Министерство науки и высшего образования
Российской Федерации
Братский педагогический колледж
федерального государственного бюджетного
образовательного учреждения высшего
образования
«Братский государственный университет»

АРХИТЕКТУРА АППАРАТНЫХ СРЕДСТВ

**методические рекомендации
по выполнению лабораторных работ**

для студентов II курса
очной формы обучения
специальности
09.02.07 Информационные системы и программирование

Автор: Ю.Н. Войтухов

Братск, 2020

Архитектура аппаратных средств. Методические рекомендации по лабораторных работ. /Сост. Ю.Н. Войтухов.- Братск, 2020. - 41 с.

Содержат указания к выполнению лабораторных работ по дисциплине «Архитектура компьютерных систем».

Предназначены для студентов специальности 09.02.07 Информационные системы и программирование.

Печатается по решению научно-методического совета
Братского педагогического колледжа ФГБОУ ВО «БрГУ»
665709, г. Братск, ул. Макаренко 40

Содержание

Описание архитектуры учебной ЭВМ

Программная модель позволяет реализовать доступ к различным элементам ЭВМ, обеспечивая удобство и наглядность. С другой стороны, модель позволяет игнорировать те особенности работы реальной ЭВМ, которые на данном уровне рассмотрения не являются существенными.

Далее приводится описание программной модели учебной ЭВМ1, предназначенной для начальных этапов изучения архитектуры (в т. ч. на младших курсах вуза и даже в школе). Именно этим объясняется использование в модели десятичной системы счисления для кодирования команд и представления данных.

Структура ЭВМ

Моделируемая ЭВМ включает процессор, оперативную (ОЗУ) и сверхоперативную память, устройство ввода (УВв) и устройство вывода (УВыв). Процессор, в свою очередь, состоит из центрального устройства управления (УУ),

арифметического устройства (АУ) и системных регистров (CR, PC, SP и др.). Структурная схема ЭВМ показана на рис. 8.1.

В ячейках ОЗУ хранятся команды и данные. Емкость ОЗУ составляет 1000 ячеек. По сигналу MWr выполняется запись содержимого регистра данных (MDR) в ячейку памяти с адресом, указанным в регистре адреса (MAR). По сигналу MRd происходит считывание — содержимое ячейки памяти с адресом, содержащимся в MAR, передается в MDR.

Сверхоперативная память с прямой адресацией содержит десять регистров общего назначения R0—R9. Доступ к ним осуществляется (аналогично доступу к ОЗУ) через регистры RAR и RDR.

АУ осуществляет выполнение одной из арифметических операций, определяемой кодом операции (COP), над содержимым аккумулятора (Ace) и регистра операнда (DR). Результат операции всегда помещается в Ace. При завершении выполнения операции АУ вырабатывает сигналы признаков результата: Z (равен 1, если результат равен нулю); S (равен 1, если результат отрицателен); OV (равен 1, если при выполнении операции произошло переполнение разрядной сетки). В случаях, когда эти условия не выполняются, соответствующие сигналы' имеют нулевое значение.

В модели ЭВМ предусмотрены внешние устройства двух типов. Во-первых, это регистры IR и OR, которые могут обмениваться с

аккумулятором с помощью безадресных команд in (Ace := IR) и out (OR := Ace). Во-вторых, это набор моделей внешних устройств, которые могут подключаться к системе и взаимодействовать с ней в соответствии с заложенными в моделях алгоритмами. Каждое внешнее устройство имеет ряд программно-доступных регистров, может иметь собственный обозреватель (окно видимых элементов). Подробнее эти внешние устройства описаны в разд. 8.6.

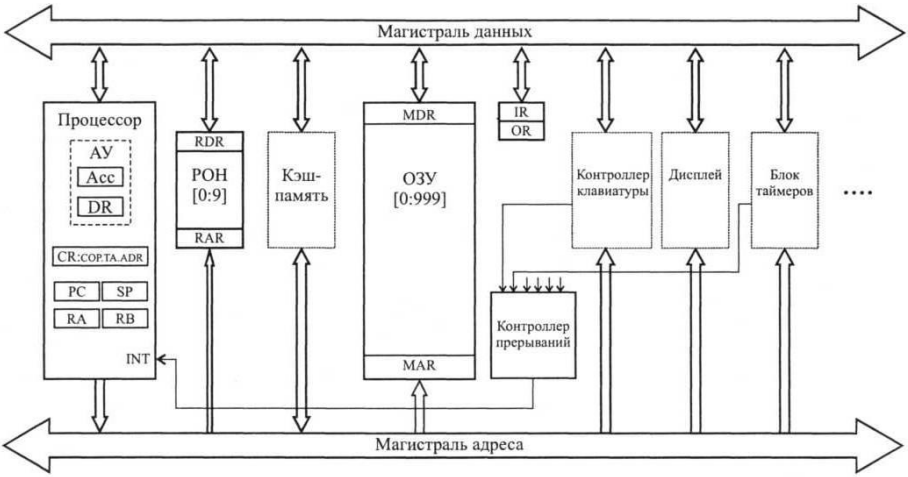
УУ осуществляет выборку команд из ОЗУ в последовательности, определяемой естественным порядком выполнения команд (т. е. в порядке возрастания адресов команд в ОЗУ) или командами передачи управления; выборку из ОЗУ операндов, задаваемых адресами команды; инициирование выполнения операции, предписанной командой; останов или переход к выполнению следующей команды.

В качестве сверхоперативной памяти в модель включены регистры общего назначения (РОН), и может подключаться модель кэш-памяти.

В состав УУ ЭВМ входят:

□ PC — счетчик адреса команды, содержащий адрес текущей команды; П CR — регистр команды, содержащий код команды;

□ RB — регистр базового адреса, содержащий базовый адрес;



□ SP — указатель стека, содержащий адрес верхушки стека;

□ RA — регистр адреса, содержащий исполнительный адрес при косвенной адресации.

Регистры Acc, DR, IR, OR, CR и все ячейки ОЗУ и РОН имеют длину 6 десятичных разрядов, регистры PC, SP, RA и RB — 3 разряда.

Представление данных в модели

Данные в ЭВМ представляются в формате, показанном на рис. 8.2. Это целые десятичные числа, изменяющиеся в диапазоне "-99 999...+99 999", содержащие знак и 5 десятичных цифр.

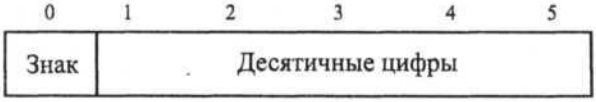


Рис. 8.2. Формат десятичных данных учебной ЭВМ

Старший разряд слова данных используется для кодирования знака: плюс (+) изображается как 0, минус (-)— как 1. Если результат арифметической операции выходит за пределы указанного диапазона, то говорят, что произошло переполнение разрядной сетки. АЛУ в этом случае вырабатывает сигнал переполнения OV = 1. Результатом операции деления является целая часть частного. Деление на ноль вызывает переполнение.

8.3. Система команд

При рассмотрении системы команд ЭВМ обычно анализируют три аспекта: форматы, способы адресации и систему операций.

Форматы команд

Большинство команд учебной ЭВМ являются одноадресными или безадресными, длиной в одно машинное слово (6 разрядов). Исключение составляют двухсловные команды с непосредственной адресацией и команда mov, являющаяся двухадресной.

В форматах команд выделяется три поля:

- два старших разряда [0:1] определяют код операции COP;
- разряд 2 может определять тип адресации (в одном случае (формат 5а) он определяет номер регистра);
- разряды [3:5] могут определять прямой или косвенный адрес памяти, номер регистра (в команде mov номера двух регистров), адрес перехода или короткий непосредственный операнд. В двухсловных командах непосредственный операнд занимает поле [6:11].

