий 1 и 2 п. 5 без использования косвенной адресации. Сравните эти программы с исходными по затратам памяти и времени.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Бугаков, С. С*. Проектирование цифровых систем на комплектах микропрограммируемых БИС / С. С. Бугаков. – М. : Радио и связь, 1984. – 240 с.

2. *Видгорчук, Г. В*. Основы программирования на Ассемблере СМ ЭВМ / Г. В. Видгорчук. – М. : Финансы и статистика, 1983. – 256 с.

3. Проектирование микропроцессорных систем : учеб. пособие / В. Б. Буланкин, В. Л. Григорьев ; ВПИ. – Владимир, 1986. – 92с.

4. Моделирование работы процессора на секционированных микропроцессорах : метод. указания к лабораторным работам / Владим. гос. ун-т ; сост.: В. Б. Буланкин, Т. А. Луценко. – Владимир : Изд-во Владим. гос. ун-та, 1993. – 48 с.

ОГЛАВЛЕНИЕ

ВИРТУАЛЬНАЯ ЭВМ

Методические указания к лабораторным работам

Составитель
БУЛАНКИН Валерий Борисович

Ответственный за выпуск – зав. кафедрой профессор В.Н. Ланцов

Редактор Е.В. Афанасьева
Технический редактор Н. В. Тупицына
Корректор В.В. Гурова
Компьютерная верстка С.В. Павлухиной

Подписано в печать 14.09.06.
Формат 60х84/16. Бумага для множит. техники. Гарнитура Таймс.
Печать на ризографе. Усл. печ. л. 3,25. Уч.-изд. л. 2,79. Тираж 100 экз.
Заказ
Издательство
Владимирского государственного университета.
600000, Владимир, ул. Горького, 87.