Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ

Кафедра компьютерных систем в управлении и проектировании (КСУП)

Е.А. Потапова

ИНФОРМАТИКА. АССЕМБЛЕР ДЛЯ ПРОЦЕССОРА I8086

Методические указания по выполнению контрольных и лабораторных работ

2013

Корректор: Осипова Е.А.

Потапова Е.А.
Информатика. Ассемблер для процессора i8086: методические указания по выполнению контрольных и лабораторных работ. — Томск: Факультет дистанционного обучения, ТУСУР, 2013. — 50 с.

ОГЛАВЛЕНИЕ

1 ВВЕДЕНИЕ

Цель курса — создание основы (базиса) для изучения и использования вычислительных систем в других курсах. Данная цель достигается путем обучения основам программирования на языке ассемблера для микропроцессора Intel 8086 (сокращенно — i8086).

Выбор в качестве объекта изучения именно этого языка обусловлен следующим. Во-первых, только ассемблер позволяет получить представление об организации и функционировании аппаратуры ЭВМ. Другие языки программирования не предоставляют (или почти не предоставляют) такой возможности. Во-вторых, среди языков ассемблера данный язык является наиболее распространенным. Несмотря на то, что сам процессор i8086 практически стал достоянием компьютерной истории, его машинный язык аппаратно поддерживается в последующих моделях фирмы INTEL. Собственные машинные языки (и языки-ассемблеры) этих моделей включают язык для i8086 в качестве своего подмножества, отличаясь от него большей сложностью, которая делает их малопригодными для первоначального знакомства с предметом.

Выбор варианта контрольной работы № 2 и лабораторных работ осуществляется по общим правилам с использованием следующей формулы:

$$V = (N*K) \text{ div } 100,$$

где V — искомый номер варианта,

N — общее количество вариантов,

div — целочисленное деление (после деления дробная часть отбрасывается),

при V= 0 выбирается максимальный вариант,

K — значение 2-х последних цифр пароля.

При изучении данного курса Вам потребуется:

1) операционная система MS-DOS. Допускается применение ее в самостоятельном режиме или под управлением операционной системы Windows;

2) Norton Commander или другая подобная утилита;

3) DEBUG — отладчик;

4) NASM — транслятор ассемблера i8086.

2 МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

Настоящие методические указания и учебное пособие [1] содержат информацию, достаточную для выполнения лабораторных и контрольных работ. Для более подробного рассмотрения излагаемых вопросов рекомендуется обратиться к литературе, список которой приведен в данном пособии.

Данный лабораторный курс состоит из двух частей. В процессе выполнения первой части (лабораторная работа № 1) производится знакомство с языком, занимающим промежуточное положение между машинным языком и ассемблером. Обладая ассемблерной формой записи кодов операций и регистров, данный язык не имеет ассемблерных псевдооператоров и использует численные (не символьные) адреса. Знакомство с данным языком производится с помощью отладчика Debug и прежде всего предназначено для изучения механизма выполнения процессором машинных программ. Кроме того, создается основа для последующего написания и отладки ассемблерных программ.

Во второй части курса (лабораторная работа № 2) производится знакомство с основами программирования на языке ассемблера. При этом объектами рассмотрения являются не только операторы ассемблера, но и методы проектирования и отладки программ.

Для того, чтобы успешно выполнить лабораторные и контрольные работы, внимательно изучите учебное пособие [1], выполняйте все задания, предлагаемые к исполнению.

Изучите тему 1 пособия [1]. Представление информации.

Выполните задание:

Без использования калькулятора выполните следующие действия над двумя четырехзначными целыми положительными десятичными числами:

1) запишите исходные числа в двоичной системе счисления;

2) найдите сумму двоичных чисел из п. 1;

3) найдите разность двоичных чисел из п. 1 (от меньшего числа отнять большее);

4) запишите исходные числа в шестнадцатеричной системе счисления;

5) найдите сумму шестнадцатеричных чисел из п. 4;

6) найдите разность шестнадцатеричных чисел из п. 4 (от большего числа отнять меньшее число).

Примечание. В качестве проверки рекомендуется найти сумму результатов п. 3 и п. 6, предварительно представив их в одной и той же системе счисления (двоичной или шестнадцатеричной). Указанная сумма должна быть равна 0. В самом деле, если обозначим исходные числа как A и B, то результат п. 3: $A - B$, а результат п. 6: $B - A$. Найдем искомую сумму: $(A - B) + (B - A) = 0$.

ЛАБОРАТОРНАЯ РАБОТА № 1
АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ, ВЫВОД СИМВОЛОВ, ВЫВОД ДВОИЧНЫХ ЧИСЕЛ

Цель работы

Целью настоящей работы является первоначальное знакомство с программой Debug — важнейшим помощником разработчика программ на языке Ассемблер. С помощью этой программы производится анализ и заполнение ячеек регистровой и оперативной памяти, осуществляется пошаговое выполнение программы. Другая цель: знакомство с некоторыми инструкциями Ассемблера, выполняющими арифметические операции, знакомство с инструкциями программного прерывания, с инструкциями пересылки данных, операторами сдвигов, операторами циклов.

В процессе выполнения работы решается практически важная задача вывода чисел на экран, осуществляется вывод на экран двоичного числа в виде последовательности единиц и нулей.

Для успешного выполнения лабораторной работы № 1 нужно изучить из пособия [1] темы 3. ПРОГРАММИРОВАНИЕ АРИФМЕТИЧЕСКИХ ОПЕРАЦИЙ, 4. ВЫВОД СИМВОЛОВ НА ЭКРАН, 5. ВЫВОД НА ЭКРАН ДВОИЧНЫХ ЧИСЕЛ. Для формирования строк символов и вывода их на экран воспользуйтесь значениями из таблицы 1 и таблицы 2.

Таблица 1 — **Коды ASCII**

Символ ASCII	16-рич. код	Символ ASCII	16-рич. код	Символ ASCII	16-рич. код	Символ ASCII	16-рич. код
	20	8	38	P	50	H	68
!	21	9	39	Q	51	I	69
“	22	:	3A	R	52	J	6A
#	23	;	3B	S	53	k	6B
$	24	<	3C	T	54	L	6C
%	25	=	3D	U	55	M	6D
&	26	>	3E	V	56	n	6E
‘	27	?	3F	W	57	o	6F
(	28	@	40	X	58	p	70
)	29	A	41	Y	59	q	71
*	2A	B	42	Z	5A	r	72

Окончание табл. 1

Символ ASCII	16-рич. код	Символ ASCII	16-рич. код	Символ ASCII	16-рич. код	Символ ASCII	16-рич. код
+	2B	C	43	[	5B	s	73
,	2C	D	44	\	5C	t	74
–	2D	E	45	]	5D	u	75
.	2E	F	46	^	5E	v	76
/	2F	G	47	_	5F	w	77
0	30	H	48	`	60	x	78
1	31	I	49	a	61	y	79
2	32	J	4A	b	62	z	7A
3	33	K	4B	c	63	{	7B
4	34	L	4C	d	64	|	7C
5	35	M	4D	e	65	}	7D
6	36	N	4E	f	66	~	7E
7	37	O	4F	g	67	DEL	7F

Таблица 2 — **Коды букв русского алфавита**

Символ	Код(16)	Символ	Код(16)	Символ	Код(16)	Символ	Код(16)
А	80	Р	90	а	A0	р	E0
Б	81	С	91	б	A1	с	E1
В	82	Т	92	в	A2	т	E2
Г	83	У	93	г	A3	у	E3
Д	84	Ф	94	д	A4	ф	E4
Е	85	Х	95	е	A5	х	E5
Ж	86	Ц	96	ж	A6	ц	E6
З	87	Ч	97	з	A7	ч	E7
И	88	Ш	98	и	A8	ш	E8
Й	89	Щ	99	й	A9	щ	E9
К	8A	Ъ	9A	к	AA	ъ	EA
Л	8B	Ы	9B	л	AB	ы	EB
М	8C	Ь	9C	м	AC	ь	EC
Н	8D	Э	9D	н	AD	э	ED
О	8E	Ю	9E	о	AE	ю	EE
П	8F	Я	9F	п	AF	я	EF

Задание

Разработать с помощью Debug программу, выполняющую вывод на экран текстового сообщения и последующее вычисление выражения:

$$Y=[(X1+X2)X3–X4]/X5,$$

где X1—X5 — десятичные целые числа, выбранные в соответствии с номером варианта из таблицы 3.

Результат вычисления выражения программа помещает в регистры AX и DX. Нужно вывести эти результаты в двоичной системе счисления.

Структура выходного сообщения программы:

“Программа вычисления выражения Y=[(X1+X2)X3–X4]/X5, где X1=..., X2=..., X3=..., X4=..., X5=...”

AX=0011110111000101 DX=0000000000010101

Промежуточные результаты можно будет наблюдать при запуске программы в debug, выполняя пошаговое выполнение инструкций.

Вместо точек должны выводиться заданные числа (в шестнадцатеричной системе).

Таблица 3 **— Варианты заданий к лабораторной работе № 1**

№ варианта	X1	X2	X3	X4	X5	№ варианта	X1	X2	X3	X4	X5
1	275	361	15	2250	10	11	427	135	14	3221	15
2	413	228	14	4353	13	12	327	393	13	1929	12
3	199	328	16	3833	11	13	562	248	11	2520	17
4	355	436	13	4318	20	14	682	133	10	3563	22
5	359	537	11	5413	15	15	269	331	18	4151	14
6	227	199	19	2873	12	16	361	483	13	5621	22
7	436	327	13	3315	18	17	568	329	17	6209	21
8	521	263	12	2544	21	18	249	325	15	4683	16
9	324	391	17	4561	19	19	189	459	19	5394	19
10	614	134	17	8236	23	20	288	274	18	4815	20

Примечание 1. Загрузка в регистры заданных чисел (преобразованных вручную в шестнадцатеричную систему) должна производиться только с помощью инструкций MOV.

Примечание 2. Рекомендуется выполнить проверку результата выполнения программы путем сравнения его с результатом ручного счета. Так как при ручном счете используется десятичная система счисления, то перед сравнением результатов их необходимо записать в одной и той же системе.

Отчет по лабораторной работе должен содержать:

1. Титульный лист — форма титульного листа представлена в Приложении 4.

2. Задание на лабораторную работу. В задании должен быть указан номер варианта и представлены исходные данные из таблицы 3.

На проверку необходимо отправить каталог LAB1, в который нужно поместить:

1. Исполнимый файл программы, то есть файл с расширением .com. Имя файла может быть выбрано по Вашему усмотрению, но придерживайтесь правил именования файлов для DOS — имя должно состоять не более чем из 8 символов.

2. Отчет к лабораторной работе, выполненный с помощью редактора Word.

