

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г.
ЧЕРНЫШЕВСКОГО»**

ЛАБОРАТОРНАЯ №2

Отчёт о практике

студента 2 курса 251 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Архипова Ивана Сергеевича

Проверено:

Старший преподаватель

Черноусова Е. М.

1 Задание 2.1

1.1 Текст задания


Первая цифра задана в AX, вторая цифра задана в BX. Написать программу, которая выводит в одну строку первую цифру, пробел, вторую цифру.

1.2 Текст программы

```
1  stak segment stack 'stack'
2  db 256 dup (?)
3  stak ends
4
5  data segment 'data'
6  Hello db 'Arkhipov Ivan, 251 group$'
7  data ends
8
9  code segment 'code'
10 assume CS:code,DS:data,SS:stak
11
12 print PROC FAR; Процедура вывода значения в DL на экран
13 push AX
14 mov AH, 02h
15 int 21h
16 pop AX
17 ret
18 print ENDP
19
20 print_nl PROC FAR; Процедура вывода новой строки
21 push AX
22 mov AH, 2h
23 mov DL, 0Ah
24 int 21h
25 pop AX
26 ret
27 print_nl ENDP
28
29 print_str PROC FAR; Процедура вывода строки на экран
30 push AX
31 mov AX,data
32 mov DS,AX
33 mov AH,09h
34 mov DX,offset Hello
```

```
35 int 21h
36 pop AX
37 ret
38 print_str ENDP
39
40 start:
41 call print_str
42 call print_nl
43
44 mov AX, 1h
45 mov BX, 3h
46
47 mov DL, AL
48 add DL, '0'
49 call print
50
51 mov DL, 0h
52 call print
53
54 mov DL, BL
55 add DL, '0'
56 call print
57
58 mov AX, 4C00h
59 int 21h
60 code ends
61
62 end start
```

1.3 Скриншот запуска программы



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

C:\>tasm.exe 1.asm
Turbo Assembler Version 2.0 Copyright (c) 1988, 1990 Borland International

Assembling file: 1.asm
Error messages: None
Warning messages: None
Passes: 1
Remaining memory: 491k

C:\>tlink.exe /x 1.obj
Turbo Link Version 3.0 Copyright (c) 1987, 1990 Borland International