Полный список форматов команд показан на рис. 8.3, где приняты следующие обозначения:

- COP — код операции;
- ADR — адрес операнда в памяти;
- ADC — адрес перехода;
- I — непосредственный операнд;
- R, Rl, R2 — номер регистра;
- TA — тип адресации;

□ X — разряд не используется.

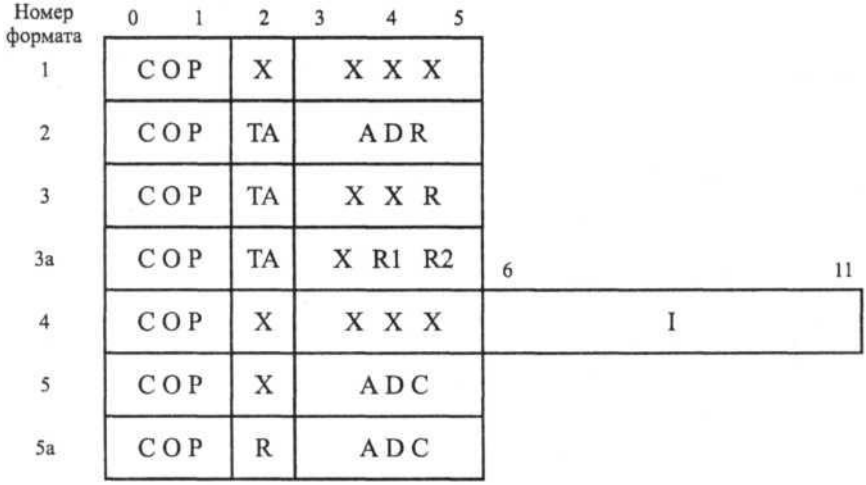


Рис. 8.3. Форматы команд учебной ЭВМ

Способы адресации

В ЭВМ принято различать пять основных способов адресации: прямая, косвенная, непосредственная, относительная, безадресная.

Каждый способ имеет разновидности. В модели учебной ЭВМ реализованы семь способов адресации, приведенные в табл. 8.1.

Таблица 8.1. Адресация в командах учебной ЭВМ

Код ТА	Тип адресации	Исполнительный адрес
0	Прямая (регистровая)	ADR (R)
1	Непосредственная	—
2	Косвенная	ОЗУ(ADR)[3:5]
3	Относительная	ADR + RB
4	Косвенно-регистровая	РОН(R)[3:5]
5	Индексная с постинкрементом	РОН(R)[3:5], R:^ R +1
6	Индексная с преддекрементом	R:= R -1, РОН(R)[3:5]

8.3.3. Система операций

Система команд учебной ЭВМ включает команды следующих классов:

П арифметико-логические и специальные: сложение, вычитание, умножение, деление;

□ пересылки и загрузки: чтение, запись, пересылка (из регистра в регистр), помещение в стек, извлечение из стека, загрузка указателя стека, загрузка базового регистра;

□ ввода/вывода: ввод, вывод;

□ передачи управления: безусловный и шесть условных переходов, вызов подпрограммы, возврат из подпрограммы, цикл, программное прерывание, возврат из прерывания;

□ системные: пустая операция, разрешить прерывание, запретить прерывание, стон.

Список команд учебной ЭВМ приведен в табл. 8.4 и 8.6.

Состояния и режимы работы ЭВМ

Ядром УУ ЭВМ является управляющий автомат (УА), вырабатывающий сигналы управления, которые инициируют работу АЛУ, РОН, ОЗУ и УВВ, передачу информации между регистрами устройств ЭВМ и действия над содержимым регистров УУ.

ЭВМ может находиться в одном из двух состояний: Останов и Работа.

В состояние Работа ЭВМ переходит по действию команд Пуск или Шаг. Команда Пуск запускает выполнение программы, представляющую собой последовательность команд, записанных в ОЗУ, в автоматическом режиме до

команды hlt или точки останова. Программа выполняется по командам, начиная с ячейки ОЗУ, на которую указывает PC, причем изменение состояний объектов модели отображается в окнах обозревателей.

В состояние Останов ЭВМ переходит по действию команды Стоп или автоматически в зависимости от установленного режима работы.

Команда Шаг, в зависимости от установленного режима работы, запускает выполнение одной команды или одной микрокоманды (если установлен Режим микрокоманд), после чего переходит в состояние Останов.

В состоянии Останов допускается просмотр и модификация объектов модели: регистров процессора и РОН, ячеек ОЗУ, устройств ввода/вывода. В процессе модификации ячеек ОЗУ и РОН можно вводить данные для программы, в ячейки ОЗУ — программу в кодах. Кроме того, в режиме Останов можно менять параметры модели и режимы ее работы, вводить и/или редактировать программу в мнемокодах, ассемблировать мнемокоды, выполнять стандартные операции с файлами.

Вспомогательные таблицы

В данном разделе представлены вспомогательные таблицы (табл. 8.4—8.8) для работы с моделью учебной ЭВМ.

Таблица 8.4. Таблица команд учебной ЭВМ

1чл>\0т	0	1	2	3	4
0	NOP	JMP		MOV	
1	IN	JZ	RD	RD	RDI
2	OUT	JNZ	WR	WR	
3	IRET	JS	ADD	ADD	ADI
4	WRR B	JNS	SUB	SUB	SBI
5	WRSP	JO	MUL	MUL	MULI
6	PUSH	JNO	DIV	DIV	DIVI
7	POP	JRNZ		IN	
8	RET	INT	El	OUT	
9	HLT	CALL	DI		

Таблица 8.5. Типы адресации, их коды и обозначение

□ DD— данные, формируемые командой в качестве (второго) операнда: прямо или косвенно адресуемая ячейка памяти или трехразрядный непосредственный операнд;

□ R* — содержимое регистра или косвенно адресуемая через регистр ячейка памяти;

□ ADR* — два младших разряда ADR поля регистра CR;

□ V — адрес памяти, соответствующий вектору прерывания;

□ M(*) — ячейка памяти, прямо или косвенно адресуемая в команде;

□ I — пятиразрядный непосредственный операнд со знаком.

Таблица 8.6. Система команд учебной ЭВМ

КОП	Мнемокод	Название	Действие
12	JNZ	Переход, если не 0	if Acc Ф 0 then PC <- CR[ADR]
13	JS	Переход, если отрицательно	if Acc < 0 then PC <- CR[ADR]
14	JNS	Переход, если положительно	if Acc > 0 then PC <- CR[ADR]
15	JO	Переход, если переполнение	if |Acc| > 99999 then PC <- CR[ADR]
16	JNO	Переход, если нет переполнения	if |Acc| < 99999 then PC <- CR[ADR]
17	JRNZ	Цикл	DEC(R); if R > 0 then PC <- CR[ADR]
18	INT	Программное прерывание	DEC(SP); M(SP) <- FLAGS.PC; PC <r- M(V)
19	CALL	Вызов подпрограммы	DEC(SP); M(SP) <- PC; PC <- CR(ADR)
20	Нет		
21	RD	Чтение	Acc <- DD
22	WR	Запись	M(*) <- Acc
23	ADD	Сложение	Acc <- Acc + DD
24	SUB	Вычитание	Acc <- Acc - DD
25	MUL	Умножение	Acc <- Acc x DD
26	DIV	Деление	Acc <-Acc/DD
27	Нет		
28	El	Разрешить прерывание	IF <- 1
29	DI	Запретить прерывание	IF <— 0
30	MOV	Пересылка	Rl<-R2
31	RD	Чтение	Acc <-R*
32	WR	Запись	R*<-Acc
33	ADD	Сложение	Acc <- Acc + R*

34	SUB	Вычитание	Acc <- Acc - R*
35	MUL	Умножение	Acc <- Acc x R*
36	DIV	Деление	Acc <-Acc/R*
37	IN	Ввод	Acc <- By(CR[ADR*])
КОП	Мнемокод	Название	Действие
38	OUT	Вывод	BY(CR[ADR*]) <- Acc
39	Нет		
40	Нет		
41	RDI	Чтение	Acc <-I
42	Нет		
43	ADI	Сложение	Acc <- Acc +1
44	SBI	Вычитание	Acc <- Acc -1
45	MULI	Умножение	Acc <- Acc x I
46	DIVI	Деление	Acc <-Acc/1

Лабораторные работы

Цикл лабораторных работ рассчитан на выполнение студентами в рамках курса "Архитектура ЭВМ" и других, подобных по содержанию.

