МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра программной инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«15» 12 2017г.

АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНЫХ МАШИН И СИСТЕМ

Методические указания и задания к выполнению
лабораторных работ
для студентов направлений подготовки 09.03.04 «Программная инженерия» и 09.03.02 «Информационные системы и технологии»

Курск 2017

27.05.17

УДК 004.2

Составители: А.П.Жмакин, В.В. Ефремов, И.Н. Ефремова

Рецензент
Кандидат технических наук, доцент *Е.И.Аникина*

Архитектура вычислительных машин и систем: методические указания к выполнению лабораторных работ / Юго-Зап. гос. ун-т; сост.: А.П.Жмакин, В.В.Ефремов, И.Н. Ефремова. Курск, 2017. - 24 с.: 12 табл.

Содержат формулировку заданий к лабораторной работе работе, методические рекомендации по выполнению задания, а также требования к содержанию и оформлению отчёта и контрольные вопросы.

Предназначены для студентов направлений подготовки 09.03.04 «Программная инженерия» и 09.03.02 «Информационные системы и технологии»

Текст печатается в авторской редакции

Подписано в печать 15.12.17 Формат 60х84 1/16.
Усл. печ. л. 1,8. Уч.-изд. л. 1,7. Тираж 100 экз. Заказ 48. Бесплатно 4888
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Содержание

Лабораторные работы

Цикл лабораторных работ полностью основан на учебнике Жмакин А. П. Архитектура ЭВМ [Текст]: учебное пособие / А. П. Жмакин. - СПб.: БХВ-Петербург, 2006. - 320 с..

Цикл включает работы различного уровня. Лабораторные работы № 1—4 ориентированы 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**.**

Описание работы включает постановку задачи, пример выполнения, набор вариантов индивидуальных заданий, порядок выполнения работы, требования к содержанию отчета и контрольные вопросы.

1. Лабораторная работа № 1. Архитектура ЭВМ и система команд

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. Ввод программы может осуществляться как в машинных кодах непосредственно в память модели, так и в мнемокодах в окно **программы** с последующим ассемблированием.

Цель настоящей лабораторной работы — знакомство с интерфейсом ЭВМ, методами ввода и отладки программы, действиями основных команд и способов адресации. Для этого необходимо ввести в память ЭВМ и выполнить в режиме **Шаг** некоторую последовательность команд (определённую вариантом задания) и зафиксировать все

изменения на уровне программно-доступных объектов ЭВМ, происходящие при выполнении команд.

Команды в память учебной ЭВМ вводятся в виде шестиразрядных десятичных чисел.

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. 1)

Таблица 1. Команды и Коды

Последовательность	**Значения**				
Команды	RD #20	WR 30	ADD #5	WR @30	JNZ 002
Коды	211020	22030	231005	222030	120002

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 Acc, PC и ячейки ОЗУ 020 и 030) в табл. 2.

Таблица 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.
2. Записать в ОЗУ "программу", состоящую из пяти команд— варианты задания выбрать из табл. 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. 2).
6. Если в программе образуется цикл, необходимо просмотреть не более двух повторений каждой команды, входящей в тело цикла.

Таблица 3. Варианты задания 1

№	IR	Команда 1	Команда 2	Команда 3	Команда 4	Команда 5
1	000007	IN	MUL #2	WR 10	WR @10	JNS 001
2	X	PD #17	SUB #9	WR 16	WR @16	JNS 001
3	100029	IN	ADD #16	WR 8	WR @8	JS 001
4	X	RD #2	MUL #6	WR 11	WR @11	JNZ 00
5	000016	IN	WR 8	DIV #4	WR @8	JMP 002
6	X	RD #4	WR 11	RD @11	ADD #330	JS 000
7	000000	IN	WR 9	RD @9	SUB #1	JS 001
8	X	RD 4	SUB #8	WR 8	WR @8	JNZ 001
9	100005	IN	ADD #12	WR 10	WR @10	JS 004
10	X	RD 4	ADD #15	WR 13	WR @13	JMP 001
11	00315	IN	SUB #308	WR 11	WR @11	JMP 001
12	X	RD #988	ADD #19	WR 9	WR @9	JNZ 001
13	000017	IN	WR 11	ADD 11	WR @11	JMP 002
14	X	RD #5	MUL #9	WR 10	WR @10	JNZ 002

9.1.4. Содержание отчета

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

9.1.5. 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 отличия между ними?
6. Какие ограничения накладываются на способ представления длины 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?

2. 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

причем ***x*** вводится с устройства ввода IR, результат ***y*** выводится на OR.

Граф-схема алгоритма решения задачи показана на рис. 1.