C:\>1
1 3
C:\>_
```

Рис. 1 – Программа 1

1.4 Таблица трассировки

Таблица 1 – Трассировка программы №1

Шаг	Машинный код	Команда	Регистры									Флаги
			AX	BX	CX	DX	SP	DS	SS	CS	IP	CZSOPAID
1	0E	push cs	0000	0000	0000	0000	0100	489D	48AD	48BF	001F	00000010
2	E8EDFF	call 0010	0000	0000	0000	0000	00FE	489D	48AD	48BF	0020	00000010
3	50	push ax	0000	0000	0000	0000	00FC	489D	48AD	48BF	0010	00000010
4	B8BD48	mov ax, 48BD	0000	0000	0000	0000	00FA	489D	48AD	48BF	0011	00000010
5	8ED8	mov ds, ax	48BD	0000	0000	0000	00FA	489D	48AD	48BF	0014	00000010
6	B409	mov ah, 09	48BD	0000	0000	0000	00FA	48BD	48AD	48BF	0016	00000010
7	BA0000	mov dx, 0000	09BD	0000	0000	0000	00FA	48BD	48AD	48BF	0018	00000010
8	CD21	int 21	09BD	0000	0000	0000	00FA	48BD	48AD	48BF	001B	00000010
9	58	pop ax	09BD	0000	0000	0000	00FA	48BD	48AD	48BF	001D	00000010
10	CB	retf	0000	0000	0000	0000	00FC	48BD	48AD	48BF	001E	00000010
11	0E	push cs	0000	0000	0000	0000	0100	48BD	48AD	48BF	0023	00000010
12	E8E0FF	call 0007	0000	0000	0000	0000	00FE	48BD	48AD	48BF	0024	00000010
13	50	push ax	0000	0000	0000	0000	00FC	48BD	48AD	48BF	0007	00000010
14	B402	mov ah, 02	0000	0000	0000	0000	00FA	48BD	48AD	48BF	0008	00000010
15	B20A	mov dl, 0A	0200	0000	0000	0000	00FA	48BD	48AD	48BF	000A	00000010
16	CD21	int 21	0200	0000	0000	000A	00FA	48BD	48AD	48BF	000C	00000010
17	58	pop ax	020A	0000	0000	000A	00FA	48BD	48AD	48BF	000E	00000010

Шаг	Машинный код	Команда	Регистры									Флаги
			AX	BX	CX	DX	SP	DS	SS	CS	IP	CZSOPAID
18	CB	retf	0000	0000	0000	000A	00FC	48BD	48AD	48BF	000F	00000010
19	B80100	mov ax, 0001	0000	0000	0000	000A	0100	48BD	48AD	48BF	0027	00000010
20	BB0300	mov bx, 0003	0001	0000	0000	000A	0100	48BD	48AD	48BF	002A	00000010
21	8AD0	mov dl, al	0001	0003	0000	000A	0100	48BD	48AD	48BF	002D	00000010
22	80C230	add dl, 30	0001	0003	0000	0001	0100	48BD	48AD	48BF	002F	00000010
23	0E	push cs	0001	0003	0000	0031	0100	48BD	48AD	48BF	0032	00000010
24	E8CAFF	call 0000	0001	0003	0000	0031	00FE	48BD	48AD	48BF	0033	00000010
25	50	push ax	0001	0003	0000	0031	00FC	48BD	48AD	48BF	0000	00000010
26	B402	mov ah, 02	0001	0003	0000	0031	00FA	48BD	48AD	48BF	0001	00000010
27	CD21	int 21	0201	0003	0000	0031	00FA	48BD	48AD	48BF	0003	00000010
28	58	pop ax	0231	0003	0000	0031	00FA	48BD	48AD	48BF	0005	00000010
29	CB	retf	0001	0003	0000	0031	00FC	48BD	48AD	48BF	0006	00000010
30	B200	mov dl, 00	0001	0003	0000	0031	0100	48BD	48AD	48BF	0036	00000010
31	0E	push cs	0001	0003	0000	0000	0100	48BD	48AD	48BF	0038	00000010
32	E8C4FF	call 0000	0001	0003	0000	0000	00FE	48BD	48AD	48BF	0039	00000010
...
33	8AD3	mov dl, bl	0001	0003	0000	0000	0100	48BD	48AD	48BF	003C	00000010
34	80C230	add dl, 30	0001	0003	0000	0003	0100	48BD	48AD	48BF	003E	00000010
35	0E	push cs	0001	0003	0000	0033	0100	48BD	48AD	48BF	0041	00001010
36	E8BBFF	call 0000	0001	0003	0000	0033	00FE	48BD	48AD	48BF	0042	00001010
...
37	B8004C	ax, 4C00	0001	0003	0000	0033	0100	48BD	48AD	48BF	0045	00001010

2 Задание 2.2

2.1 Текст задания

Первая цифра задана в AX, вторая цифра задана в BX. Написать программу, которая выводит в одну строку первую цифру (AX), пробел, вторую цифру (BX). Далее совершает обмен значений регистров AX и BX и снова в новой строке на экране выводит в одну строку первую цифру (AX), пробел, вторую цифру (BX). Обмен совершить без использования дополнительной памяти, регистров. Структура программы должна обязательно содержать одну или более вспомогательных процедур.

2.2 Текст программы

```
1 stak segment stack 'stack'
2 db 256 dup (?)
3 stak ends
4
5 data segment 'data'
6 Hello db 'Arkhipov Ivan, 251 group$'
7 data ends
8
9 code segment 'code'
10 assume CS:code,DS:data,SS:stak
11
12 print PROC FAR; Процедура вывода значения в DL на экран
13 push AX
14 mov AH, 02h
15 int 21h
16 pop AX
17 ret
18 print ENDP
19
20 print_nl PROC FAR; Процедура вывода новой строки
21 push AX
22 mov AH, 2h
23 mov DL, 0Ah
24 int 21h
25 pop AX
26 ret
27 print_nl ENDP
28
```

```

29 print_str PROC FAR; Процедура вывода строки на экран
30 push AX
31 mov AX,data
32 mov DS,AX
33 mov AH,09h
34 mov DX,offset Hello
35 int 21h
36 pop AX
37 ret
38 print_str ENDP
39
40 printA_B PROC FAR; Процедура вывода значений регистров AX и BX
    на экран
41 mov DL, AL
42
43 add DL, '0'
44 call print
45
46 mov DL, 0h
47 call print
48
49 mov DL, BL
50 add DL, '0'
51 call print
52
53 ret
54 printA_B ENDP
55
56 start:
57 call print_str
58 call print_nl
59
60 mov AX, 9h
61 mov BX, 1h
62 call printA_B
63
64 push AX
65
66 mov AH, 2h
67 mov DL, 0Ah
68 int 21h
69

```