ЛАБОРАТОРНАЯ РАБОТА № 2
ВВОД-ВЫВОД ЧИСЕЛ, ПРОГРАММИРОВАНИЕ НА АССЕМБЛЕРЕ

Цель работы

В процессе выполнения работы решается практически важная задача вывода чисел на экран и их ввода с клавиатуры. Данная задача решается в следующей последовательности. Во-первых, рассматривается задача вывода на экран шестнадцатеричных чисел. Во-вторых, рассматривается ввод шестнадцатеричных чисел с клавиатуры.

В ходе работы производится знакомство с очень важными понятиями флагов состояния, стека и процедуры. Изучаются инструкции для работы с этими объектами, а также инструкции сдвига, цикла, условных переходов и некоторые другие.

Для успешного выполнения лабораторной работы № 2 нужно изучить из пособия [1] темы: 6. ВЫВОД НА ЭКРАН ЧИСЕЛ В ШЕСТНАДЦАТЕРИЧНОЙ ФОРМЕ, 9. ВВОД С КЛАВИАТУРЫ ШЕСТНАДЦАТЕРИЧНЫХ ЧИСЕЛ.

Одной из целей работы является развитие навыков алгоритмизации задач и отладки программ.

До сих пор нашим единственным помощником при написании и отладке машинных программ была системная программа Debug. Мы и далее будем широко использовать Debug при отладке своих программ. Что касается написания программы, то тут помощь Debug явно недостаточна, и процесс написания сколько-нибудь сложной программы скорее всего продлится очень долго. По этой причине мы переходим к написанию программ на языке ассемблера.

Целью выполнения данной работы является получение начальных навыков по разработке программ на языке ассемблера. А именно — рассматриваются псевдооператоры, позволяющие разрабатывать простые ассемблерные программы, а также производится первоначальное знакомство с системными программами (EDIT, NASM), обеспечивающими преобразование программы на языке ассемблера в машинную программу.

Для этого из пособия [1] изучите темы и выполните все задания к ним 11. ПРОСТЫЕ ПРОГРАММЫ НА АССЕМБЛЕРЕ,

12. ОСНОВНЫЕ ОПЕРАТОРЫ АССЕМБЛЕРА, 13. Пример программы на ассемблере, 14. ВЫВОД НА ЭКРАН ДЕСЯТИЧНЫХ И ШЕСТНАДЦАТЕРИЧНЫХ ЧИСЕЛ.

Задание

Разработать на ассемблере и отладить программу, которая выполняет:

1) ввод с клавиатуры двух 4-значных шестнадцатеричных чисел (для ввода с клавиатуры можно использовать любые числа), которые записываются в качестве содержимого регистров BP и DI;

2) вывод на экран содержимого регистров, заполненных на шаге 1, в виде шестнадцатеричных чисел;

3) вывод на экран содержимого регистров, заполненных на шаге 1, в виде десятичных чисел;

4) вывод на экран содержимого регистров, заполненных на шаге 1, в виде двоичных чисел.

Пример информации на экране:

ВВЕДИТЕ СОДЕРЖИМОЕ РЕГИСТРА BP AD56<Enter>

ВВЕДИТЕ СОДЕРЖИМОЕ РЕГИСТРА DI 7F09<Enter>

ЧИСЛА В ШЕСТНАДЦАТЕРИЧНОЙ СИСТЕМЕ

(BP) = AD56 (DI) = 7F09

ЧИСЛА В ДЕСЯТИЧНОЙ СИСТЕМЕ

(BP) = 44374 (DI) = 32521

ЧИСЛА В ДВОИЧНОЙ СИСТЕМЕ

(BP) = 1010110101010110 (DI) = 0111111100001001

Примечание 1. Файловая структура программы должна включать два файла типа .asm. В одном из них содержатся главная подпрограмма и тексты выводимых сообщений. Все остальные процедуры содержатся во втором файле.

Примечание 2. Все процедуры должны иметь вводные и текущие комментарии.

Примечание 3. Рекомендуется дополнительно разработать процедуру, выполняющую ввод шестнадцатеричного числа в 16-битный регистр, процедуру вывода содержимого такого реги-

стра в шестнадцатеричном виде, а также процедуру вывода содержимого 16-битного регистра в десятичной системе счисления и в двоичной системе счисления.

Для того чтобы реализовать вывод на экран чисел в десятичной системе счисления, воспользуйтесь алгоритмом вывода десятичного числа из [1].

Примечание 4. При реализации вывода второй и третьей шестнадцатеричных цифр числа, сдвигу числа вправо должен предшествовать его сдвиг влево. Для выполнения сдвига влево используйте инструкцию SHL («Shift Left» — логический сдвиг влево). Использование этой инструкции аналогично SHR. Выполнение SHL имеет такой же эффект, как и умножение на два, четыре, восемь и так далее, в зависимости от числа (соответственно единицы, двойки или тройки), хранящегося в CL.

Примечание 5. Для получения на экране достаточно хорошей формы представления информации выполняйте вывод промежуточных пробелов. Число пробелов определяйте опытным путем.

Отчет по лабораторной работе № 2 должен содержать:

1. Титульный лист — форма титульного листа представлена в Приложении 4.

2. Задание на лабораторную работу, для ввода с клавиатуры можно использовать любые числа.

3. Дерево подпрограмм. Пример дерева подпрограмм приведен в приложении 3 (рис. 4).

4. Файловую структуру программы. Примеры файловой структуры программы приведены в [1] рис. 56 и 57.

На проверку необходимо отправить каталог LAB2, в который нужно поместить:

1. Исходный файл программы, то есть файл с расширением .asm. Имя файла может быть выбрано по Вашему усмотрению, но придерживайтесь правил именования файлов для DOS — имя должно состоять не более чем из 8 символов.

2. Исполнимый файл программы, то есть файл с расширением .com

3. Отчет к лабораторной работе, выполненный с помощью редактора Word.

3 МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ПОДГОТОВКЕ К КОНТРОЛЬНЫМ РАБОТАМ

КОНТРОЛЬНАЯ РАБОТА № 1 ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ В ЭВМ И ЭЛЕМЕНТЫ ЯЗЫКА АССЕМБЛЕРА

Контрольная работа № 1 компьютерная и выполняется в диалоге с контролирующей программой. Ниже приводятся примеры решений заданий и методические рекомендации по их выполнению.

Задание 1. Получить двоичное представление заданного целого десятичного числа.

Указания. Для преобразования десятичного числа в двоичное можно использовать метод вычитаний.

Пример. Получить двоичное представление десятичного числа 5000.

Решение. Выполним вычитания:

```
_ 5000
  4096  (бит 12)
  _904
   512  (бит 9)
_  392
    256  (бит 8)
_  136
   128  (бит 7)
    _ 8
      8  (бит 3)
      0
```

Заполним биты двоичного числа, начиная с младшего:

1001110001000 (ответ).

Задание 2. Найти разность двух заданных положительных двоичных чисел.

Указания. Вычитание двух положительных чисел следует выполнить сложением первого из них с дополнением второго (отрицательного) числа. Прежде чем получать дополнение отрицательного числа, необходимо исходное положительное число дополнить слева незначащими нулями до байта или слова (в зависимости от величины числа). Иначе — знаковый бит числа (бит 7 для байта, бит 15 для слова) будет отсутствовать.

Пример. Найти разность двоичных чисел 1010100100111 и 110010010011001.

Решение. Расширим второе число до 16 бит: 0110010010011001. Найдем дополнительный код этого же числа, но взятого со знаком «–» :

1) инвертируем все биты: 1001101101100110

2) прибавляем 1: + 1

1001101101100111

Суммируем первое число с полученным дополнением:

1010100100111

\+ 1001101101100111

1011000010001110 (ответ)

Задание 3. Найти разность двух заданных положительных шестнадцатеричных чисел. (Из большего числа вычесть меньшее.)

Указания. Вычитание шестнадцатеричных чисел выполняется аналогично вычитанию десятичных чисел с той лишь разницей, что занимаемая из старшего разряда единица, в текущем разряде составляет не 10, а 16 единиц.

Пример. Найти разность шестнадцатеричных чисел A74F и 8E5C.

Решение: _A74F

8E5C

18F3 (ответ).

Задание 4. Пусть в данный момент времени некоторые регистры содержат:

(регистр 1) =XXXXh, . . ., (регистр n) = XXXXh

Каков (в шестнадцатеричной системе) физический адрес ячейки ОП, содержащей:

1) младший байт следующей исполняемой на ЦП инструкции;

2) младший байт вершины стека;

3) байт данных, обрабатываемый текущей инструкцией [инструкция].

Указания. Для получения физического адреса искомой ячейки необходимо просуммировать содержимое соответствующего регистра сегмента, умноженное на 16 (10h), с содержимым регистра, в котором находится внутрисегментное смещение.

Пример. Пусть в данный момент времени некоторые регистры содержат:

. . . . (DS) = 3FA5h . . . (BX) = 4A84h

Найти физический адрес ячейки ОП, содержащей байт данных, обрабатываемый текущей инструкцией: MOV [BX], AL

Решение.

R = (DS) x 16 + (BX) = 3FA5 x 10h + 4A84 = 3FA50 + 4A84 =

= **444D4** (ответ).

Задание 5. Пусть в данный момент времени некоторые регистры содержат:

(регистр 1) =XXXXh, . . ., (регистр n) = XXXXh

Каково будет содержимое указателя команды (или указателя стека) в результате выполнения следующих машинных инструкций [инструкции с указанием их длины].

Указания. Каждая выполняемая инструкция обработки данных влияет на содержимое указателя команды IP в соответствии со своей длиной, а инструкция передачи управления — в соответствии со своим адресом перехода.

На указатель стека SP влияют только некоторые инструкции, а именно: PUSH, POP, CALL, RET, INT, IRET, PUSHF, POPF. При этом следует учесть, что стек «растет» в сторону меньших адресов.

Пример. Пусть в данный момент времени некоторые регистры содержат:

. . . (SP) = FE4A .

Каково будет содержимое указателя стека после выполнения следующих инструкций:

PUSH AX (длина 1 байт)

PUSH BX (длина 1 байт)

RET (длина 1 байт)

CALL 200h (длина 3 байта)

Решение. Инструкции PUSH и CALL добавляют в стек по одному слову (2 байта), а инструкция RET берет слово из стека. Следовательно, длина стека увеличится на 4 байта. Новое содержимое регистра SP:

(SP) = FE4A – 4 = **FE46** (ответ).

Задание 6. Записать содержимое (в шестнадцатеричной системе) заданного регистра, полученное в результате выполнения следующих операторов ассемблера [операторы].

Указания. Для подготовки к выполнению задания рекомендуется использовать информацию о логических операторах.

Пример. Записать содержимое регистра BX, полученное в результате выполнения операторов:

```
MOV     BX, 0ABCDh
MOV     CX, 0707h
AND     BX, CX
```

Решение. Сначала запишем двоичное содержимое регистров BX и CX до выполнения операции AND:

(BX) = 1010101111001101

(CX) = 0000011100000111

Выполним операцию AND и переведем результат в шестнадцатеричную систему:

```
      1010101111001101
AND   0000011100000111
      0000001100000101
      [ 0 ][ 3 ] [ 0 ] [ 5 ]
```

Ответ: **305.**

Задание 7. Указать номер неправильного оператора передачи данных в следующем списке операторов [операторы MOV].