Цикл включает работы различного уровня. Лабораторные работы № 1—4 ориентированны на первичное знакомство с архитектурой процессора, системой команд, способами адресации и основными приемами программирования на машинно-ориентированном языке. Лабораторная работа № 5 иллюстрирует реализацию командного цикла процессора на уровне микроопераций. Лабораторная работа № 6 посвящена способам организации связи процессора с внешними устройствами, а в лабораторных работах № 7 и 8 рассматривается организация кэш-памяти и эффективность различных алгоритмов замещения.

Все работы выполняются на программной модели учебной ЭВМ и взаимодействующих с ней в программных моделях ВУ и кэш-памяти, описанных в главе 8.

Описание работы включает постановку задачи, пример

выполнения, набор вариантов индивидуальных заданий, порядок выполнения работы, требования к содержанию отчета и контрольные вопросы.

Лабораторная работа № 1. Архитектура ЭВМ и система команд

1.1. Общие положения

Для решения с помощью ЭВМ некоторой задачи должна быть разработана программа. Программа на языке ЭВМ представляет собой последовательность команд. Код каждой команды определяет выполняемую операцию, тип

адресации и адрес. Выполнение программы, записанной в памяти ЭВМ, осуществляется последовательно по командам в порядке возрастания адресов команд или в порядке, определяемом командами передачи управления.

Для того чтобы получить результат выполнения программы, пользователь должен:

- □ ввести программу в память ЭВМ;
- □ определить, если это необходимо, содержимое ячеек ОЗУ и РОН, содержащих исходные данные, а также регистров IR и BR;
- □ установить в PC стартовый адрес программы;
- □ перевести модель в режим Работа.

Каждое из этих действий выполняется посредством интерфейса модели, описанного в главе 8. Ввод программы может осуществляться как в машинных кодах непосредственно в память модели, так и в мнемокодах в окно Текст программы с последующим ассемблированием.

Цель настоящей лабораторной работы — знакомство с интерфейсом модели ЭВМ, методами ввода и отладки программы, действиями основных классов команд и способов адресации. Для этого необходимо ввести в память ЭВМ и выполнить в режиме Шаг некоторую последовательность команд (определенную вариантом задания) и зафиксировать все изменения на уровне программно-доступных объектов ЭВМ, происходящие при выполнении этих команд.

Команды в память учебной ЭВМ вводятся в виде шестиразрядных десятичных чисел (см. форматы команд на рис. 8.3, коды команд и способов адресации в табл. 8.2—8.4).

В настоящей лабораторной работе будем программировать ЭВМ в машинных кодах.

1.2. Пример 1

Дана последовательность мнемокодов, которую необходимо преобразовать в машинные коды, занести в ОЗУ ЭВМ, выполнить в режиме Шаг и зафиксировать изменение состояний программно-доступных объектов ЭВМ (табл. 9.1).

Таблица 9.1. Команды и коды

Последовательность	Значения				
Команды	RD#20	WR30	ADD #5	WR@3O	JNZ 002
Коды	21 1020	22 0 030	23 1 005	22 2 030	12 0 002

Введем полученные коды последовательно в ячейки ОЗУ, начиная с адреса 000. Выполняя команды в режиме Шаг, будем фиксировать изменения программно-доступных объектов (в данном случае это Асе, РС и ячейки ОЗУ 020 и 030) в табл. 9.2.

Таблица 9.2. Содержимое регистров

PC	Acc	M(30)	M(20)	PC	Acc	M(30)	M(20)
000	000000	000000	000000	004			000025
001	000020			002			
002		000020		003	000030		
003	000025			004			000030

1.3. Задание 1

1. Ознакомиться с архитектурой ЭВМ (см. часть I).
2. Записать в ОЗУ "программу", состоящую из пяти команд— варианты задания выбрать из табл. 9.3. Команды разместить в последовательных ячейках памяти.
3. При необходимости установить начальное значение в устройство ввода IR.
4. Определить те программно-доступные объекты ЭВМ, которые будут изменяться при выполнении этих команд.
5. Выполнить в режиме Шаг введенную последовательность команд, фиксируя изменения значений объектов, определенных в п. 4, в таблице (см. форму табл. 9.2).

6. Если в программе образуется цикл, необходимо просмотреть не более двух повторений каждой команды, входящей в тело цикла.

Таблица 9.3. Варианты задания I

№	IR	Команда 1	Команда 2	Команда 3	Команда 4	Команда 5
1	000007	IN	MUL #2	WR10	WR 010	JNS 001
2	X	RD #17	SUB #9	WR16	WR 016	JNS 001
3	100029	IN	ADD #16	WR8	WR08	JS 001
4	X	RD #2	MUL #6	WR 11	WR 011	JNZ 00
5	000016	IN	WR8	DIV #4	WR @8	JMP 002
6	X	RD #4	WR 11	RD @11	ADD #330	JS 000
№	IR	Команда 1	Команда 2	Команда 3	Команда 4	Команда 5
7	000000	IN	WR9	RD @9	SUB#1	JS 001
8	X	RD 4	SUB #8	WR8	WR @8	JNZ 001
9	100005	IN	ADD #12	WR 10	WR 010	JS 004
10	X	RD 4	ADD #15	WR 13	WR 013	JMP 001
11	000315	IN	SUB #308	WR11	WR 011	JMP 001
12	X	RD #988	ADD #19	WR9	WR 09	JNZ 001
13	000017	IN	WR11	ADD 11	WR 011	JMP 002
14	X	RD #5	MUL #9	WR10	WR 010	JNZ 001

1.4. Содержание отчета

1. Формулировка варианта задания.
2. Машинные коды команд, соответствующих варианту задания.
3. Результаты выполнения последовательности команд в форме табл. 9.2.

1.5. Контрольные вопросы

1. Из каких основных частей состоит ЭВМ и какие из них представлены в модели?
2. Что такое система команд ЭВМ?
3. Какие классы команд представлены в модели?
4. Какие действия выполняют команды передачи управления?

5. Какие способы адресации использованы в модели ЭВМ? В чем отличие между ними?

6. Какие ограничения накладываются на способ представления данных в модели ЭВМ?

7. Какие режимы работы предусмотрены в модели и в чем отличие между ними?

8. Как записать программу в машинных кодах в память модели ЭВМ?

9. Как просмотреть содержимое регистров процессора и изменить содержимое некоторых регистров?

10. Как просмотреть и, при необходимости, отредактировать содержимое ячейки памяти?

11. Как запустить выполнение программы в режиме приостановки работы после выполнения каждой команды?

12. Какие способы адресации операндов применяются в командах ЭВМ?

13. Какие команды относятся к классу передачи управления?

Лабораторная работа № 2. Программирование разветвляющегося процесса

Для реализации алгоритмов, пути в которых зависят от исходных данных, используют команды условной передачи управления.

2.1. Пример 2

В качестве примера (несколько упрощенного по сравнению с заданиями лабораторной работы № 2) рассмотрим программу вычисления функции

$$y=\begin{cases}(x-11)^2-125, & \text{при } x\geq 16,\\ \dfrac{x^2+72x-6400}{-168}, & \text{при } x<16,\end{cases}$$

причем x вводится с устройства ввода IR, результат y выводится на OR. Граф-схема алгоритма решения задачи показана на рис. 9.1.