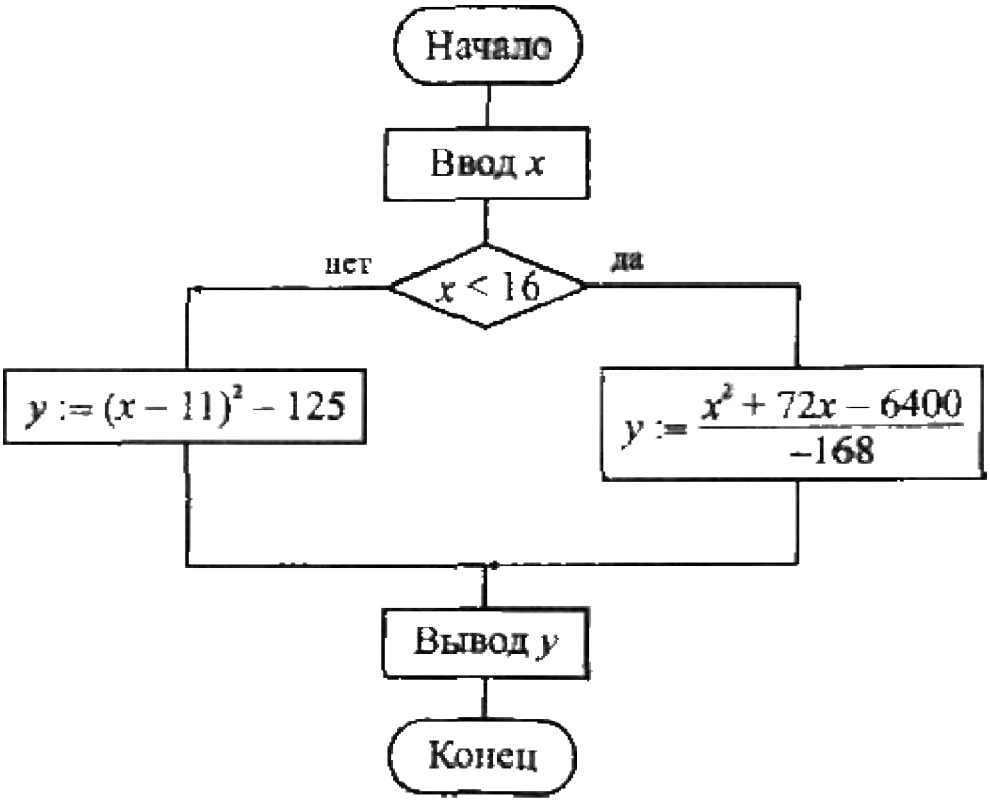

Рис 1. Граф-схема алгоритма

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. 4.

Таблица 4. Пример программы

Адрес	Команда		Примечание
	Мнемокод	**Код**	
000	IN	010000	Ввод x
001	WR 30	220030	Размещение x в ОЗУ (ОЗО)
002	SUB #16	241016	Сравнение с границей – (x-16)
003	JS 010	130010	Переход по отрицательной разности
004	RD 30	210030	Вычитание по первой формуле
005	SUB #11	241011	
006	WR 31	220031	
007	MUL 31	250031	
008	SUB #125	241125	
009	JMP 020	100020	Переход на выход результата
010	RD 30	210030	Вычисления по второй формуле
011	MUL 30	250030	
012	WR 31	220031	
013	RD 30	210030	
014	MUL #72	251072	
015	ADD 31	230031	
016	ADI 106400	430000	
017		106400	
018	DIVI 100168	460000	
019		100169	
020	OUT	020000	Вывод результата
021	HLT	090000	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. 5, варианты заданий — в табл. 6.
2. Исходя из допустимых пределов изменения аргумента функций (табл. 5) и значения параметра *a* для своего варианта задания (табл. 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 п. 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 п. *5, а* — 5, в;
е) записать в IR значение аргумента $x < a$ (в области допустимых значений); выполнить п. 5, б и 5, в;
ж) записать в IR недопустимое значение аргумента x и выполнить п. 5, *б* и 5, *е.*
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 5. Функции

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 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?

3. 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 $C_1, C_2 \ldots$ Cn. Исходными данными в этой задаче являются: n — количеств суммируемых чисел и $C_1, C_2,\ldots, Cn$ — массив суммируемых чисел. Заметем, что должно выполняться условие $n > 1$, 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: A*i* — адрес числа C*i*, *i* □ {l, 2,…,10}; ОЗУ(A*i*) — число по адресу A*i, 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: *Ai* — 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.