Указание. Для подготовки к выполнению задания рекомендуется использовать информацию о операторах передачи данных и о вспомогательных псевдооператорах.

Пример. Указать номер неправильного оператора в списке:

1) MOV AX, 0FFFFh
2) MOV AL, DL
3) MOV BH, OFFSET Table
4) MOV Arg, AX
5) MOV DL, BYTE PTR Table

Решение. Неправильным является оператор 3, так как для размещения смещения адреса (оно задается псевдооператором OFFSET) всегда требуются 16 бит памяти, то есть слово. А длина регистра BH — 8 битов. Ответ: **3**.

Задание 8. Записать содержимое (в шестнадцатеричной системе) заданного регистра, полученное в результате выполнения следующих операторов ассемблера [операторы].

Указания. Для подготовки к выполнению задания рекомендуется использовать информацию о операторах сдвига.

Пример. Записать содержимое регистра AL, полученное в результате выполнения операторов:

```
MOV     AL, 7Ah
STC
RCR     AL, 1
```

Решение. Определим содержимое регистра AL до выполнения оператора RCR:

(AL) = 7Ah = 01111010

Оператор RCR (циклический сдвиг вправо через перенос) выталкивает содержимое младшего бита первого операнда и помещает его в флаг CF, а прежнее содержимое CF заносит в старший бит операнда. Так как оператор STC поместил в CF 1, то в результате выполнения RCR:

(AL) = 10111101 = BDh, ответ: **BD**.

Задание 9. Записать содержимое (в десятичной системе) заданного регистра, полученное в результате выполнения следующих операторов ассемблера [операторы].

Указания. Для подготовки к выполнению задания рекомендуется использовать информацию об операторах условных переходов и об операторах цикла.

Пример. Записать содержимое регистра AX, полученное в результате выполнения операторов:

```
        XOR     AX, AX
        MOV     BX, 20
A1:     CMP     BX, 10h
```

```
        JBE     Next
        INC     AX
        DEC     BX
        JMP     A1
Next: . . . . . .
```

Решение. В записанном фрагменте программы первые два оператора выполняют инициализацию цикла ПОКА, а последующие операторы кодируют сам этот цикл. Условием повторения цикла является условие (BX)>10h. Оператор условного перехода JBE (перейти, если меньше или равно) реализует выход из цикла. Этот оператор используется для переходов после сравнения беззнаковых чисел, каковыми являются операнды оператора CMP — (BX) и 10h.

При первом выполнении оператора CMP значение (BX)=20, что явно больше, чем 16 (10h). В результате первого выполнения цикла (AX)=1, а (BX)=19. В результате четвертого выполнения — (AX)=4, (BX)=16. Это выполнение цикла является последним, так как следующее выполнение оператора JBE реализует выход из цикла на оператор с меткой Next.

Ответ: **4**.

Задание 10. Предлагается определить длину зарезервированного участка памяти или длину выводимого на экран сообщения.

Указания. Для подготовки к выполнению задания рекомендуется использовать информацию о псевдооператорах определения данных.

Пример. Указать длину (в байтах) участка памяти, резервируемого следующими операторами:

```
Mas1    DB   10  DUP (?)
Pere    DW  10, 20h, ‘Ab’
Text    ‘Hello’
```

Решение. ОП резервируется тремя псевдооператорами. Первый оператор резервирует 10 байтов, второй — 6 байтов (3 сло-

ва), третий — 5 байтов. Общее число резервируемых байтов памяти — **21** (ответ).

Остальные задания — на знание теории. Чтобы успешно выполнить эти задания, внимательно изучите все разделы пособия [1].

КОНТРОЛЬНАЯ РАБОТА № 2
РАЗРАБОТКА ПРОГРАММЫ НА АССЕМБЛЕРЕ

Введение

Целью выполнения данной работы является комплексная проверка навыков программирования на языке ассемблера.

Результаты работы представляются в виде совокупности следующих документов:

1) титульный лист;

2) дерево подпрограмм;

3) файловая структура программы;

4) блок-схемы алгоритмов процедур;

5) исходный файл (файлы) программы;

6) загрузочный модуль программы.

Пример титульного листа приведен в Приложении 4.

Пример дерева подпрограмм приведен в Приложении 3 (рис. 4).

Примеры файловой структуры программы приведены в [1] рис. 56 и 57.

Основным требованием к блок-схемам алгоритмов процедур является выполнение требований структурного программирования [приложение 3]. Примеры алгоритмов процедур приведены в [1] на рис. 54.

Основным требованием к исходным модулям (файлам) программы является наличие комментариев.

Дерево подпрограмм, файловая структура программы и блок-схемы процедур представляются в виде файлов, полученных с помощью текстового редактора Word. Остальные документы представляются в виде файлов с расширениями .asm и .com и помещаются в папку CONTR2.

Варианты заданий контрольной работы № 2

Вариант 1. По запросу программы пользователь вводит с клавиатуры последовательность целых двузначных положительных десятичных чисел, разделенных пробелами. Ввод последовательности заканчивается нажатием <Enter>.

Программа выводит на экран сумму этих чисел, представленную в десятичной и шестнадцатеричной системах счисления.

Вариант 2. По запросу программы пользователь вводит с клавиатуры целое положительное десятичное число N. По следующему запросу он вводит с клавиатуры N целых трехзначных положительных десятичных чисел, разделенных пробелами.

Программа выводит на экран сумму этих чисел, представленную в десятичной и «троичной» системах счисления.

Вариант 3. По запросу программы пользователь вводит с клавиатуры последовательность целых трехзначных положительных десятичных чисел, разделенных пробелами. Ввод последовательности заканчивается нажатием <Enter>.

Программа выводит наибольшее число из введенных, представленное в десятичной и «пятиричной» системах счисления.

Вариант 4. По запросу программы пользователь вводит с клавиатуры целое положительное десятичное число N. По следующему запросу он вводит с клавиатуры N целых трехзначных положительных десятичных чисел, разделенных пробелами.

Программа выводит наибольшее число из введенных, представленное в десятичной и шестнадцатеричной системах счисления.

Вариант 5. По запросу программы пользователь вводит с клавиатуры последовательность целых трехзначных положительных десятичных чисел, разделенных пробелами. Ввод последовательности заканчивается нажатием <Enter>.

Программа выводит наименьшее число из введенных, представленное в десятичной и восмеричной системах счисления.

Вариант 6. По запросу программы пользователь вводит с клавиатуры целое положительное десятичное число N. По следующему запросу он вводит с клавиатуры N целых трехзначных положительных десятичных чисел, разделенных пробелами.

Программа выводит наименьшее число из введенных, представленное в десятичной и «шестиричной» системах счисления.

Вариант 7. По запросу программы пользователь вводит с клавиатуры последовательность целых трехзначных положительных десятичных чисел, разделенных пробелами. Ввод последовательности заканчивается нажатием <Enter>.

Программа выводит последовательность этих же чисел, но записанных в обратном порядке и в шестнадцатеричной системе счисления.

Вариант 8. По запросу программы пользователь вводит с клавиатуры последовательность целых трехзначных положительных десятичных чисел, разделенных пробелами. Ввод последовательности заканчивается нажатием <Enter>.

Программа выводит эти же числа на экран в порядке убывания величины числа, причем в двоичной системе счисления.

Вариант 9. По запросу программы пользователь вводит с клавиатуры последовательность целых трехзначных положительных десятичных чисел, разделенных пробелами. Ввод последовательности заканчивается нажатием <Enter>.

Программа выводит эти же числа на экран в порядке возрастания величины числа, причем в «девятиричной» системе счисления.

Вариант 10. По запросу программы пользователь вводит с клавиатуры целое положительное десятичное число N. По следующему запросу он вводит с клавиатуры N целых трехзначных положительных десятичных чисел, разделенных пробелами.

Программа выводит эти же числа на экран в порядке возрастания величины числа, причем в «троичной» системе счисления.

Вариант 11. По запросу программы пользователь вводит с клавиатуры целое положительное десятичное число N. По следующему запросу он вводит с клавиатуры N целых трехзначных положительных десятичных чисел, разделенных пробелами.

Программа выводит последовательность этих же чисел, но записанных в обратном порядке и в восьмеричной системе счисления.

Вариант 12. По запросу программы пользователь вводит с клавиатуры целое положительное десятичное число N. По следующему запросу он вводит с клавиатуры N целых трехзначных положительных десятичных чисел, разделенных пробелами.

Программа выводит эти же числа на экран в порядке убывания величины числа, причем в шестнадцатеричной системе счисления.

Вариант 13. По запросу программы пользователь вводит с клавиатуры сообщение на русском языке, заканчивающееся символом «.» или «!».

Программа выводит на экран это же сообщение, записанное только заглавными буквами.

Вариант 14. По запросу программы пользователь вводит с клавиатуры сообщение на русском языке, заканчивающееся символом «.» или «?».

Программа выводит на экран это же сообщение, записанное только строчными (малыми) буквами.

Вариант 15. По запросу программы пользователь вводит с клавиатуры сообщение на английском языке, заканчивающееся символом «.» или «?».

Программа выводит на экран это же сообщение, записанное только заглавными буквами.

Вариант 16. По запросу программы пользователь вводит с клавиатуры сообщение на английском языке, заканчивающееся символом «.» или «!».

Программа выводит на экран это же сообщение, записанное только строчными (малыми) буквами.

Вариант 17. По запросу программы пользователь вводит с клавиатуры два целых четырехзначных положительных десятичных числа, разделенных знаком операции «+» или «–».

Программа выводит на экран результат операции в двух системах счисления — в десятичной и в двоичной (в дополнительном коде).

Вариант 18. По запросу программы пользователь вводит с клавиатуры два целых четырехзначных положительных десятичных числа, разделенных знаком операции « * »

Программа выводит на экран результат операции умножения.

Вариант 19. По запросу программы пользователь вводит с клавиатуры два целых четырехзначных положительных десятичных числа, разделенных знаком операции « / »

Программа выводит на экран результат операции деления (частное и остаток).

Вариант 20. По запросу программы пользователь вводит с клавиатуры два целых трехзначных положительных десятичных числа.

Программа выводит на экран сообщение о том, делится ли первое число на второе без остатка, а затем сообщение — делится ли без остатка второе число на первое.

Примечание 1

При вводе с клавиатуры десятичного числа следует учесть, что получение двоичного представления такого числа выполняется иначе по сравнению с шестнадцатеричным числом. При этом каждую очередную десятичную цифру следует умножить на вес позиции числа, а затем просуммировать результаты умножения. Например, при вводе 3-значного числа первая цифра умножается на сто, вторая — на десять, а третья цифра берется без изменения.