```

70 pop AX
71
72 xchg AX, BX
73 call printA_B
74
75 mov AX,4C00h
76 int 21h
77 code ends
78
79 end start

```

2.3 Скриншот запуска программы

```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

C:\>tasm.exe 2.asm
Turbo Assembler Version 2.0 Copyright (c) 1988, 1990 Borland International

Assembling file: 2.asm
Error messages: None
Warning messages: None
Passes: 1
Remaining memory: 491k

C:\>tlink.exe /x 2.obj
Turbo Link Version 3.0 Copyright (c) 1987, 1990 Borland International

C:\>2
Arkhipov Ivan, 251 group
9 1
1 9
C:\>

```

Рис. 2 – Программа 2

2.4 Таблица трассировки

Таблица 2 – Трассировка программы №2

Шаг	Машинный код	Команда	Регистры									Флаги
			AX	BX	CX	DX	SP	DS	SS	CS	IP	CZSOPAID
1	0E	push cs	0000	0000	0000	0000	0100	489D	48AD	48BF	0038	00000010
2	E8D4FF	call 0010	0000	0000	0000	0000	00FE	489D	48AD	48BF	0039	00000010
3	50	push ax	0000	0000	0000	0000	00FC	489D	48AD	48BF	0010	00000010
4	B8BD48	mov ax, 48BD	0000	0000	0000	0000	00FA	489D	48AD	48BF	0011	00000010

Шаг	Машинный код	Команда	Регистры									Флаги
			AX	BX	CX	DX	SP	DS	SS	CS	IP	CZSOPAID
5	8ED8	mov ds, ax	48BD	0000	0000	0000	00FA	489D	48AD	48BF	0014	00000010
6	B409	mov ah, 09	48BD	0000	0000	0000	00FA	48BD	48AD	48BF	0016	00000010
7	BA0000	mov dx, 0000	09BD	0000	0000	0000	00FA	48BD	48AD	48BF	0018	00000010
8	CD21	int 21	09BD	0000	0000	0000	00FA	48BD	48AD	48BF	001B	00000010
9	58	pop ax	09BD	0000	0000	0000	00FA	48BD	48AD	48BF	001D	00000010
10	CB	retf	0000	0000	0000	0000	00FC	48BD	48AD	48BF	001E	00000010
11	0E	push cs	0000	0000	0000	0000	0100	48BD	48AD	48BF	003C	00000010
12	E8C7FF	call 0007	0000	0000	0000	0000	00FE	48BD	48AD	48BF	003D	00000010
13	50	push ax	0000	0000	0000	0000	00FC	48BD	48AD	48BF	0007	00000010
14	B402	mov ah, 02	0000	0000	0000	0000	00FA	48BD	48AD	48BF	0008	00000010
15	B20A	mov dl, 0A	0200	0000	0000	0000	00FA	48BD	48AD	48BF	000A	00000010
16	CD21	int 21	0200	0000	0000	000A	00FA	48BD	48AD	48BF	000C	00000010
17	58	pop ax	020A	0000	0000	000A	00FA	48BD	48AD	48BF	000E	00000010
18	CB	retf	0000	0000	0000	000A	00FC	48BD	48AD	48BF	000F	00000010
19	B80900	mov ax, 0009	0000	0000	0000	000A	0100	48BD	48AD	48BF	0040	00000010
20	B80100	mov bx, 0001	0009	0000	0000	000A	0100	48BD	48AD	48BF	0043	00000010
21	0E	push cs	0009	0001	0000	000A	0100	48BD	48AD	48BF	0046	00000010
22	E8D5FF	call 001F	0009	0001	0000	000A	00FE	48BD	48AD	48BF	0047	00000010
23	8AD0	mov dl, al	0009	0001	0000	000A	00FC	48BD	48AD	48BF	001F	00000010