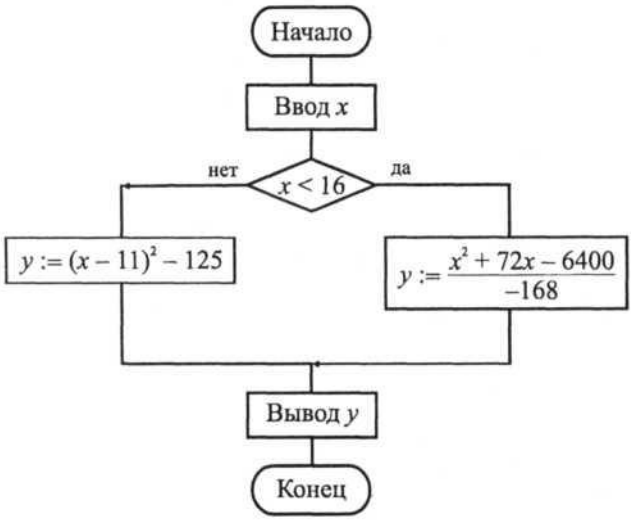


Рис. 9.1. Граф-схема алгоритма

В данной лабораторной работе используются двухсловные команды с непосредственной адресацией, позволяющие оперировать отрицательными числами и числами по модулю, превышающие 999, в качестве непосредственного операнда.

Оценив размер программы примерно в 20—25 команд, отведем для области данных ячейки ОЗУ, начиная с адреса 030. Составленная программа с комментариями представлена в виде табл. 9.4.

Таблица 9.4. Пример программы

Адрес	Кома» Мнемокод	ада Код	Примечание
000	IN	01 0 000	Ввод x
001	WR 30	22 0 030	Размещение x в ОЗУ(ОЗО)
002	SUB #16	24 1 016	Сравнение с границей — (x -16)
003	JS 010	130010	Переход по отрицательной разности
004	RD 30	21 0 030	Вычисления по первой формуле
005	SUB #11	24 1 011	
006	WR 31	22 0 031	
007	MUL 31	25 0 031	

008	SUB #125	24 1 125	
009	JMP 020	10 0 020	Переход на вывод результата
010	RD 30	21 0 030	Вычисления по второй формуле
011	MUL 30	25 0 030	
012	WR 31	22 0 031	
013	RD 30	21 0 030	
014	MUL #72	25 1 072	
015	ADD 31	23 0 031	
016	ADI 106400	43 0 000	
017		106400	
018	DIVI 100168	46 0 000	
019		100168	
Адрес	Кома Мнемокод	ада Код	Примечание
020	OUT	02 0 000	Вывод результата
021	HLT	09 0 000	Стоп

2.2. Задание 2

1. Разработать программу вычисления и вывода значения функции:

$$y = \begin{cases} F_i(x), & \text{при } x \geq a, \\ F_j(x), & \text{при } x < a, \end{cases}$$

для вводимого из IR значения аргумента x. Функции и допустимые пределы изменения аргумента приведены в табл. 9.5, варианты заданий — в табл. 9.6.

2. Исходя из допустимых пределов изменения аргумента функций (табл. 9.5) и значения параметра a для своего варианта задания (табл. 9.6) выделить на числовой оси Ox области, в которых функция y вычисляется по представленной в п. 1 формуле, и недопустимые значения аргумента. На недопустимых значениях аргумента программа должна выдавать на OR максимальное отрицательное число: 199 999.

3. Ввести текст программы в окно Текст программы, при этом возможен набор и редактирование текста непосредственно в окне Текст программы или загрузка текста из файла, подготовленного в другом редакторе.

4. Ассемблировать текст программы, при необходимости исправить синтаксические ошибки.

5. Отладить программу. Для этого:

а) записать в IR значение аргумента x> a (в области допустимых значений);

б) записать в PC стартовый адрес программы;

в) проверить правильность выполнения программы (т. е. правильность результата и адреса останова) в автоматическом режиме. В случае наличия ошибки выполнить пп. 5, г и 5, д; иначе перейти к п. 5, е;

г) записать в PC стартовый адрес программы;

д) наблюдая выполнение программы в режиме Шаг, найти команду, являющуюся причиной ошибки; исправить ее; выполнить пп. 5, а — 5, в;

е) записать в IR значение аргумента x < a (в области допустимых значений); выполнить пп. 5, б и 5, в;

ж) записать в IR недопустимое значение аргумента x и выполнить пп. 5, б

и 5, в.

6. Для выбранного допустимого значения аргумента x наблюдать выполнение отлаженной программы в режиме Шаг и записать в форме табл. 9.2 содержимое регистров ЭВМ перед выполнением каждой команды.

Таблица 9.5. Функции

0щг8гнгнгнгн8гн8нгнгн8ш9ш9ш9шгшгггггшгш

k	$F_k(x)$	k	$F_k(x)$
1	$\frac{x+17}{1-x}$; $2 \le x \le 12$	5	$\frac{(x+2)^2}{15}$; $50 \le x \le 75$
2	$\frac{(x+3)^2}{x}$; $1 \le x \le 50$	6	$\frac{2x^2+7}{x}$; $1 \le x \le 30$
3	$\frac{1000}{x+10}$; $-50 \le x \le -15$	7	$\frac{x^2+2x}{10}$; $-50 \le x \le 50$
4	$(x+3)^3$; $-20 \le x \le 20$	8	$\frac{8100}{x^2}$; $1 \le x \le 90$

Таблица 9.6. Варианты задания 2

Номер варианта	i	j	a	Номер варианта	i	j	a
1	2	1	12	8	8	6	30
2	4	3	-20	9	2	6	25
3	8	4	15	10	5	7	50
4	6	1	12	11	2	4	18
5	5	2	50	12	8	1	12
6	7	3	15	13	7	6	25
7	6	2	11	14	1	4	5

2.3. Содержание отчета

Отчет о лабораторной работе должен содержать следующие разделы:

1. Формулировка варианта задания.
2. Граф-схема алгоритма решения задачи.
3. Размещение данных в ОЗУ.
4. Программа в форме табл. 9.4.
5. Последовательность состояний регистров ЭВМ при выполнении программы в режиме Шаг для одного значения аргумента.
6. Результаты выполнения программы для нескольких значений аргумента, выбранных самостоятельно.

2.4. Контрольные вопросы

1. Как работает механизм косвенной адресации?
2. Какая ячейка будет адресована в команде с косвенной адресацией через ячейку 043, если содержимое этой ячейки равно 102 347?
3. Как работают команды передачи управления?
4. Что входит в понятие "отладка программы"?
5. Какие способы отладки программы можно реализовать в модели?

Лабораторная работа № 3. Программирование цикла с переадресацией

При решении задач, связанных с обработкой массивов, возникает необходимость изменения исполнительного адреса при повторном выполнении некоторых команд. Эта задача может быть решена путем использования косвенной адресации.

3.1. Пример 3

Разработать программу вычисления суммы элементов массива чисел Сх, С2,..., Сп . Исходными данными в этой задаче являются: п — количество суммируемых чисел и С], С2,..., Сп — массив суммируемых чисел. Заметим, что должно выполняться условие п > 1, т. к. алгоритм предусматривает, по крайней мере, одно суммирование. Кроме того, предполагается, что суммируемые числа записаны в ОЗУ подряд, т. е. в ячейки памяти с последовательными адресами. Результатом является сумма S.