Примечание 2

Для перевода числа из десятичной системы счисления в любую другую воспользуйтесь алгоритмом вывода на экран десятичных и шестнадцатеричных чисел, тема в пособии 14. ВЫВОД НА ЭКРАН ДЕСЯТИЧНЫХ И ШЕСТНАДЦАТЕРИЧНЫХ ЧИСЕЛ.

4 РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Потапова Е. А. Информатика. Ассемблер для процессора Intel 8086 : учеб. пособие / Е. А. Потапова. — Томск : Эль Контент, 2013. — 166 с.

2. Одиноков В. В. Информатика. Ассемблер для процессора i8086 : учеб. пособие / В. В. Одиноков. — Томск : ТУСУР, 2000. — 93 с.

3. Информатика. Базовый курс : учебник для вузов / С. В. Симонович [и др.] ; ред. : С. В. Симонович. — 2-е изд. — СПб. : Питер, 2007. — 639 с.

4. Питер Абель. Ассемблер и программирование для IBM PC / Абель Питер. — Корона-Век, 2009. — 736 с.

5. Одиноков В. В. Программирование на ассемблере : учеб. пособие для вузов / В. В. Одиноков, В. П. Коцубинский. — М. : Горячая линия-Телеком, 2011

6. Острейковский В. А. Информатика: учебник для вузов / В. А. Острейковский. — М. : Высшая школа, 2001. — 512 с.

7. Юров В. И. Assembler : учебник для вузов / В. И. Юров. — СПб. : Питер, 2001. — 624 с..

ПРИЛОЖЕНИЕ 1
РАБОТА В СРЕДЕ MS-DOS

Язык управления *DOS*

DOS — дисковая операционная система. Этим названием обозначается не одна, а целая группа ОС, имеющих схожие пользовательские и программные интерфейсы. Примерами являются *MS-DOS — DOS* фирмы *Microsoft*, а также *Free DOS — DOS*, распространяемая свободно. Любая *DOS* — простая однопользовательская, однопрограммная операционная система. К ее достоинствам относятся весьма малый объем занимаемой памяти, а также большое число разработанных для нее прикладных программ.

Запуск DOS

Запуск операционной системы *DOS* зависит от ее типа. При этом *MS-DOS* версии 7.0 не является самостоятельной ОС. Поэтому ее запуск и последующее выполнение производится в среде *WINDOWS*. Способы запуска имитатора *MS-DOS* из *WINDOWS 2000* или *XP*:

1) <пуск> → <программы> → <командная строка>;

2) <пуск> → <выполнение> → <“*cmd*”;

3) запустить *DOS Navigator*, *FAR* или другую подобную утилиту, а затем воспользоваться командной строкой *MS-DOS* внизу экрана.

В отличие от *MS-DOS Free DOS* является самостоятельной ОС, выполняемой на «голой» аппаратуре. Поэтому для ее запуска требуется или отсутствие на ЭВМ другой ОС, или, что более удобно, — предварительный запуск из имеющейся на ЭВМ операционной системы (*WINDOWS* или *UNIX*) эмулятора аппаратуры.

Запуск *MS-DOS* производится или в самостоятельном режиме, или в среде операционной системы *Windows*. Для запуска *MS-DOS* в самостоятельном режиме требуется, чтобы на логическом системном диске (*c*:) находились файлы, образующие *MS-DOS*:

1) *Io.sys* — содержит подпрограммы ввода-вывода;
2) *Msdos.sys* — содержит ядро операционной системы;
3) *Command.com* — интерпретатор команд *MS-DOS*.

Эти три файла никогда нельзя корректировать. Следующие два файла, относящиеся к *MS-DOS*, можно корректировать:

Config.sys — перечень сведений об устройствах, имеющихся в ЭВМ, и как с этими устройствами работать. Запуск драйверов устройств производится из этого файла;

Autoexec.bat — перечень команд ОС, которые она должна выполнить сразу же после своей загрузки в ОП.

Запуск *MS-DOS* в самостоятельном режиме обычно производится при отсутствии на системном диске операционной системы *Windows*. В этом случае запуск *MS-DOS* производится автоматически после включения питания. При наличии *Windows* с помощью дополнительных действий также можно запустить *MS-DOS* в самостоятельном режиме, но этого не делают по той причине, что *Windows* позволяет запускать *MS-DOS* (а точнее, ее имитатор) более простыми способами.

Общие сведения о командах DOS

После запуска *DOS* любым из перечисленных выше способом на экране появляется ее приглашение, например следующего типа:

C:\SIMP>

Это приглашение означает, что в текущий момент времени *DOS* ожидает от вас команды и что она находится в точке файловой структуры — *C:\SIMP*, где *SIMP* — текущий каталог на логическом диске *C:* .

Строка экрана, в которой появилось приглашение, называется ***командной строкой***. В этой строке вы набираете команду:

< имя программы > [< параметры >] ,

где часть команды в квадратных скобках не обязательна.

Имя программы — имя файла, содержащего загрузочный модуль программы (расширение имени файла — *.com* или *.exe*), или имя командного файла (расширение — *.bat*). Расширение имени файла в команде может отсутствовать.

Набрав команду, вы нажимаете клавишу <*ENTER*>. В результате команда поступает в интерпретатор команд *DOS*, который поступает следующим образом. Во-первых, он определяет имя программы, которая должна быть выполнена — это последовательность символов, заканчивающаяся первым пробелом. Во-

вторых, ИК ищет адрес требуемой программы — определяет, на каком логическом диске находится загрузочный модуль программы, а также ее адрес на физическом диске. С учетом того, что в файловой структуре системы могут находиться несколько файлов с требуемым именем, необходимо четко представлять алгоритм поиска требуемого файла, по которому работает интерпретатор команд *DOS*:

1) во-первых, он просматривает свои внутренние таблицы. Если при этом имя файла найдено, то запускается соответствующая программа. Иначе — переход на шаг 2;

2) просматриваются все файлы текущего каталога;

3) поиск файла во всех каталогах, указанных в команде *PATH* файла *Autoexec.bat*. Например, команда *PATH c:\len; c:\auto* сообщает о том, что программные файлы следует искать в каталогах *LEN* и *AUTO* логического диска *C*: .

Если в своей команде вы не задали расширение имени файла, то в любом из трех перечисленных шагов приоритет поиска следующий:

1) ищется файл с расширением «*.com*»;

2) с расширением «*.exe*»;

3) с расширением «*.bat*».

Если ни один из трех перечисленных шагов поиска не увенчался успехом, на экран выводится: *Bad command or file name* (имя команды или файла указано неверно) и вновь выдается приглашение *DOS*.

Некоторые системные команды DOS

Естественно, что перечислить все команды *DOS* невозможно, так как имя любой программы может рассматриваться как команда. Речь может идти только о перечислении команд, требующих выполнения системных программ — утилит, лингвистических процессоров и драйверов. Соответствующая системная программа может находиться внутри *DOS* или существовать в виде отдельного *.com*- или *.exe*-файла. Команды, соответствующие первому типу программ, называются внутренними, а второму — внешними. Вот некоторые из системных команд.

1. **Задание текущего логического диска** (внутренняя команда). Для этого в ответ на приглашение *DOS* достаточно набрать требуемое имя логического диска, например:

C: или *D:*

2. **Задание текущего каталога** (внутренняя команда):

CD <имя каталога>

Например, в результате выполнения команды «*CD \SIMP\SET*» текущим каталогом станет дочерний каталог каталога *SIMP — SET*. Текущий логический диск при этом не меняется.

3. Вывод на экран содержимого каталога (внутренняя команда):

DIR [<имя лог. диска, или имя каталога, или имя файла>][/***p***][/***w***] ,

где квадратными скобками выделены необязательные параметры.

Если параметры-имена отсутствуют, то на экран выводится содержимое текущего каталога:

DIR

Если задано имя логического диска, то выводится содержимое корневого каталога на этом диске. Например, следующая команда выводит на экран содержимое корневого каталога на логическом диске *A:*

DIR A:

Если задано имя каталога, то на экран выводится его содержимое. Например, следующая команда выводит на экран содержимое каталога *SIMP*, являющегося дочерним каталогом по отношению к текущему каталогу:

DIR SIMP

Если задано имя файла, то на экран выводятся сведения только об этом файле. При задании имени файла разрешается вместо любой последовательности символов в имени (в том числе, и вместо расширения имени) задать символ «*». В этом случае на экран будут выведены сведения обо всех файлах, имеющих в своих именах последовательности символов, заданные в команде. Например, следующая команда выводит сведения обо всех файлах текущего каталога, имеющих расширение «*.exe*»:

*DIR *.exe*

Если информация в каталоге слишком велика, чтобы уместиться на одном экране, то используют параметр «/*p*». В этом

случае заполнение экрана приводит к приостановке вывода до нажатия вами любой клавиши.

Параметр «*/w*» используется для сжатия выводимой на экран информации за счет того, что для каждого файла выводится лишь имя, а атрибуты (размер, дата и время создания) опускаются. Допускается одновременное применение и параметра «*/p*», и параметра «*/w*».

4. Создание каталога:

MD <имя каталога>.

5. Уничтожение каталога:

RD < имя каталога >

6. **Копирование файла** (внутренняя команда):

COPY <имя файла1> <имя каталога или имя файла2>

Данная команда или создает копию файла с именем «имя файла 1» в заданном каталоге, или создает копию файла в прежнем каталоге, но с новым именем файла «имя файла 2». Например, команда

COPY abc.exe c:\simp

копирует файл *abc.exe*, расположенный в текущем каталоге текущего логического диска, в файл с таким же простым именем, но расположенный в каталоге *simp* на логическом диске *c:*.

В результате выполнения команды

COPY abc.exe 123.exe

текущий каталог содержит два файла с одинаковым содержанием, но с разными именами.

7. **Удаление файла** (внутренняя команда):

DEL <имя файла> ,

где <имя файла> — имя удаляемого файла.

8. **Переименование файла** (внутренняя команда):

REN <имя файла 1> <имя файла 2> ,

где <имя файла 1> — старое имя файла;

<имя файла 2> — новое имя файла.

9. Вывод содержимого текстового файла на экран:

TYPE <имя файла> ,

где <имя файла> — имя текстового файла в коде *ASCII*.

10. Создание или корректировка текстового файла:

EDIT <имя файла>

Командные файлы

Командный файл — файл, содержащий несколько команд *DOS*. Он создается пользователем как обычный текстовый файл, имеющий любое собственно имя, но обязательно расширение имени — *.bat*. При этом каждая команда набирается на отдельной строке экрана. Один командный файл играет особую роль — *Autoexec.bat*. Он запускается автоматически (то есть без нашего участия) *DOS* в самом начале ее работы. Обычно, среди прочих команд, этот файл содержит:

1) команду запуска подпрограммы *PATH*;

2) команды запуска драйверов клавиатуры и экрана;

3) команду запуска утилиты *DOS Navigator*, *FAR* или другой подобной утилиты.

