

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования

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

ЛАБОРАТОРНАЯ РАБОТА №2

Отчёт о практике

студента 2 курса 251 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Тюменцева Радомира Александровича

Проверено:

Старший преподаватель

Е. М. Черноусова

Саратов 2025

1 Задание 2

Сначала программы должны печатать фамилию, имя и номер группы студента и переходить на новую строку. Затем аналогично рассмотренному упражнению выполните следующие задания:

1.1 Задание 2.1

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

1.2 Задание 2.2

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.**

1.3 Тексты программ на языке ассемблера с комментариями

```

1 .model tiny
2 .code
3 org 100h
4
5 start:
6     ; Вывод фамилии, имени и номера группы
7     mov DX, offset my_name
8     call out_string
9     call new_line
10
11     mov AX, 1      ; Занесение первой цифры в регистр AX
12     mov BX, 2      ; Занесение второй цифры в регистр BX
13     ; Перевод цифр в коды соответствующих символов ASCII с помощью команды add
14     add AX, 30h
15     add BX, 30h
16
17     ; Вывод первой цифры
18     mov DX, AX
19     call out_char
20
21     ; Вывод пробела
22     call out_space
23
24     ; Вывод второй цифры
25     mov DX, BX
26     call out_char
27
28     ; Завершение программы
29     mov AX, 4C00h
30     int 21h
31
32     ; Процедура вывода строки
33     out_string proc
34         mov AH, 09h
35         int 21h
36         ret
37     out_string endp
38
39     ; Процедура вывода символа
40     out_char proc
41         mov AH, 02h
42         int 21h
43         ret
44     out_char endp
45
46     ; Процедура вывода пробела
47     out_space proc
48         mov DL, 00h ; Код пробела в ASCII
49         mov AH, 02h
50         int 21h
51         ret
52     out_space endp
53

```

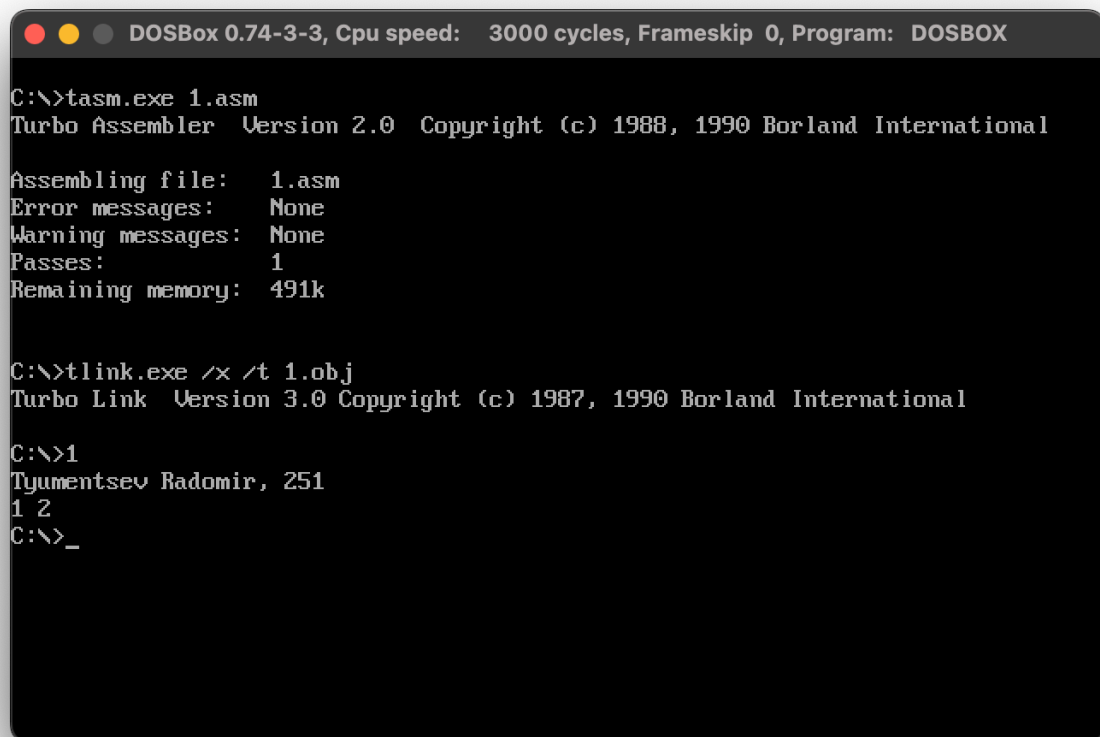
Текст программы 1

```

1 .model tiny
2 .code
3 org 100h
4
5 start:
6 ; Вывод фамилии, имени и номера группы
7 mov DX, offset my_name
8 call out_string
9 call new_line
10
11 mov AX, 1; Занесение первой цифры в регистр AX
12 mov BX, 2; Занесение второй цифры в регистр BX
13 ; Перевод цифр в коды соответствующих символов ASCII с помощью команды add
14 add AX, 30h
15 add BX, 30h
16
17 ; Сохранение значения регистра AX в стек
18 ; так как затем в него будут записываться номера функций DOS
19 push AX
20
21 ; Вывод первой цифры
22 mov DX, AX
23 call out_char
24
25 ; Вывод пробела
26 call out_space
27
28 ; Вывод второй цифры
29 mov DX, BX
30 call out_char
31
32 pop AX; Восстановление значения регистра AX из стека
33
34 xchg AX, BX; Обмен значениями регистров AX и BX
35
36 call new_line; Переход на новую строку
37
38 ; Вывод первой цифры
39 mov DX, AX
40 call out_char
41
42 ; Вывод пробела
43 call out_space
44
45 ; Вывод второй цифры
46 mov DX, BX
47 call out_char
48
49 ; Завершение программы
50 mov AX, 4C00h
51 int 21h
52
53 ; Промедление вывода строки