24	80C230	add dl, 30	0009	0001	0000	0009	00FC	48BD	48AD	48BF	0021	00000010
25	0E	push cs	0009	0001	0000	0039	00FC	48BD	48AD	48BF	0024	00001010
26	E8D8FF	call 0000	0009	0001	0000	0039	00FA	48BD	48AD	48BF	0025	00001010
27	50	push ax	0009	0001	0000	0039	00F8	48BD	48AD	48BF	0000	00001010
28	B402	mov ah, 02	0009	0001	0000	0039	00F6	48BD	48AD	48BF	0001	00001010
29	CD21	int 21	0209	0001	0000	0039	00F6	48BD	48AD	48BF	0003	00001010
30	58	pop ax	0239	0001	0000	0039	00F6	48BD	48AD	48BF	0005	00001010
31	CB	retf	0009	0001	0000	0039	00F8	48BD	48AD	48BF	0006	00001010
32	B200	mov dl, 00	0009	0001	0000	0039	00FC	48BD	48AD	48BF	0028	00001010
33	0E	push cs	0009	0001	0000	0000	00FC	48BD	48AD	48BF	002A	00001010
34	E8D2FF	call 0000	0009	0001	0000	0000	00FA	48BD	48AD	48BF	002B	00001010
...
35	8AD3	mov dl, bl	0009	0001	0000	0000	00FC	48BD	48AD	48BF	002E	00001010
36	80C230	add dl, 30	0009	0001	0000	0001	00FC	48BD	48AD	48BF	0030	00001010
37	0E	push cs	0009	0001	0000	0031	00FC	48BD	48AD	48BF	0033	00000010
38	E8C9FF	call 0000	0009	0001	0000	0031	00FA	48BD	48AD	48BF	0034	00000010
...
39	CB	retf	0009	0001	0000	0031	00FC	48BD	48AD	48BF	0037	00000010
40	50	push ax	0009	0001	0000	0031	0100	48BD	48AD	48BF	004A	00000010
41	B402	mov ah, 02	0009	0001	0000	0031	00FE	48BD	48AD	48BF	004B	00000010
42	B20A	mov dl, 0A	0209	0001	0000	0031	00FE	48BD	48AD	48BF	004D	00000010
43	CD21	int 21	0209	0001	0000	000A	00FE	48BD	48AD	48BF	004F	00000010
44	58	pop ax	020A	0001	0000	000A	00FE	48BD	48AD	48BF	0051	00000010
45	93	xchg bx, ax	0009	0001	0000	000A	0100	48BD	48AD	48BF	0052	00000010
46	0E	push cs	0001	0009	0000	000A	0100	48BD	48AD	48BF	0053	00000010

Шаг	Машинный код	Команда	Регистры									Флаги
			AX	BX	CX	DX	SP	DS	SS	CS	IP	CZSOPAID
47	E8C8FF	call 001F	0001	0009	0000	000A	00FE	48BD	48AD	48BF	0054	00000010
...
48	B8004C	mov ax, 4C00	0001	0009	0000	0039	0100	48BD	48AD	48BF	0057	00001010
49	CD21	int 21	4C00	0009	0000	0039	0100	48BD	48AD	48BF	005A	00001010

3 Контрольные вопросы

3.1 В какой регистр надо поместить код выводимого символа? Какой код Dos-функции используется для вывода отдельного символа на экран?

Код выводимого символа надо поместить в регистр DL. Для вывода отдельного символа на экран используется DOS-функция 02h.

3.2 Какая операция позволяет получить для цифры её код в кодовой таблице?

Чтобы получить код цифры в кодовой таблице, надо прибавить к ней 30h, шестнадцатеричный код символа 0. Это можно сделать с помощью команды add:

```
1 mov DL, 9
2 add DL, 30h; теперь DL содержит ASCII-код символа '9'
3 ; команда add DL, '0' сработала бы так же
```

3.3 Объясните назначение процедуры. Как определяются начало и конец процедуры?

Все современные программы разрабатываются по модульному принципу – программа обычно состоит из одной или нескольких небольших частей, называемых подпрограммами или процедурами, и одной главной программы, которая вызывает эти процедуры на выполнение, передавая им управление процессором.

Достоинством такого метода является возможность разработки программ значительно большего объема небольшими функционально законченными частями. Кроме того, эти подпрограммы можно использовать в других программах, не прибегая к переписыванию частей программного кода. В довершение ко всему, так как размер сегмента не может превышать 64 КБ, то при разработке программ с объемом кода более 64 КБ, просто не обойтись без модульного принципа.

Описание процедуры имеет следующий синтаксис:

```
1 <имя_процедуры> PROC <параметр>
2 <тело_процедуры>
3 RET ; Возврат из подпрограммы в точку вызова
4 <имя_процедуры> ENDP
```

Директива PROC указывает на начало процедуры, а ENDP на её конец.

3.4 Ваша программа состоит из главной процедуры и процедур-подпрограмм. Каким может быть взаимное расположение главной процедуры и подпрограмм?

В общем случае, размещать подпрограмму в теле программы можно где угодно, но при этом следует помнить, что сама по себе подпрограмма выполняться не должна, а должна выполняться лишь при обращении к ней. Поэтому подпрограммы принято размещать либо в конце сегмента кода, после команд завершения программы, либо в самом начале сегмента кода, перед точкой входа в программу. В больших программах подпрограммы нередко размещают в отдельном кодовом сегменте.

Итак, подпрограммы могут располагаться относительно главной процедуры следующим образом:

1 .code	1 .code	1 .code1
2 start:	2 proc1 PROC	2 proc1 PROC FAR
3 ...	NEAR	3 ...
4 mov ax, 4C00h	3 ...	4 proc1 ENDP
5 int 21h	4 proc1 ENDP	5 .code2
6	5 start:	6 start:
7 proc1 PROC	6 ...	7 ...
NEAR	7 mov ax, 4C00h	8 mov ax, 4C00h
8 ...	8 int 21h	9 int 21h
9 proc1 ENDP	9 end start	10 end start
10 end start		

3.5 Как процессор использует стек при работе с любой процедурой?

При вызове процедуры в стеке сохраняется адрес возврата в вызывающую программу:

- при вызове ближней процедуры — слово, содержащее смещение точки возврата относительно текущего кодового сегмента;
- при вызове дальней процедуры — слово, содержащее адрес сегмента, в котором расположена точка возврата, и слово, содержащее смещение точки возврата в этом сегменте.

Внутрисегментный вызов NEAR CALL указывает новое значение регистра IP и сохраняет в стеке адрес возврата, т.е. IP команды, следующей за командой CALL. Межсегментный вызов FAR CALL задает новые значения сегмента CS и смещения

IP для дальнейшего выполнения программы и сохраняет в стеке как регистр IP, так и регистр CS.

Внутрисегментный возврат извлекает из стека одно слово и помещает его в регистр IP, а межсегментный возврат извлекает из стека два слова, помещая слова из меньшего адреса в регистр IP, а слово из большего адреса — в регистр CS.

3.6 С помощью какой команды вызывается процедура? Как меняется значение регистра SP после вызова процедуры? Приведите пример из вашей таблицы трассировки.

Для вызова процедуры используется команда `call`.

Для процедур типа NEAR в стек помещается 2 байта в виде значения регистра IP, и SP уменьшается на 2 байта, чтобы указывать на новую вершину стека.

Для типа FAR же в стек помещаются 4 байта: сначала значения регистра CS, а затем IP, и SP, аналогично, уменьшается на 4 байта.

Приведём пример из таблицы трассировки второй программы:

Таблица 3 – Часть трассировки второй программы

Шаг	Машинный код	Команда	Регистры									Флаги	
			AX	BX	CX	DX	SP	DS	SS	CS	IP	CZ	SOPAID
...
20	B80100	<code>mov bx, 0001</code>	0009	0000	0000	000A	0100	48BD	48AD	48BF	0043	00000010	
21	0E	<code>push cs</code>	0009	0001	0000	000A	0100	48BD	48AD	48BF	0046	00000010	
22	E8D5FF	<code>call 001F</code>	0009	0001	0000	000A	00FE	48BD	48AD	48BF	0047	00000010	
23	8AD0	<code>mov dl, al</code>	0009	0001	0000	000A	00FC	48BD	48AD	48BF	001F	00000010	
...

Командой `push cs` в стек было добавлено значение регистра CS, которое занимает 2 байта в памяти, поэтому SP уменьшился на те же 2 байта. Команда `call 001F` добавила в стек значение регистра IP, которое также занимает 2 байта в памяти, из-за чего SP снова уменьшился на 2 байта.

Итого, при вызове процедуры типа FAR SP уменьшился на 4 байта.

3.7 После какой команды процедуры из стека извлекается адрес возврата?

Адрес возврата извлекается из стека после команды процедуры `ret`.

Внутрисегментный возврат извлекает из стека одно слово и помещает его в регистр IP, а межсегментный возврат извлекает из стека два слова, помещая слова из меньшего адреса в регистр IP, а слово из большего адреса — в регистр CS.