Пример простого командного файла, выполняющего переход в заданный каталог *\katalog*, вывод текстового файла *1.txt*, содержащегося в этом каталоге, а затем уничтожение этого файла:

```
cd  \katalog
type  1.txt
del  1.txt
```

Если командный файл, содержащий перечисленные команды, имеет имя *2.bat*, то для запуска этого файла на выполнение достаточно использовать команду *DOS*:

```
c:\> 2.bat
```

ПРИЛОЖЕНИЕ 2
УТИЛИТА DOS NAVIGATOR

Представление на экране файловой структуры

Утилита *DOS Navigator* предназначена для того, чтобы предоставить пользователю *DOS* удобный интерфейс для общения с этой системой. Это обеспечивается, во-первых, наглядным выводом на экран информации о файловой структуре системы. Во-вторых, *DOS Navigator* существенно упрощает для пользователя ввод команд *DOS*.

Для того, чтобы запустить *DOS Navigator*, достаточно набрать команду *DOS — dn*. Часто эту команду включают в файл *Autoexec.bat*, и поэтому она выполняется автоматически. В любом случае в верхней части экрана появляются две серых панели, каждая из которых содержит перечень файлов, расположенный в одном из каталогов файловой структуры системы. При этом в заголовке каждой панели указаны имя логического диска и имя каталога. Ниже панелей располагается командная строка *DOS* с обычным ее приглашением и мерцающим курсором. В последней строке экрана находится список клавиш <*F1*> — <*F10*> с кратким обозначением их функций.

Каталоги, отображаемые на левой и правой панелях, могут совпадать или не совпадать, но в любом случае одна из панелей, называемая ***активной панелью***, отображает текущий каталог. Заголовок активной панели выделяется зеленым цветом. Кроме того, одно из имен файлов на активной панели выделено курсором-маркером. Смена активной панели производится нажатием клавиши <*Tab*>.

Перемещение курсора-маркера внутри активной панели производится с помощью клавиш управления курсором — вниз, вверх, влево, вправо. Нажатие клавиши <*End*> приводит к установке курсора-маркера на последнюю, а <*Home*> — на первую строку панели. Щелчком левой клавиши мыши можно установить курсор-маркер в любую позицию не только активной, но и соседней панели (происходит смена активной панели).

В общем случае панель содержит строки трех типов:

1) строку «**..**», обозначающую выход в «родительский каталог» данного каталога;

2) строки с именами подкаталогов данного каталога (выделяются белым цветом);

3) строки с именами файлов данного каталога. При этом цвет строки зависит от типа файла: серый для обычных текстовых файлов; светлозеленый для исполняемых файлов (расширение имени файла — *com*, *exe* или *bat*); яркозеленый для архивных файлов; голубой для скрытых файлов; розовый для текстовых файлов редактора *WORD*.

В последней строке панели, как правило, записано имя выделенного файла, его длина, дата и время создания или последней модификации. Еще ниже выводится общее число свободной памяти (в байтах) на логическом диске, к которому относится рассматриваемый на панели каталог.

Наличие у *DOS Navigator* своих команд для работы с каталогами и файлами существенно упрощает работу пользователя по управлению ОС. Рассмотрим основные из этих команд.

1. **Смена логического диска, соответствующего панели**, производится одновременным нажатием двух клавиш: а) для левой панели — <***Alt***> + <***F1***>; б) для правой панели — <***Alt***> + <***F2***>. На экране появится диалоговое окно-меню из имен логических дисков. После этого следует установить курсор-маркер на требуемое имя и нажать клавишу <*Enter*>.

2. **Переход на подкаталог текущего каталога** (смена текущего каталога + вывод на экран его содержимого) производится установкой на него курсора-маркера и нажатием <*Enter*>. Для возврата в родительский каталог требуется установить курсор-маркер на «**..**» и нажать <*Enter*>. Перемещаясь подобным образом вверх-вниз по файловой структуре логического диска, можно сделать текущим любой каталог на нем.

3. **Создание каталога**. Для этого достаточно установить в заголовке активной панели «родительский» каталог по отношению к вновь создаваемому каталогу, а затем нажать <***F7***>. На экране появится диалоговое окно с приглашением набрать имя нового каталога. В ответ следует набрать имя каталога прописными или строчными буквами и нажать <*Enter*>. В результате на активной панели появится имя нового каталога.

4. **Копирование файла**. Для этого требуется установить в заголовке одной из панелей «родительский» каталог по отношению к копируемому файлу, а в заголовке другой панели — «родительский» каталог по отношению к файлу-копии. Далее следует установить курсор-маркер на копируемый файл и нажать клавишу <***F5***>. На экране появится диалоговое окно с сообщением о готовности выполнить копирование. В ответ достаточно нажать клавишу <*Enter*>. (Для отмены этой или другой команды следует нажать <*Esc*>.) В результате на второй панели появится имя скопированного файла.

Для того чтобы создать копию файла в том же каталоге, что и исходный файл, необходимо обеспечить, чтобы старый и новый файлы имели разные имена. Для этого следует в диалоговом окне, появившемся после нажатия <*F5*>, набрать имя файла-копии, а уж затем нажать <*Enter*>.

Копирование каталога выполняется аналогично копированию обычного файла данных. При этом копируется все поддерево файловой структуры, начинающееся с данного каталога.

Копирование нескольких «дочерних» файлов (каталогов) текущего каталога можно ускорить, используя ***выделение группы файлов***. Для выделения файла необходимо установить на его имя курсор-маркер и нажать клавишу <***Ins***>. В результате имя файла высвечивается желтым цветом, и оно включается в группу. (Для исключения файла из группы необходимо повторить эти же два действия — установку курсора-маркера и нажатие <*Ins*>.) После того, как требуемая группа файлов выделена (в нижней строке панели информация об общем числе выделенных файлов и их общем объеме), ее можно скопировать последовательным нажатием <*F5*> и <*Enter*>.

5. **Перенос файла или каталога**. Для этого требуется установить курсор-маркер на имя переносимого файла (каталога) и нажать <*F6*>. На экране появится диалоговое окно с просьбой подтвердить намерение перенести файл. В ответ достаточно нажать <*Enter*> (для отмены — <*Esc*>).

6. **Уничтожение файла или каталога**. Для этого требуется установить курсор-маркер на имя уничтожаемого файла (каталога) и нажать <***F8***>. На экране появится диалоговое окно с прось-

бой подтвердить намерение удалить файл. В ответ достаточно нажать <*Enter*> (для отмены — <*Esc*>).

Выделив группу файлов (аналогично выделению при копировании), ее можно уничтожить нажатием <***F8***> и последующими нажатиями <*Enter*> в ответ на вопросы *DOS Navigator*.

7. **Переименование файла**. Для этого следует установить курсор-маркер на требуемый файл и нажать <***F6***>. В появившемся диалоговом окне следует набрать новое имя файла, а затем нажать <*Enter*>.

8. **Просмотр содержимого текстового файла** осуществляется установкой курсора-маркера на требуемый файл и последующим нажатием <***F3***>. Если во время просмотра нажать <***F8***>, то произойдет смена используемой кодировки русских букв с кодировки, предназначенной для *DOS*, на кодировку, предназначенную для *WINDOWS* (или наоборот).

Если во время просмотра вы нажмете <***F4***>, то файл будет просматриваться в шестнадцатеричном режиме. Такой режим особенно полезен для просмотра не текстовых, а двоичных файлов.

9. **Создание текстового файла**. Для создания нового файла необходимо одновременно нажать клавиши <***Shift***>+<***F4***>. В появившемся на экране окне необходимо набрать имя создаваемого файла и нажать <*Enter*>. В ответ на последующий вопрос также нажимается эта клавиша. Далее выполняется ввод с клавиатуры содержимого файла, по завершению которого следует нажать клавишу <***F2***>.

10. **Редактирование текстового файла**. Для работы с ранее созданным текстовым файлом необходимо установить курсор-маркер в открытом каталоге на нужный файл и нажать <***F4***>. Для того, чтобы сохранить этот файл, скопировав его из ОП на диск, необходимо нажать <***F2***>.

11. **Прекращение работы** *DOS Navigator* происходит в результате нажатия нескольких клавиш на клавиатуре и нарисованных на экране:

<***F10***> → [*File*] → [*Exit*] → [*Yes*]

Заметим, что в результате нажатия <*F10*> на экран выводится главное меню *DOS Navigator*, через которое можно осуществить доступ к различным функциям этой программы.

Ввод команд DOS

В отличие от рассмотренных выше операций с файлами, для выполнения которых *DOS Navigator* предоставляет пользователю свои команды, формат остальных команд *DOS* остается без изменений. При этом помощь *DOS Navigator* заключается в ускорении набора этих команд.

1. Крайний случай — пользователь набирает команду в командной строке *DOS* полностью вручную, не пользуясь помощью *DOS Navigator*. Такой метод используется для ввода коротких или редко используемых команд.

2. Второй метод — для заполнения командной строки используются имена файлов, высвечиваемые на панелях. Для того чтобы скопировать имя файла с панели в командную строку, достаточно установить курсор-маркер на требуемое имя файла и одновременно нажать две клавиши — <*Ctrl*>+<*Enter*>. При этом введенное имя файла можно редактировать точно так же, как и остальную часть командной строки.

3. Третий метод позволяет обойтись вообще без записи в командную строку. Установив курсор-маркер на имени исполняемого файла (расширение имени — *.com, .exe* или *.bat*), следует нажать <*Enter*>. Данный метод обычно применяется тогда, когда команда не имеет параметров.

4. Четвертый метод заключается в том, что команда *DOS* переписывается в командную строку из протокола команд. ***Протокол команд*** — список последних команд, сохраненный в *DOS Navigator*.

Поиск нужной команды в протоколе может быть выполнен двумя способами. В первом способе на экран выводится весь список (меню) команд. Для этого достаточно нажать комбинацию клавиш — <*Alt*>+<*F8*>. Установив курсор-маркер на требуемую команду в меню, следует нажать <*Enter*>.

Во втором способе список команд в протоколе просматривается последовательно, команда за командой. Для перехода к первой из них (самой новой) требуется нажать комбинацию <*Ctrl*>+<*E*>. При этом данная команда будет скопирована в командную строку, где она может быть скорректирована до нажатия <*Enter*>. Если данная команда не устраивает, то вместо нажатия <*Enter*> следует опять нажать <*Ctrl*>+<*E*>. В результате в ко-

мандной строке окажется следующая команда из протокола. Для передвижения по протоколу на одну команду назад следует нажать комбинацию клавиш <*Ctrl*>+<*X*>.

Если программа, запускаемая любым способом из *DOS Navigator*, выводит какие-то данные на экран, то чтение их будет невозможно из-за присутствия на экране панелей. Для того, чтобы убрать с экрана обе панели, требуется одновременно нажать <*Ctrl*>+<*O*>. Для восстановления панелей достаточно опять нажать <*Ctrl*>+<*O*>. Удаление (восстановление) только левой панели производится одновременным нажатием <*Ctrl*>+<*F1*>, а правой — <*Ctrl*>+<*F2*>.