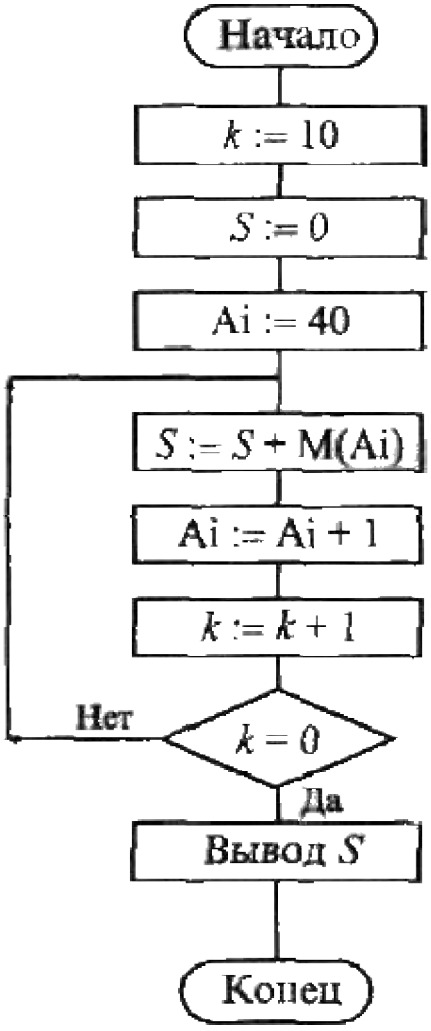

Рис 2. Граф-схема алгоритма для примера 3

Таблица 7. Текст программы примера 3

Адрес	Команда	Примечание
000	RD #40	Загрузка начального адреса массива 040
001	WR 30	в ячейку 030
002	RD #10	Загрузка параметра цикла k=10 в ячейку 031
003	WR 31	
004	RD #0	Загрузка начального значения суммы S=0
005	WR 32	в ячейку 032
006	M1: RD 32	Добавление
007	ADD @30	к текущей сумме
008	WR 32	очередного элемента массива
009	RD 30	Модификация текущего
010	ADD #1	адреса массива
011	WR 30	(переход к следующему адресу)
012	RD 31	Уменьшение счетчика
013	SUB #1	(параметр цикла)
014	WR 31	на 1
015	JNZ M1	Проверка параметра цикла и переход при k≠0
016	RD 32	Вывод
017	OUT	результата
018	NLT	Стоп

3.2. Задание 3

1. Написать программу определения заданной характеристики последовательности чисел C_1, C_2,…,C_n . Варианты заданий приведены в табл. 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 8. Вариант задания 3

Номер варианта	Характеристика последовательности чисел $C_1, C_2, \dots, C_n$
1	Количество чётных чисел
2	Номер минимального числа
3	Произведение всех чисел
4	Номер первого отрицательного числа
5	Количество чисел, равных C_1

6	Количество отрицательных чисел
7	Максимальное отрицательное число
8	Номер первого положительного числа
9	Минимальное положительное число
10	Номер максимального числа
11	Количество нечётных чисел
12	Количество чисел, меньших C_1
13	Разность сумм чётных и нечётных элементов массивов
14	Отношение сумм чётных и нечётных элементов массивов

Примечание. Под четными (нечетными) элементами массивов понимаются элементы массивов, имеющие четные (нечетные) индексы. Четные числа — элементы массивов, делящиеся без остатка на 2.

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. 7, если в ней будет отсутствовать команда WR 31 по адресу 014?
4. Как поведет себя программа, приведенная в табл. 7, если метки Ml будет поставлена по адресу 005? 007?

4. Лабораторная работа № 4. Подпрограммы и стек

В программировании часто встречаются ситуации, когда одинаковые действия необходимо выполнять многократно в разных частях программы (например, вычисление функции sin x). 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 подпрограммы и возврат в точку вызова по завершению подпрограммы.

Для *вызова* подпрограммы необходимо указать ее начальный адрес и 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 неподвижной 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 предекрементная (см. табл. 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.

9.4.1. 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— 110 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 массивом Подпрограмма получает параметры через регистры R1 (начальный адрес массива) и R2 (длина массива). Эти регистры используются подпрограммы 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. 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
CALL M	Вызов подпрограммы
WR R7	Сохранение результата
RD #110	Загрузка
WR R1	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 @R1+	Чтение элемента и модификация адреса
WR R4	Сравнение
SUB R3	и замена,
JS L1	если R3<R4
MOV R3,R4	
L1: JRNZ R2, L2	Цикл
RD R3	Чтение результата в Acc
RET	Возврат

4.2. Задание 4

Составить и отладить программу учебной ЭВМ для решения следующей дачи. 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 вариант заданий 3 и 4 устанавливается по табл. 9.10.

Таблица 10. Соответствия 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 предекрементной адресации, организовать дополнительный стек в произвольной области памяти, не связанный с SP ?

5. 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. 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**
□ **JRNZ R3,M**
□ **MOV R4,R2**
□ **JMP M**
□ **CALL M**
□ **RET: PUSH R3**
□ **POP R5**

9.5.3. 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 микрокоманд, реализующих команду) для несуществующей команды "умножение модулей чисел".
4. Что изменится в работе процессора, если в каждой микропрограмме микрокоманду увеличения программного счетчика PC := PC + 1 переместить в самый конец микропрограммы?