Составим программу для вычисления суммы со следующими конкретными параметрами: число элементов массива— 10, элементы массива расположены в ячейках ОЗУ по адресам 040, 041, 042,..., 049. Используемые для решения задачи промежуточные переменные имеют следующий смысл: At — адрес числа Cj, / е {l, 2,..., 10}; ОЗУ(Ai) — число по адресу Ai9 S — текущая сумма; к — счетчик цикла, определяющий число повторений тела цикла.

Распределение памяти таково. Программу разместим в ячейках ОЗУ, начиная с адреса 000, примерная оценка объема программы — 20 команд; промежуточные переменные: At — в ячейке ОЗУ с адресом 030, к — по адресу 031, S — по адресу 032. ГСА программы показана на рис.

9.2, текст программы с комментариями приведен в табл. 9.7.

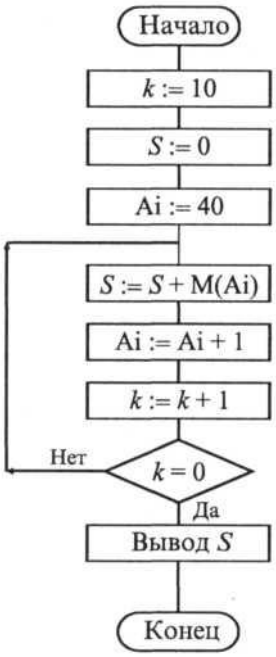


Рис. 9.2. Граф-схема алгоритма для примера 3

Таблица 9.7. Текст программы примера 3

Адрес	Команда	Примечание
000	RD #40	Загрузка начального адреса массива 040
001	WR 30	в ячейку 030
Адрес	Команда	Примечание
002	RD #10	Загрузка параметра цикла к = 10 в ячейку 031
003	WR 31	
004	RD #0	Загрузка начального значения суммы 5 = 0
005	WR 32	в ячейку 032
006	Ml: RD 32	Добавление
007	ADD @30	к текущей сумме

008	WR 32	очередного элемента массива
009	RD30	Модификация текущего
010	ADD #1	адреса массива
01l	WR 30	(переход к следующему адресу)
012	RD 31	Уменьшение счетчика
013	SUB #1	(параметра цикла)
014	WR 31	на 1
015	JNZ Ml	Проверка параметра цикла и переход при к * 0
016	RD 32	Вывод
017	OUT	результата
018	HLT	Стоп

3.2. Задание 3

1. Написать программу определения заданной характеристики последовательности чисел Cj, С2,..., Сп . Варианты заданий приведены в табл. 9.8.

2. Записать программу в мнемокодах, введя ее в поле окна Текст программы.

3. Сохранить набранную программу в виде текстового файла и произвести ассемблирование мнемокодов.

4. Загрузить в ОЗУ необходимые константы и исходные данные.

5. Отладить программу.

Таблица 9.8. Варианты задания 3

Номер варианта	Характеристика последовательности чисел С\, С2 ,..., Сп
1	Количество четных чисел
2	Номер минимального числа
3	Произведение всех чисел
4	Номер первого отрицательного числа
5	Количество чисел, равных С\
6	Количество отрицательных чисел

7	Максимальное отрицательное число
8	Номер первого положительного числа
9	Минимальное положительное число
10	Номер максимального числа
11	Количество нечетных чисел
12	Количество чисел, меньших C\
13	Разность сумм четных и нечетных элементов массивов
14	Отношение сумм четных и нечетных элементов массивов

3.3. Содержание отчета

1. Формулировка варианта задания.
2. Граф-схема алгоритма решения задачи.
3. Распределение памяти (размещение в ОЗУ переменных, программы и необходимых констант).
4. Программа.
5. Значения исходных данных и результата выполнения программы.

3.4. Контрольные вопросы

1. Как организовать цикл в программе?
2. Что такое параметр цикла?
3. Как поведет себя программа, приведенная в табл. 9.7, если в ней будет отсутствовать команда wr 31 по адресу 014?
4. Как поведет себя программа, приведенная в табл. 9.7, если метка mi будет поставлена по адресу 005? 007?

Лабораторная работа № 4. Подпрограммы и стек

В программировании часто встречаются ситуации, когда одинаковые действия необходимо выполнять многократно в разных частях программы (например, вычисление функции sinx). При этом с целью экономии памяти не следует многократно повторять одну и ту же последовательность команд — достаточно один раз написать так называемую подпрограмму (в терминах языков высокого уровня —

процедуру) и обеспечить правильный вызов этой подпрограммы и возврат в точку вызова по завершению подпрограммы.

Для вызова подпрограммы необходимо указать ее начальный адрес в памяти и передать (если необходимо) параметры — те исходные данные, с которыми будут выполняться предусмотренные в подпрограмме действия. Адрес подпрограммы указывается в команде вызова call, а параметры могут передаваться через определенные ячейки памяти, регистры или стек.

Возврат в точку вызова обеспечивается сохранением адреса текущей команды (содержимого регистра PC) при вызове и использованием в конце подпрограммы команды возврата ret, которая возвращает сохраненное значение адреса возврата в PC.

Для реализации механизма вложенных подпрограмм (возможность вызова подпрограммы из другой подпрограммы и т. д.) адреса возврата целесообразно сохранять в стеке. Стек ("магазин")— особым образом организованная безадресная память, доступ к которой осуществляется через единственную ячейку, называемую верхушкой стека. При записи слово помещается в верхушку стека, предварительно все находящиеся в нем слова смещаются вниз на одну позицию; при чтении извлекается содержимое верхушки стека (оно при этом из стека исчезает), а все оставшиеся слова смещаются вверх на одну позицию. Такой механизм напоминает действие магазина стрелкового оружия (отсюда и второе название). В программировании называют такую дисциплину обслуживания LIFO (Last In First Out, последним пришел — первым вышел) в отличие от дисциплины типа очередь — FIFO (First In First Out, первым пришел — первым вышел).

В обычных ОЗУ нет возможности перемещать слова между ячейками, поэтому при организации стека перемещается не массив слов относительно непод-

вижной верхушки, а верхушка относительно неподвижного массива. Под стек отводится некоторая область ОЗУ, причем адрес верхушки хранится в специальном регистре процессора — указателе стека SP.

В стек можно поместить содержимое регистра общего назначения по команде push или извлечь содержимое верхушки в регистр общего назначения по команде pop. Кроме того, по команде вызова подпрограммы call значение программного счетчика PC (адрес следующей команды) помещается в верхушку стека, а по команде ret содержимое верхушки стека извлекается в PC. При каждом обращении в стек указатель SP автоматически модифицируется.

В большинстве ЭВМ стек "растет" в сторону меньших адресов, поэтому перед каждой записью содержимое SP уменьшается на 1, а после каждого извлечения содержимое SP увеличивается на 1. Таким образом, SP всегда указывает на верхушку стека.

Цель настоящей лабораторной работы — изучение организации программ с использованием подпрограмм. Кроме того, в процессе организации циклов мы будем использовать новые возможности системы команд модели ЭВМ, которые позволяют работать с новым классом памяти — сверхоперативной (регистры общего назначения — РОН). В реальных ЭВМ доступ в РОН занимает значительно меньшее время, чем в ОЗУ; кроме того, команды обращения с регистрами короче команд обращения к памяти. Поэтому в РОН размещаются наиболее часто используемые в программе данные, промежуточные результаты, счетчики циклов, косвенные адреса и т. п.

В системе команд учебной ЭВМ для работы с РОН используются специальные команды, мнемоники которых совпадают с мнемониками соответствующих команд для работы с ОЗУ, но в адресной части содержат символы регистров RO—R9.

Кроме обычных способов адресации (прямой и косвенной) в регистровых командах используются два новых — постинкрементная и преддекрементная (см. табл. 8.5). Кроме того, к регистровым относится команда организации цикла JRNZ R,M. По этой команде содержимое указанного в команде регистра уменьшается на 1, и если в результате вычитания содержимого регистра не равно 0, то управление передается на метку м. Эту команду следует ставить в конце тела цикла, метку м— в первой команде тела цикла, а в регистр R помещать число повторений цикла.