ПРИЛОЖЕНИЕ 3
ПРОЕКТИРОВАНИЕ ПРОГРАММЫ

Введение

Разработка любой программы включает три этапа:

1) проектирование;

2) кодирование;

3) отладка.

Целью этапа ***проектирование*** является получение алгоритма решения задачи по переработке информации. Целью этапа ***кодирование*** является запись полученного алгоритма на выбранном языке (языках) программирования. ***Отладка*** программы включает ***тестирование*** (проверку правильности ее работы) и ***исправление ошибок***, найденных в результате тестирования.

Только для очень небольших программ этапы разработки программы выполняются строго последовательно во времени (рис. 1). На практике эти этапы выполняются ***параллельно*** (одновременно). То есть одновременно какая-то часть программы проектируется, другая ее часть кодируется, а третья — отлаживается. Естественно, что проектирование при этом несколько опережает этап кодирования, а кодирование — отладку (рис. 2).

Рис. 1 — Последовательное выполнение этапов разработки программы

Существующие технологии программирования различаются между собой реализацией указанных трех этапов. Главным из них является этап проектирования. При правильном его выполнении этапы кодирования и отладки выполняются достаточно просто. И наоборот, недостаточно хорошо выполненное проектирование программы всегда предшествует чрезмерно продолжительным этапам кодирования и отладки.

Рис. 2 — Параллельное выполнение этапов разработки программы

После завершения разработки программы начинается ее ***сопровождение***. Во время сопровождения решаются две основные задачи:

1) исправление выявляющихся в ходе эксплуатации ошибок;

2) корректировка программы, вызванная изменениями в решаемой ею задаче по переработке информации.

Результаты проектирования

В процессе проектирования программы обычно получают три вида документов: 1) системную схему; 2) дерево подпрограмм; 3) алгоритмы подпрограмм.

Системная схема — небольшой документ, показывающий взаимодействие между программой — с одной стороны, и устройствами ввода-вывода и файлами во внешней памяти — с другой. Иными словами, системная схема описывает кратко интерфейс программы. На рис. 3 приведен пример системной схемы для программы, выполняющей запросы своих пользователей по предоставлению сведений о сотрудниках организации.

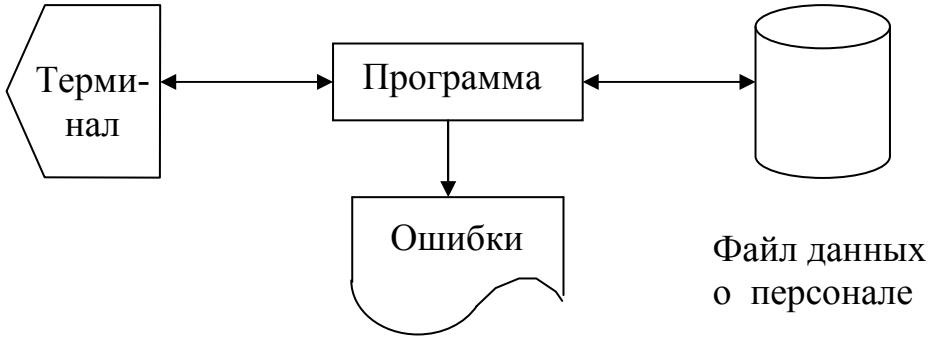


Рис. 3 — Пример системной схемы

Краткое описание интерфейсов программы, предоставляемое системной схемой, необходимо дополнить их подробным словесным описанием. Для этого необходимо описать структуру входных и выходных сообщений программы. ***Входные сообщения*** — сообщения, вводимые в программу пользователем с помощью клавиатуры терминала. ***Выходные сообщения*** — сообщения, выводимые программой пользователю (на экран его терминала). Кроме того, необходимо привести описание файлов, с которыми работает программа.

Два интерфейса не показываются на системной схеме: 1) между программой и ЦП; 2) между программой и ОП. Для описания этих интерфейсов используются ***параметры эффективности*** программы: 1) затраты времени ЦП на выполнение программы; 2) объем ОП, требуемый программе. В начале проектирования необходимо задать предельные значения этих двух параметров.

Дерево подпрограмм показывает — из каких подпрограмм состоит программа и как эти подпрограммы связаны по управлению. ***Связь по управлению*** — вызов (инициирование) одной подпрограммы другой. В состав подпрограмм входят процедуры, а также подпрограммы, вызываемые через программные прерывания. Причем системные подпрограммы, вызываемые через прерывания, в дерево подпрограмм обычно не включают.

Корнем дерева подпрограмм является процедура, называемая ***главной*** (***основной***) ***подпрограммой***. На следующем уровне дерева находятся подпрограммы, которые вызываются из основной подпрограммы и т.д. (рис. 4).

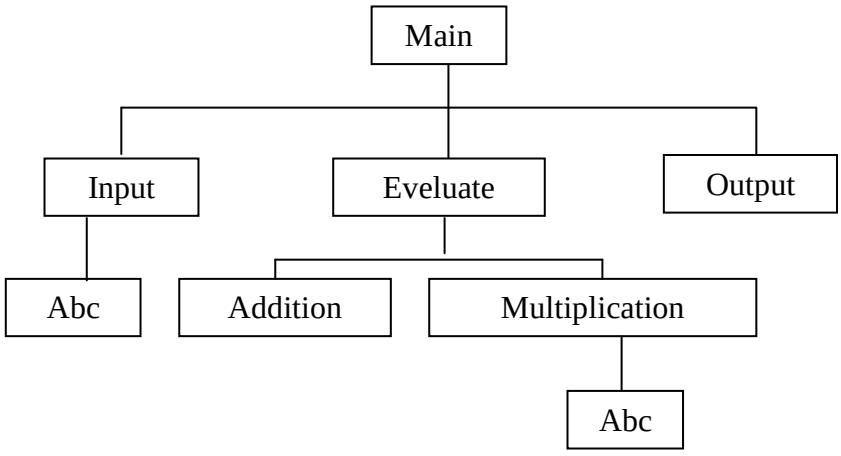


Рис. 4 — Пример дерева подпрограмм

Как видно из рис. 4, одна и та же подпрограмма (Abc) может быть многократно изображена на одном и том же дереве подпрограмм. Реально в памяти находится лишь одно тело подпрограммы, а ее повторения в дереве делаются для того, чтобы не затемнять рисунок пересечениями. (Дерево без пересечений воспринимается намного лучше.)

Проектирование любой подпрограммы включает два этапа: 1) определение интерфейсов; 2) получение алгоритма. На этапе ***определения интерфейсов*** подпрограмма рассматривается как «черный ящик» и для нее определяются границы (интерфейсы) с внешним миром. Основные подсистемы внешнего мира, с которыми взаимодействует подпрограмма:

1) подпрограмма (подпрограммы), вызывающая данную подпрограмму;

2) структуры данных (переменные, файлы), с которыми работает данная подпрограмма, но которые определены вне ее;

3) пользователь разрабатываемой программы.

Из перечисленных трех типов внешних подсистем только вызывающая подпрограмма является обязательной. Остальные подсистемы (пользователь и внешние структуры данных) при разработке конкретной подпрограммы могут отсутствовать. В этом случае интерфейс подпрограммы (процедуры) — перечень данных, которыми она обменивается с вызывающей ее программой. Сюда относятся входные и выходные параметры процедуры, а также области данных, на которые эти параметры указывают. Чем этот перечень меньше, тем лучше. (В идеальном случае подпрограмма вообще не имеет информационного обмена с вызывающими ее программами, но подобные подпрограммы могут выполнять лишь ограниченный набор функций.)

Результаты этапа выявления интерфейсов подпрограммы могут быть зафиксированы в системной схеме подобно тому, как это делалось для всей разрабатываемой программы. Но обычно это не делают, а ограничиваются описанием интерфейсов в виде вводных комментариев подпрограммы (п. 8.6). В любом случае точное описание интерфейсов подпрограммы должно предшествовать разработке ее алгоритма. Иными словами, не выполнив постановку задачи, нельзя приступать к ее решению.

Одним из приемов, упрощающих восприятие логики алгоритма, является последовательное применение регистров для информационного обмена между подпрограммами. Рекомендуется следующее использование регистров:

DL, DX — для передачи байта или слова в подпрограмму;

AL, AX — для возвращения байта или слова из подпрограммы;

BX:AX — для возвращения двойного слова;

DS:DX — передача и возвращение адресов;

CX — счетчик повторения и другие счетчики;

флаг CF — устанавливается при ошибке, при этом код ошибки возвращается в одном из регистров, например в AL или в AX.

На этапе ***разработки алгоритма*** выявляется совокупность шагов, обеспечивающих перевод входных данных подпрограммы в ее выходные данные. Для представления алгоритма может быть использован один из специально предназначенных для этого языков, например ***язык блок-схем***. На рис. 5 приведены основные символы (блоки) этого языка.

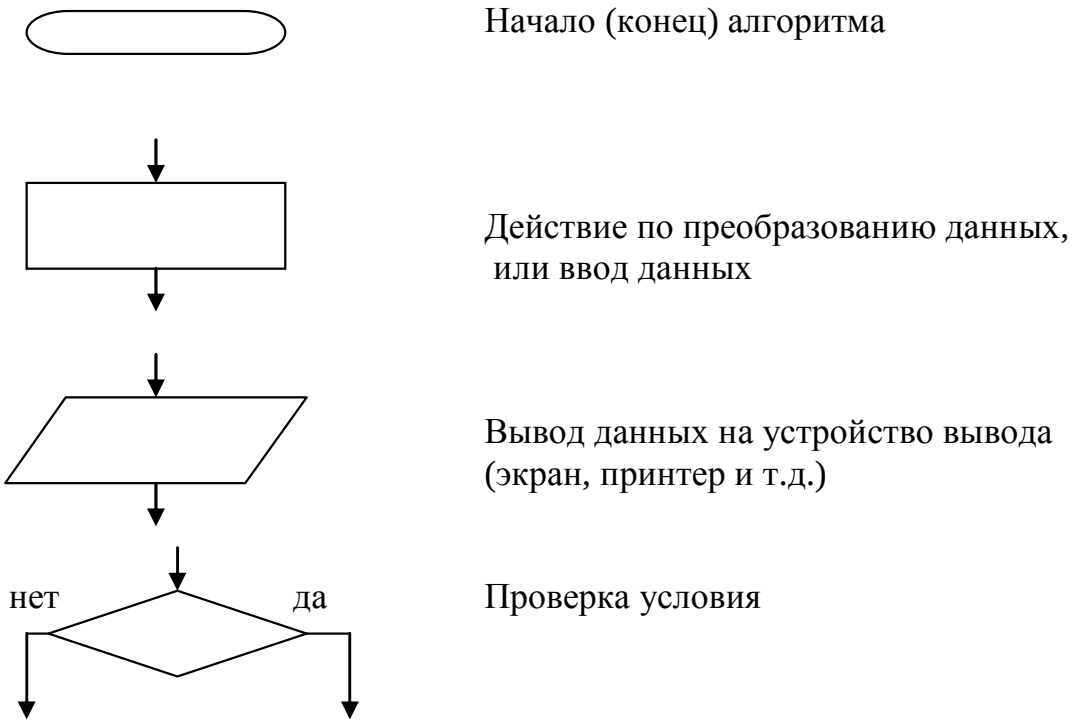


Рис. 5 — Основные элементы блок-схем

Основным требованием, предъявляемым к алгоритму, является его простота. Так как чем проще алгоритм программы, тем проще ее кодирование, отладка и сопровождение (эксплуатация). Для того, чтобы обеспечить простоту алгоритма, необходимо, чтобы он удовлетворял требованиям структурного программирования.