Таблица 11. Состояние модели в режиме моделирования на уровне микрокоманд

ОЗУ		CR			АУ		Ячейки	
MAR	MDR	COP	TA	AD	Acc	DR	020	030
000	000000	00	0	000	000000	000000	000000	000000
000								
	211020							
		21	1	020				
					000020			
001								
	220030							
		22	0	030				
030								
	000020							
								000020
002								
	231005							
		23	1	005				
						000005		

Адрес (PC)	Мнемокод	Микрокоманда
000	RD #20	MAR:=PC
		MRd
		CR:=MDR
		PC:=PC+1
001		Acc:=000.ADR
	WR 30	MAR:=PC
		MRd
		CR:=MDR
		PC:=PC+1
002		MAR:=ADR
		MDR:=Acc
		MWr
	ADD #5	MAR:=PC
		MRd
		CR:=MDR
		PC:=PC+1
003		DR:=000.ADR
		$F_{AY:=}$ALI

6. 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 программ таким образом, чтобы события обрабатывались в режиме прерывания программы. Поскольку "фоновая" (основная) задача для этого случая в задание отсутствует, роль ее может сыграть "пустой цикл":

```
M: NOP
   NOP
   JMP M
```

Таблица 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 введённый символ – старшая цифра) и размещение в ячейке ОЗУ. ASCII-коды не-цифры 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 трёх текстовых сообщений, в зависимости от нажатой клавиши	Дисплей, дисплей	<1> - вывод на дисплей первого текстового сообщения, <2> - второго, <3> - третьего, остальные символы – нет реакции
5	Выбирать из потока ASCII-кодов только цифры и выводить их на дисплей	Клавиатура, дисплей, тоногенератор	Вывод каждой цифры сопровождается коротким звуковым сигналом
6	Выводить на дисплей каждый введённый с клавиатуры символ,	Клавиатура, дисплей, тоногенератор	Вывод каждой цифры сопровождается троекратным звуковым сигналом

	причём цифру выводить «в трёх экземплярах»		
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 трёх 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	Отчистка буфера клавиатуры после ввода 35 символов

Таблица 12. (Окончание)

№ варианта	**Задание**	**Используемые ВУ**	**Пояснения**
10	Выводить на дисплей каждый введённый с клавиатуры символ, причём заглавную русскую букву выводить «в двух экземплярах»	Клавиатура, дисплей, таймер	Отчистка буфера клавиатуры после ввода 48 символов, очистка экрана каждые 15 с
11	Вывод на дисплей содержимого группы ячеек памяти в числовой форме (адрес и длина группы – параметры подпрограммы)	Дисплей, таймер	Содержимое ячейки распаковывается (с учётом знака), каждая цифра преобразуется в соответствующий ASCII-код и выдаётся 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 выдаётся на OR. (Учитывая инерционность модели, нажатия не следует производить слушком 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 введённых символов). Параметр *T* вводится из IR. Результат *S* — средняя скорость ввода (символ/с) выдается на OR. Учитывая, что модель учебно 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 T_1 секунд с промежутками T_2 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 =. Результатi 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?**

Список используемых источников

1. Мелехин, Виктор Федорович . Вычислительные машины, системы и сети [Текст] : учебник / В. Ф. Мелехин, Е. Г. Павловский. - 3-е изд., стер. - М. : Академия, 2010. - 560 с.

2. Олифер В. Г. Компьютерные сети. Принципы, технологии, протоколы [Текст] : учебник для вузов / В. Г. Олифер, Н. А. Олифер. - 4-е изд. - Санкт-Петербург : Питер, 2015. - 943 с.

3. Громов Ю.Ю. Архитектура ЭВМ и систем [Электронный ресурс] : учебное пособие / Ю.Ю. Громов, О. Г. Иванова, М. Ю. Серегин. - Тамбов : Издательство ФГБОУ ВПО «ТГТУ», 2012. - 200 с. .- Режим доступа: http://biblioclub.ru/

4. Бройдо В. Л. Архитектура ЭВМ и систем [Текст] : учебник для вузов / В. Л. Бройдо, О.П. Ильина. - 2-е изд. - СПб.: Питер, 2009. - 720 с.

5. Хорошевский В. Г. Архитектура вычислительных систем [Текст] : учебное пособие / В. Г. Хорошевский. - М.: МГТУ им. Н. Э. Баумана, 2005. - 512 с.

6. Жмакин А. П. Архитектура ЭВМ [Текст]: учебное пособие / А. П. Жмакин. - СПб.: БХВ-Петербург, 2006. - 320 с.

7. Таненбаум Э. С. Архитектура компьютера [Комплект] / Э. Таненбаум. - 5-е изд. - СПб.: Питер, 2010. - 844 с.