4.1. Пример 4

Даны три массива чисел. Требуется вычислить среднее арифметическое их максимальных элементов. Каждый массив задается двумя параметрами: адресом первого элемента и длиной.

Очевидно, в программе трижды необходимо выполнить поиск максимального элемента массива, поэтому следует написать соответствующую подпрограмму.

Параметры в подпрограмму будем передавать через регистры: R1 — начальный адрес массива, R2 — длина массива.

Рассмотрим конкретную реализацию этой задачи. Пусть первый массив начинается с адреса 085 и имеет длину 14 элементов, второй— 100 и 4, третий— ПО и 9. Программа будет состоять из основной части и

подпрограммы. Основная программа задает параметры подпрограмме, вызывает ее и сохраняет результаты работы подпрограммы в рабочих ячейках. Затем осуществляет вычисление среднего арифметического и выводит результат на устройство вывода. В качестве рабочих ячеек используются регистры общего назначения R6 и R7 — для хранения максимальных элементов массивов. Подпрограмма получает параметры через регистры R1 (начальный адрес массива) и R2 (длина массива). Эти регистры используются подпрограммой в качестве регистра текущего адреса и счетчика цикла соответственно. Кроме того, R3 используется для хранения текущего максимума, а R4 — для временного хранения текущего элемента. Подпрограмма возвращает результат через аккумулятор. В табл. 9.9 приведен текст основной программы и подпрограммы. Обратите внимание, цикл в подпрограмме организован с помощью команды jrnz, а модификация текущего адреса— средствами постинкрементной адресации.

Таблица 9.9. Программа примера 4

Команда	Примечания
Основная программа	
RD #85	Загрузка
WR R1	параметров
RD #14	первого
WR R2	массива
CALL M	Вызов подпрограммы
WR R6	Сохранение результата
RD #100	Загрузка
WR R1	параметров
RD #4	второго
WR R2	массива
Команда	Примечания
CALL M	Вызов подпрограммы
WR R7	Сохранение результата
RD #110	Загрузка
WR Rl	параметров

RD #9	третьего
WR R2	массива
CALL M	Вызов подпрограммы
ADD R7	Вычисление
ADD R6	среднего
DIV #3	арифметического
OUT	Вывод результата
Подпрограмма MAX	
HLT	Стоп
M: RD @R1	Загрузка
WR R3	первого элемента в R3
L2: RD 0R1+	Чтение элемента и модификация адреса
WR R4	Сравнение
SUB R3	и замена,
JS LI	если R3 < R4
MOV R3,R4	
LI: JRNZ R2,L2	Цикл
RD R3	Чтение результата в Асе
RET	Возврат

4.2. Задание 4

Составить и отладить программу учебной ЭВМ для решения следующей задачи. Три массива в памяти заданы начальными адресами и длинами. Вычислить и вывести на устройство вывода среднее арифметическое параметров этих массивов. Параметры определяются заданием к предыдущей лабораторной работе (см. табл. 9.8), причем соответствие между номерами вариантов заданий 3 и 4 устанавливается по табл. 9.10.

Таблица 9.10. Соответствие между номерами заданий

Номер варианта задания 4	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Номер строки в табл. 9.9	5	7	13	11	9	12	1	10	14	3	6	8	2	4

4.3. Содержание отчета

1. Формулировка варианта задания.
2. Граф-схема алгоритма основной программы.
3. Граф-схема алгоритма подпрограммы.
4. Распределение памяти (размещение в ОЗУ переменных, программы и необходимых констант).
5. Тексты программы и подпрограммы.
6. Значения исходных данных и результата выполнения программы.

4.4. Контрольные вопросы

1. Как работает команда mov R3, R7?
2. Какие действия выполняет процессор при реализации команды call?
3. Как поведет себя программа примера 4, если в ней вместо команд call m использовать команды jmp m?
4. После начальной установки процессора (сигнал Сброс) указатель стека SP устанавливается в 000. По какому адресу будет производиться запись в стек первый раз, если не загружать SP командой wrsp?
5. Как, используя механизмы постинкрементной и преддекрементной адресации, организовать дополнительный стек в произвольной области памяти, не связанный с SP?

Лабораторная работа № 5. Командный цикл процессора

Реализация программы в ЭВМ сводится к последовательному выполнению команд. Каждая команда, в свою очередь, выполняется как последовательность микрокоманд, реализующих элементарные действия над операционными элементами процессора.

В программной модели учебной ЭВМ предусмотрен Режим микрокоманд, в котором действие командного цикла реализуется и отображается на уровне микрокоманд. Список микрокоманд текущей команды выводится в специальном окне Микрокомандный уровень (см. рис. 8.8).

5.1. Задание 5.1

Выполнить снова последовательность команд по варианту задания 1 (см. табл. 9.3), но в режиме Шаг. Зарегистрировать изменения состояния процессора и памяти в форме табл. 9.11, в которой приведены состояния ЭВМ при выполнении примера 1 (фрагмент).

5.2. Задание 5.2

Записать последовательность микрокоманд для следующих команд модели учебной ЭВМ:

□ ADD R3
□ ADD @R3
□ ADD @R3+
□ ADD -@R3 П JRNZ R3,M
□ MOV R4,R2
□ JMP M
□ CALL M
□ RET: PUSH R3
□ POP R5

5.3. Контрольные вопросы

1. Какие микрокоманды связаны с изменением состояния аккумулятора?
2. Какие действия выполняются в модели по микрокоманде MRd? RWr?
3. Попробуйте составить микропрограмму (последовательность микро команд, реализующих команду) для несуществующей команды "умноже ние модулей чисел".
4. Что изменится в работе процессора, если в каждой микропрограмме мик рокоманду увеличения программного счетчика PC := PC + 1 переместить в самый конец микропрограммы?

Таблица 9.11. Состояние модели в режиме моделирования на уровне микрокоманд

Адрес (PC)			С)ЗУ		CR			А У		Яче 'II	IIK
			MAR	MDR	COP	TA	ADR	Acc	DR	020	030
000	RD #20	MAR := PC	000	000000	00	0	000	000000	000000	000000	000000

		MRd	000								
		CR := MDR		211020							
		PC := PC + 1			21	1	020				
001		Ace := 000.ADR									
	WR 30	MAR := PC						000020			
		MRd	001								
		CR := MDR		220030							
		PC := PC + 1			22	0	030				
002		MAR := ADR									
		MDR := Ace	030								
		MWr		000020							
	ADD #5	MAR := PC									000020
		MRd	002								
		CR := MDR		231005							
		PC := PC + 1			23	1	005				
003		DR := 000.ADR									
		FAy:=ALI							000005		
	WR @30	MAR := PC						000025			

Лабораторная работа № 6. Программирование внешних устройств

Целью этой лабораторной работы является изучение способов организации взаимодействия процессора и внешних устройств (ВУ) в составе ЭВМ.

Выше отмечалось, что связь процессора и ВУ может осуществляться в синхронном или асинхронном режиме. Синхронный режим используется для ВУ, всегда готовых к обмену. В нашей модели такими ВУ являются дисплей и тоногенератор — процессор может обращаться к этим ВУ, не анализируя их состояние (правда дисплей блокирует прием данных после ввода 128 символов, формируя флаг ошибки).

Асинхронный обмен предполагает анализ процессором состояния ВУ, которое определяет готовность ВУ выдать или принять данные или факт осуществления некоторого события, контролируемого системой. К таким устройствам в нашей модели можно отнести клавиатуру и блок таймеров.

Анализ состояния ВУ может осуществляться процессором двумя способами:

- в программно-управляемом режиме;
- в режиме прерывания.