Структурное программирование

Структурное программирование — построение программ, алгоритмы которых включают только ***допустимые управляющие структуры***. Перечислим эти структуры:

1) ***цепочка*** — простая последовательность блоков типа «действие» и «вывод» (рис. 6, *а*);

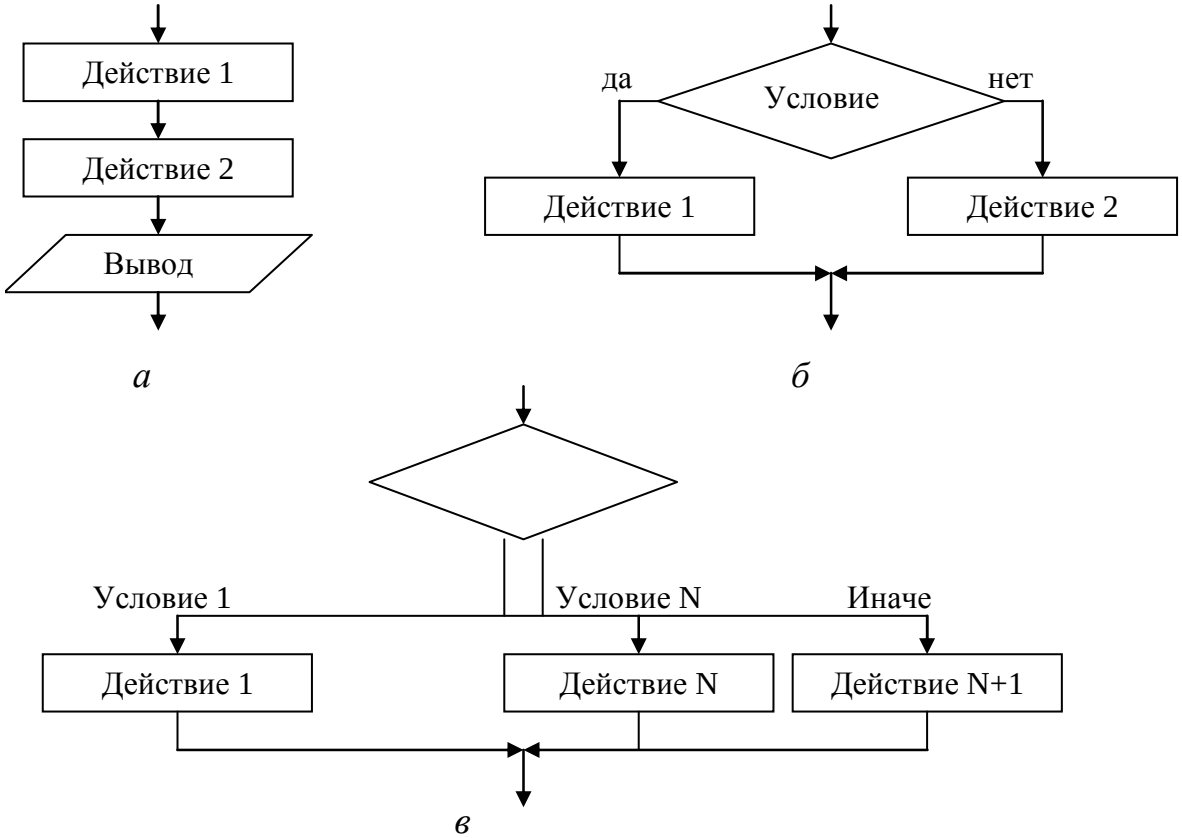


Рис. 6 — Ациклические управляющие структуры:
а — цепочка; *б* — двухальтернативный выбор;
в — многоальтернативный выбор

2) ***двухальтернативный выбор*** (структура **IF-THEN-ELSE**) — в зависимости от того, выполняется ли условие или нет, осуществляется действие 1 или действие 2 (рис. 6, *б*);

3) ***многоальтернативный выбор*** (структура **CASE**) — в зависимости от того, какое из взаимоисключающих условий 1...N выполняется, осуществляется действие 1...N. Если ни одно из этих условий не выполняется, осуществляется действие (N+1) (рис. 6, *в*);

4) ***цикл*** **ПОКА** (структура **DO-WHILE**) — повторение действия, образующего тело цикла, до тех пор, пока выполняется заданное условие (рис. 7, *а*). Для того чтобы цикл был конечным, тело цикла должно влиять на выполнение условия;

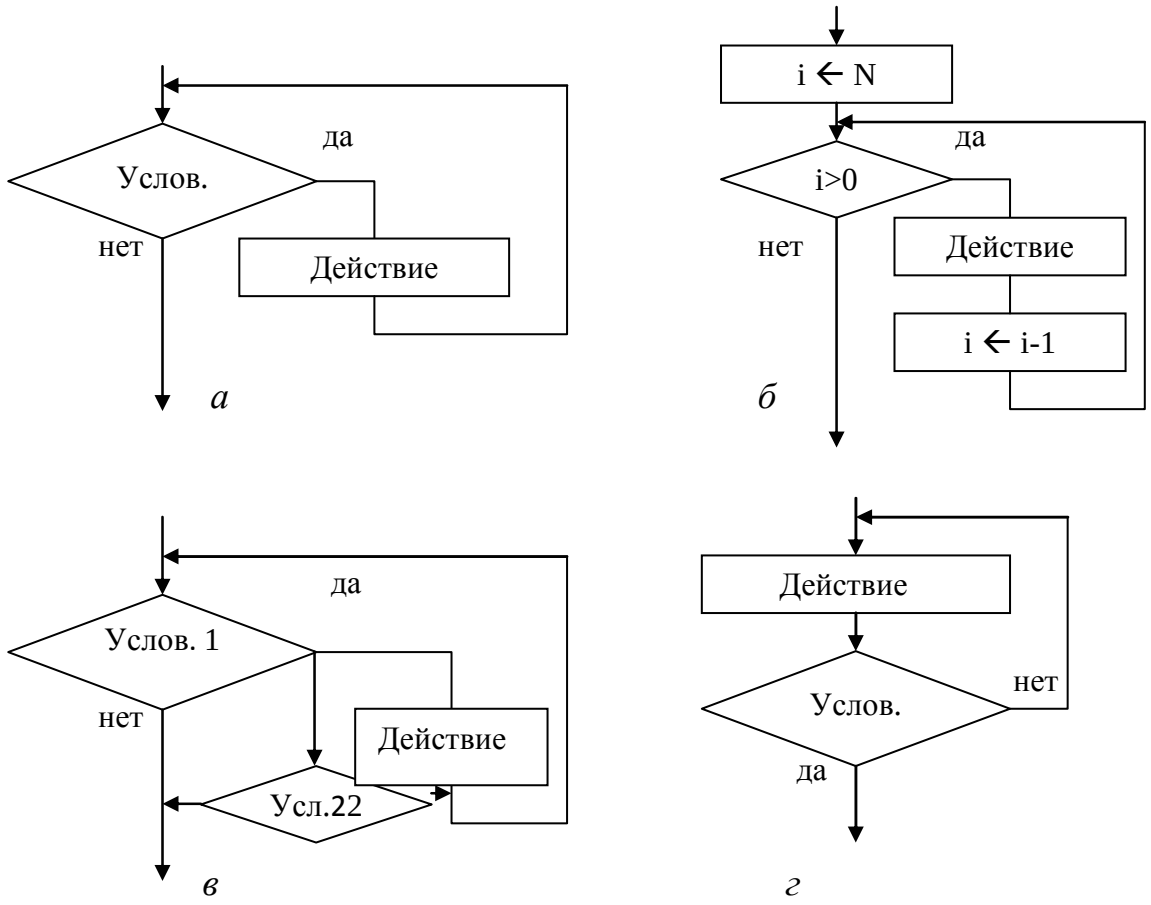


Рис. 7 — Циклические управляющие структуры:
а — цикл ПОКА; *б* — индексный цикл; *в* — цикл ПОКА с выходом;
г — цикл ДО

5) ***индексный цикл*** (частный случай цикла ПОКА) — повторение действия, образующего тело цикла, заданное число N раз (рис. 7, *б*);

6) ***цикл*** **ПОКА** с выходом — повторение действия, образующего тело цикла, до тех пор, пока выполняется заданное ус-

ловие 1. Если после очередного выполнения тела цикла будет выполнено дополнительное условие 2, осуществляется «досрочный» выход из цикла (рис. 7, *в*);

7) ***цикл* ДО** (структура **DO-UNTIL**) — повторение действия, образующего тело цикла, до тех пор, пока не выполнится заданное условие (рис. 7, *г*). В отличие от цикла ПОКА тело цикла ДО выполняется, по крайней мере, один раз.

Важной особенностью всех допустимых управляющих структур является то, что каждая из них имеет один вход и один выход. На основе любой из допустимых структур может быть получена составная управляющая структура. Для этого достаточно блок «действие», имеющийся в этой структуре, заменить на требуемую структуру. Нетрудно заметить, что полученная составная структура также имеет только один вход и только один выход и, следовательно, также отвечает требованиям структурного программирования. Используя подобную вложенность, можно записать алгоритм любого размера. На практике это реализуется путем вложенности алгоритмов подпрограмм.

Что касается размера алгоритма подпрограммы, то желательно, чтобы он был невелик (примерно 10 блоков). В любом случае данный алгоритм должен умещаться на одной странице, т.к. иначе восприятие логики алгоритма будет затруднено. Данное ограничение нетрудно обеспечить: любую допустимую управляющую структуру в алгоритме можно заменить одним действием, поручив ее выполнение «дочерней» подпрограмме.

В качестве примера рассмотрим алгоритм решения следующей простой задачи. Пусть программа должна ввести с клавиатуры два числа x и y, а затем вывести на экран сообщение о том, какое число больше. После этого программа должна попрощаться. Алгоритм программы приведен на рис. 8.

В этом алгоритме управляющей структурой верхнего уровня является цепочка, состоящая из трех элементов, средний из которых представляет собой вложенную управляющую структуру — двухальтернативный выбор. Одна из ветвей этого выбора также представляет собой вложенную структуру (двухальтернативный выбор).