В первом случае предполагается программное обращение процессора к регистру состояния ВУ с последующим анализом значения соответствующего разряда слова состояния. Такое обращение следует предусмотреть в программе с некоторой периодичностью, независимо от фактического наступления контролируемого события (например, нажатие клавиши).

Во втором случае при возникновении контролируемого события ВУ формирует процессору запрос на прерывание программы, по которому процессор и осуществляет связь с ВУ.

6.1. Задание 6

Свой вариант задания (табл. 9.12) требуется выполнить двумя способами — сначала в режиме программного контроля, далее модифицировать программу таким образом, чтобы события обрабатывались в режиме прерывания программы. Поскольку "фоновая" (основная) задача для этого случая в заданиях отсутствует, роль ее может сыграть "пустой цикл":

M: NOP NOP JMP M

Таблица 9.12. Варианты задания 6

№ варианта	Задание	Используемые ВУ	Пояснения
1	Ввод пятиразрядных чисел в ячейки ОЗУ	Клавиатура	Программа должна обеспечивать ввод последовательности ASCII-кодов десятичных цифр (не длиннее пяти), перекодировку в "8421", упаковку в десятичное число (первый введенный символ — старшая цифра) и размещение в ячейке ОЗУ. ASCII-коды не-цифр игнорировать
2	Программа ввода символов с клавиатуры с выводом на дисплей	Клавиатура, дисплей, таймер	Очистка буфера клавиатуры после ввода 50 символов или каждые 10 с
3	Вывод на дисплей трех текстов, хранящихся в памяти, с задержкой	Дисплей, таймер	Первый текст выводится сразу при запуске программы, второй — через 15 с. третий — через 20 с после второго
4	Вывод на дисплей одного из трех текстовых сообщений, в зависимости от нажатой клавиши	Клавиатура, дисплей	<1>— вывод на дисплей первого текстового сообщения. <2> — второго. <3> — третьего, остальные символы — нет реакции

5 6	Выбирать из потока ASCII-кодов только цифры и выводить их на дисплей Выводить на дисплей каждый введенный с клавиатуры символ, причем цифру выводить "в трех экземплярах"	Клавиатура, дисплей, тоногенератор Клавиатура, дисплей, тоногенератор	Вывод каждой цифры сопровождается коротким звуковым сигналом Вывод каждой цифры сопровождается троекратным звуковым сигналом
7	Селективный ввод символов с клавиатуры	Клавиатура, дисплей	Все русские буквы, встречающиеся в строке ввода — в верхнюю часть экрана дисплея (строки 1—4), все цифры — в нижнюю часть экрана (строки 5—8), остальные символы не выводить
8	Вывод содержимого заданного участка памяти на дисплей посимвольно с заданным промежутком времени между выводами символов	Дисплей, таймер	Остаток от деления на 256 трех младших разрядов ячейки памяти рассматривается как ASCII-код символа. Начальный адрес памяти, длина массива вывода и промежуток времени— параметры подпрограммы
9	Программа ввода символов с клавиатуры с выводом на дисплей	Клавиатура, дисплей	Очистка буфера клавиатуры после ввода 35 символов

№ варианта	Задание	Используемые ВУ	Пояснения
10	Выводить на дисплей каждый введенный с клавиатуры символ, причем заглавную русскую букву выводить "в двух экземплярах"	Клавиатура, дисплей, таймер	Очистка буфера клавиатуры после ввода 48 символов, очистка экрана каждые 15с
11	Вывод на дисплей содержимого группы ячеек памяти в числовой форме (адрес и длина группы — параметры подпрограммы)	Дисплей, таймер	Содержимое ячейки распаковывается (с учетом знака), каждая цифра преобразуется в соответствующий ASCII-код и выдается на дисплей. При переходе к выводу содержимого очередной ячейки формируется задержка 10 с
12	Определить промежуток времени между двумя последовательными нажатиями клавиш	Клавиатура, таймер	Результат выдается на OR. (Учитывая инерционность модели, нажатия не следует производить слишком быстро.)

6.2. Задания повышенной сложности

1. Разработать программу-тест на скорость ввода символов с клавиатуры. По звуковому сигналу включается клавиатура и таймер на T секунд. Можно начинать ввод символов, причем каждый символ отображается на дисплее, ведется подсчет количества введенных

символов (после каждых 50 дается команда на очистку буфера клавиатуры, после 128 — очищается дисплей). Переполнение таймера выключает клавиатуру и включает сигнал завершения ввода (можно тон этого сигнала сопоставить с количеством введенных символов). Параметр T вводится из IR. Результат S — средняя скорость ввода (символ/с) выдается на OR. Учитывая, что модель учебной ЭВМ оперирует только целыми числами, можно выдавать результат в формате 5x60 символов/мин.

2. Разработать программу-тест на степень запоминания текста. Три различных вариантах текста выводятся последовательно на дисплей на Tx секунд с промежутками T2 секунд. Далее эти тексты (то, что запомнилось) вводятся с клавиатуры (в режиме ввода строки) и программно сравниваются с исходными текстами. Выдается количество (процент) ошибок.

3. Разработать программу-калькулятор. Осуществлять ввод из буфера клавиатуры последовательности цифр, упаковку (см. задание 1 в табл. 9.12).

Разделители — знаки бинарных арифметических операций и =. Результат переводится в ASCII-коды и выводится на дисплей.

6.3. Порядок выполнения работы

1. Запустить программную модель учебной ЭВМ и подключить к ней определенные в задании внешние устройства (меню Внешние устройства | Менеджер ВУ).

2. Написать и отладить программу, предусмотренную заданием, с использованием программного анализа флагов готовности ВУ. Продемонстрировать работающую программу преподавателю.

3. Изменить отлаженную в п. 2 программу таким образом, чтобы процессор реагировал на готовность ВУ с помощью подсистемы прерывания. Продемонстрировать работу измененной программы преподавателю.

6.4. Содержание отчета

1. Текст программы с программным анализом флагов готовности ВУ.

2. Текст программы с обработчиком прерывания.

6.5. Контрольные вопросы

1. При каких условиях устанавливается и сбрасывается флаг готовности клавиатуры Rd?

2. Возможно ли в блоке таймеров организовать работу всех трех таймеров с разной тактовой частотой?

3. Как при получении запроса на прерывание от блока таймеров определить номер таймера, достигшего состояния 99 999 (00 000)?

4. Какой текст окажется на экране дисплея, если после нажатия в окне обозревателя дисплея кнопки Очистить и загрузки по адресу CR (11) константы #10 вывести по адресу DR (10) последовательно пять ASCII-кодов русских букв А, Б, В, Г, Д?

5. В какой области памяти модели ЭВМ могут располагаться программы — обработчики прерываний?

6. Какие изменения в работе отлаженной вами второй программы произойдут, если завершить обработчик прерываний командой ret, а не iret?

Лабораторная работа № 7. Принципы работы кэш-памяти

В разд. 8.8 данной книги описаны некоторые алгоритмы замещения строк кэш-памяти. Цель настоящей лабораторной работы — проверить работу различных алгоритмов замещения при различных режимах записи.

7.1. Задание 7

В качестве задания предлагается некоторая короткая "программа" (табл. 9.14), которую необходимо выполнить с подключенной кэш-памятью (размером 4 и 8 ячеек) в шаговом режиме для следующих двух вариантов алгоритмов замещения (табл. 9.13).

Таблица 9.13. Пояснения к вариантам задания 7

Номера вариантов	Режим записи	Алгоритм замещения
1,7, П	Сквозная	СЗ, без учета бита записи
	Обратная	О, с учетом бита записи
2,5,9	Сквозная	БИ, без учета бита записи
	Обратная	О, с учетом бита записи
3,6, 12	Сквозная	О, без учета бита записи
	Обратная	СЗ, с учетом бита записи
4,8, 10	Сквозная	БИ, без учета бита записи
	Обратная	БИ, с учетом бита записи

Таблица 9.14. Варианты задания 7

			Номераі	соманд прог	раммы		
вари анта	1	2	3	4	5	6	7
1	RD #12	WR 10	WR @10	ADD 12	WR R0	SUB 10	PUSH R0
2	RD #65	WRR2	MOV R4,R2	WR 14	PUSH R2	POP R3	CALL 002
3	RD #16	SUB #5	WR 9	WR 09	WR R3	PUSH R3	POP R4
4	RD #99	WR R6	MOV R7,R6	ADD R7	PUSH R7	CALL 006	POP R8
5	RD #11	WR R2	WR -@R2	PUSH R2	CALL 005	POP R3	RET
6	RD #19	SUB #10	WR9	ADD #3	WR ©9	CALL 006	POPR4
			Номера	команд	програм	мы	
вари анта	1	2	3	4	5	6	7
7	RD #6	CALL 006	WR11	WRR2	PUSH R2	RET	JMP 002
8	RD#8	WRR2	WR @R2+	PUSH R2	POP R3	WR -@R3	CALL 003
9	RD #13	WR14	WR@14	WR@13	ADD 13	CALL 006	RET
10	RD #42	SUB #54	WR16	WR@16	WRR1	ADD 0R1+	PUSH Rl
11	RD #10	WRR5	ADD R5	WRR6	CALL 005	PUSH R6	RET
12	JMP 006	RD #76	WR 14	WRR2	PUSH R2	RET	CALL 001

Не следует рассматривать заданную последовательность команд как фрагмент программы1. Некоторые конструкции, например, последовательность команд push R6, ret в общем случае не возвращает программу в точку вызова подпрограммы. Такие группы команд введены в задание для того, чтобы обратить внимание студентов на особенности функционирования стека.

7.2. Порядок выполнения работы

1. Ввести в модель учебной ЭВМ текст своего варианта программы (см. табл. 9.14), ассемблировать его и сохранить на диске в виде txt-файла.

2. Установить параметры кэш-памяти размером 4 ячейки, выбрать режим записи и алгоритм замещения в соответствии с первой строкой своего варианта из табл. 9.13.

3. В шаговом режиме выполнить программу, фиксируя после каждого шага состояние кэш-памяти.

4. Для одной из команд записи (WR) перейти в режим Такт и отметить, в каких микрокомандах происходит изменение кэш-памяти.

5. Для кэш-памяти размером 8 ячеек установить параметры в соответствии со второй строкой своего варианта из табл. 9.13 и выполнить программу в шаговом режиме еще раз, фиксируя последовательность номеров замещаемых ячеек кэш-памяти.

7.3. Содержание отчета

1. Вариант задания — текст программы и режимы кэш-памяти.

2. Последовательность состояний кэш-памяти размером 4 ячейки при однократном выполнении программы (команды 1—7).

3. Последовательность микрокоманд при выполнении команды wr с отметкой тех микрокоманд, в которых возможна модификация кэш-памяти.

4. Для варианта кэш-памяти размером 8 ячеек— последовательность номеров замещаемых ячеек кэш-памяти для второго варианта параметров кэшпамяти при двукратном выполнении программы (команды 1—7).

7.4. Контрольные вопросы

1. В чем смысл включения кэш-памяти в состав ЭВМ?

2. Как работает кэш-память в режиме обратной записи? Сквозной записи?

3. Как зависит эффективность работы ЭВМ от размера кэш-памяти?

4. В какую ячейку кэш-памяти будет помещаться очередное слово, если свободные ячейки отсутствуют?

5. Какие алгоритмы замещения ячеек кэш-памяти вам известны?

Лабораторная работа № 8. Алгоритмы замещения строк кэш-памяти

Цель работы— изучение влияния параметров кэш-памяти и выбранного алгоритма замещения на эффективность работы системы. Эффективность в данном случае оценивается числом кэш-попаданий по отношению к общему числу обращений к памяти. Учитывая разницу в алгоритмах в режимах сквозной и обратной записи, эффективность использования кэш-памяти вычисляется выражениям (8.2) и (8.3) соответственно для сквозной и обратной записи.

Очевидно, эффективность работы системы с кэш-памятью будет зависеть не только от параметров кэш-памяти и выбранного алгоритма замещения, но и от класса решаемой задачи. Так, линейные программы должны хорошо работать с алгоритмами замещения типа очередь, а программы с большим числом условных переходов, зависящих от случайных входных данных, могут давать неплохие результаты с алгоритмами случайного замещения. Можно предположить, что программы, имеющие большое число повторяющихся участков (часто вызываемых подпрограмм и/или циклов) при прочих равных условиях обеспечат более высокую эффективность применения кэш-памяти,

чем линейные программы. И, разумеется, на эффективность напрямую должен влиять размер кэш-памяти.

Для проверки высказанных выше предположений выполняется настоящая лабораторная работа.

8.1. Задание 8

В данной лабораторной работе все варианты задания одинаковы: исследовать эффективность работы кэш-памяти при выполнении двух разнотипных программ, написанных и отлаженных вами при выполнении лабораторных работ

№ 2 и 4.

8.2. Порядок выполнения работы

1. Загрузить в модель учебной ЭВМ отлаженную программу из лабораторной работы № 2.
2. В меню Работа установить режим Кэш-память.
3. В меню Вид выбрать команду Кэш-память, открыв тем самым окно Кэшпамять, в нем нажать первую слева кнопку на панели инструментов, открыв диалоговое окно Параметры кэш-памяти, и установить следующие параметры кэш-памяти: размер — 4, режим

записи — сквозная, алгоритм замещения — случайное, без учета бита записи (W).

4. Запустить программу в автоматическом режиме; по окончании работы просмотреть результаты работы кэш-памяти в окне Кэш-память, вычислить значение коэффициента эффективности K и записать в ячейку табл. 9.15, помеченную звездочкой.

5. Выключить кэш-память модели (Работа | Кэш-память) и изменить один из ее параметров — установить флаг с учетом бита записи (в окне Параметры кэш-памяти).

6. Повторить п. 4, поместив значение полученного коэффициента эффективности в следующую справа ячейку табл. 9.15.

7. Последовательно меняя параметры кэш-памяти, повторить пп. 3—5, заполняя все ячейки табл. 9.15.

8. Повторить все действия, описанные в пп. 1—7 для программы из лабораторной работы № 4, заполняя вторую таблицу по форме табл. 9.15.

8.3. Содержание отчета

1. Две таблицы по форме табл. 9.15 с результатами моделирования программ из лабораторных работ № 2 и 4 при разных режимах работы кэш-памяти.

2. Выводы, объясняющие полученные результаты.

8.4. Контрольные вопросы

1. Как работает алгоритм замещения очередь при установленном флажке С учетом бита записи в диалоговом окне Параметры кэш-памяти?

2. Какой алгоритм замещения будет наиболее эффективным в случае применения кэш-памяти большого объема (в кэш-память целиком помещается программа)?

3. Как скажется на эффективности алгоритмов замещения учет значения бита записи W при работе кэш-памяти в режиме обратной записи? Сквозной записи?

4. Для каких целей в структуру ячейки кэш-памяти включен бит использования. Как устанавливается и сбрасывается этот бит?

Таблица 9.15. Результаты эксперимента

Способ	Сквозная сь запи					
Алгоритм	Случайное замещен ие		Оче редь		1>н і и	
Размер	без\У	с\V	6е3W	с\V	безУУ	с\V
4	*					
8						
16						
32						
Способ	Обратная 1СЬ за					
Алгоритм	Случайное замещение		Оче редь		Бит и	
Размер	без\¥	с\V	без\У	cW	без\¥	cW
4						
8						
16						
32						