В качестве второго примера рассмотрим блок-схему процедуры ввода шестнадцатеричной цифры (рис. 9). Данная процедура выполняет ввод символа с клавиатуры. Если этот символ явля-

ется шестнадцатеричной цифрой, процедура получает на основе кода ASCII значение цифры, помещает ее в один из регистров (для передачи в качестве выходного параметра), а также выводит цифру на экран. Если символ не является шестнадцатеричной цифрой, цикл повторяется.

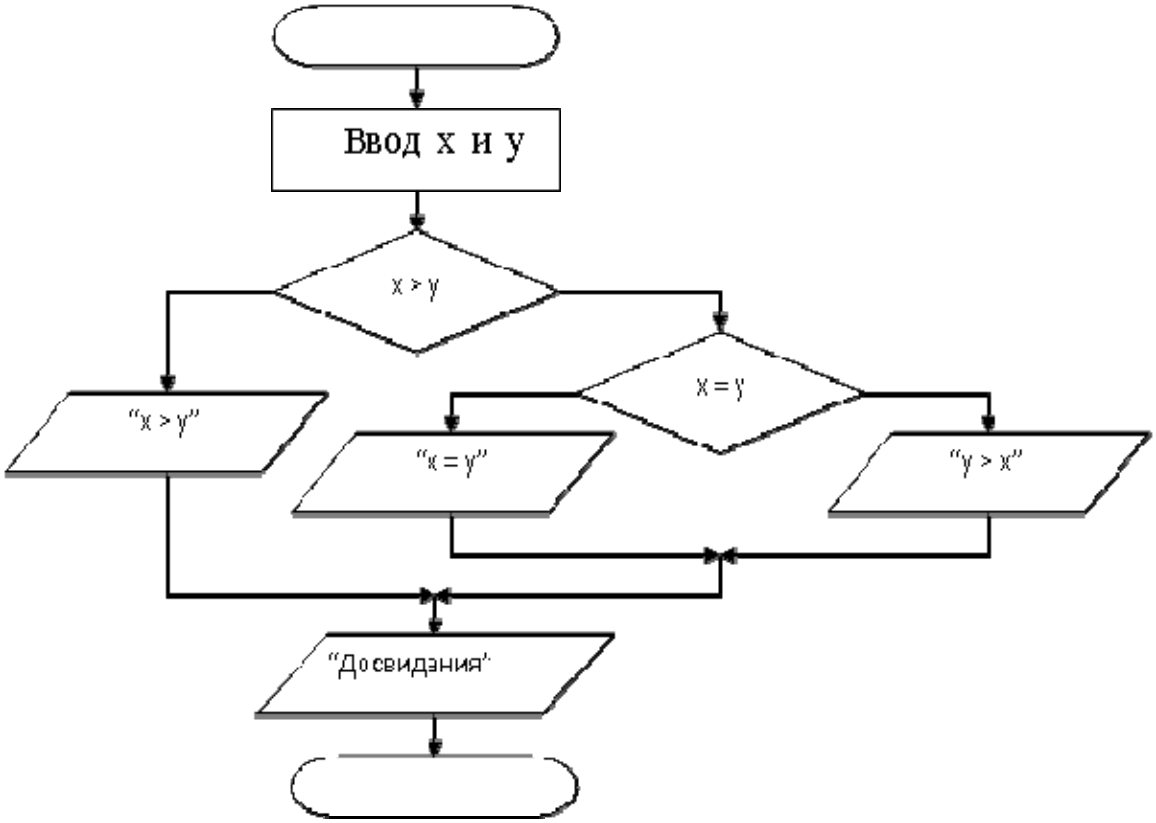
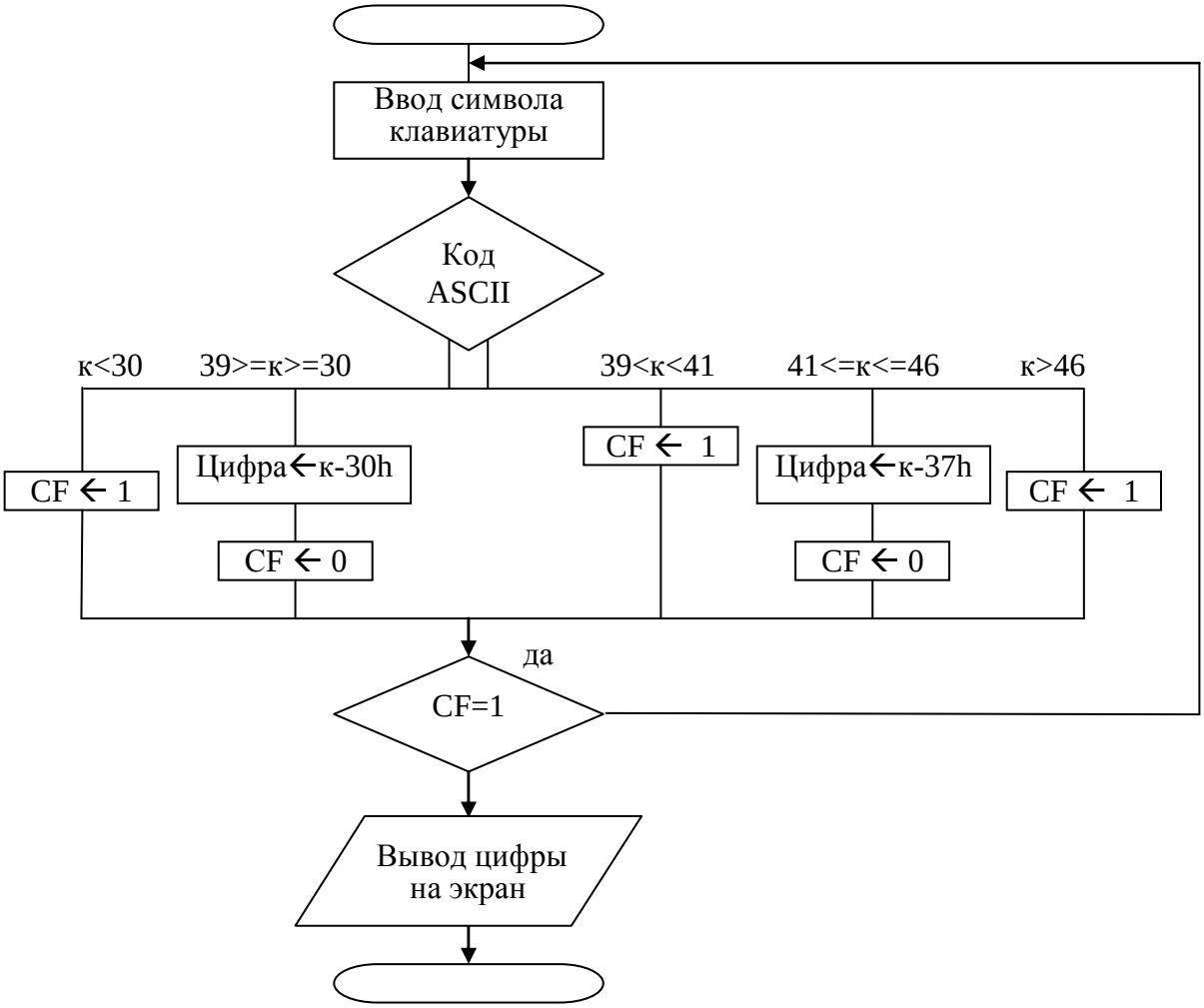


Рис. 8 — Алгоритм программы сравнения двух чисел

В этом алгоритме управляющей структурой верхнего уровня является цепочка из цикла ПОКА и этапа «Вывод цифры на экран». В качестве условия повторения цикла выступает равенство единице флага ошибки (в качестве флага ошибки обычно используют флаг переноса CF). Телом цикла является цепочка из этапа «Ввод символа с клавиатуры» и конструкции многоальтернативного выбора. В каждой ветви этого выбора устанавливается значение флага ошибки (CF=0 — введена цифра, CF=1 — цифры нет). Подобное использование флагов широко распространено на практике для обеспечения структурности алгоритмов программ.



CF — флаг ошибки (0 — ошибки нет, 1 — ошибка есть)

Рис. 9 — Алгоритм ввода одной шестнадцатеричной цифры

Методы проектирования программ

Применение любого метода проектирования начинается с получения системной схемы, т.к. не уточнив задачу, которую должна решать программа, дальнейшее ее проектирование не имеет смысла. Другие результаты проектирования (дерево подпрограмм и алгоритмы подпрограмм) могут быть получены различным образом во времени, в зависимости от используемого метода проектирования. Все эти методы можно разбить на три группы, различающиеся выбором объектов (подпрограмм), подлежащих проектированию:

1) проектирование «сверху-вниз»;
2) проектирование «снизу-вверх»;

3) проектирование «из центра».

В методе проектирования **сверху вниз** построение дерева подпрограмм начинается с корня дерева, т.е. с главной подпрограммы. Для этой подпрограммы разрабатывается алгоритм, отвечающий требованиям структурного программирования. После этого принимается решение о способе реализации каждого этапа алгоритма. При этом относительно простые этапы алгоритма должны кодироваться в рамках данной подпрограммы, а более сложные этапы должны быть реализованы путем вызова соответствующих подпрограмм. Имена этих подпрограмм заносятся в дерево подпрограмм. Далее выбирается любая нерассмотренная подпрограмма и для нее выполняется разработка алгоритма. Подобный рост дерева подпрограмм продолжается до тех пор, пока не останется подпрограмм, требующих алгоритмизации.

В методе проектирования **снизу вверх** первоначально проектируются подпрограммы, выполняющие сравнительно простые функции. (Разработка многих из этих подпрограмм сводится к выбору готовых подпрограмм, представленных в соответствующих библиотеках.) Далее разрабатываются подпрограммы, выполняющие более сложные функции и пользующиеся услугами ранее разработанных подпрограмм. Этот процесс продолжается до тех пор, пока очередная проектируемая подпрограмма не будет решать исходную задачу по переработке информации.

В методе проектирования **из центра** первоначально разрабатываются подпрограммы, предназначенные для решения задач по переработке информации, которые будут полезны (предположительно) для решения основной задачи. Но в отличие от метода «снизу-вверх» эти подпрограммы пользуются услугами таких подпрограмм, по крайней мере часть из которых на данный момент времени еще не разработана. Далее выполняется проектирование этих подпрограмм (движение «вниз»), а также проектирование вышестоящих подпрограмм (движение «вверх»). Движение «вверх» выполняется и заканчивается аналогично методу «снизу-вверх», а движение «вниз» — аналогично методу «сверху-вниз».

ПРИЛОЖЕНИЕ 4
ПРИМЕР ТИТУЛЬНОГО ЛИСТА

Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании (КСУП)

ОТЧЕТ

Лабораторная работа № 1

по дисциплине
«Информатика»

по учебно-методическому пособию Потаповой Е.А.

Выполнил студент:
специальности 220400.62
Иванов Иван Иванович

2013 г.