```

1.4 Скриншоты запуска программ



The screenshot shows a DOSBox window with a dark background and white text. The title bar at the top reads "DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX". The terminal content shows the following commands and output:

```
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 /t 1.obj
Turbo Link Version 3.0 Copyright (c) 1987, 1990 Borland International

C:\>1
Tyumentsev Radomir, 251
1 2
C:\>_
```

```

DOSBox 0.74-3-3, 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 /t 2.obj
Turbo Link Version 3.0 Copyright (c) 1987, 1990 Borland International

C:\>2
Tyumentsev Radomir, 251
1 2
2 1
C:\>_

```

Запуск программы 2

1.5 Таблицы трассировки программ

Таблица 1 – Таблица трассировки программы 1

Шаг	Машинный код	Команда	Регистры									Флаги
			AX	BX	CX	DX	SP	DS	SS	CS	IP	
1	BA4001	mov dx, 0140	0000	0000	0000	0000	FFFE	489D	489D	489D	0100	00000010
2	E82100	call 0127	0000	0000	0000	0140	FFFE	489D	489D	489D	0103	00000010
3	B409	mov ah, 09	0000	0000	0000	0000	FFFC	489D	489D	489D	0127	00000010
4	CD21	int 21	0900	0000	0000	0000	FFFC	489D	489D	489D	0129	00000010
5	C3	ret	0900	0000	0000	0000	FFFC	489D	489D	489D	012B	00000010
6	E82F00	call 0138	0900	0000	0000	0000	FFFE	489D	489D	489D	0106	00000010
7	BA5801	mov dx, 0158	0900	0000	0000	0000	FFFC	489D	489D	489D	0138	00000010
8	B409	mov ah, 09	0900	0000	0000	0158	FFFC	489D	489D	489D	013B	00000010
9	CD21	int 21	0900	0000	0000	0158	FFFC	489D	489D	489D	013D	00000010
10	C3	ret	0900	0000	0000	0158	FFFC	489D	489D	489D	013F	00000010
11	B80100	mov ax, 0001	0900	0000	0000	0158	FFFE	489D	489D	489D	0109	00000010
12	BB0200	mov bx, 0002	0001	0000	0000	0158	FFFE	489D	489D	489D	010C	00000010
13	053000	add ax, 0030	0001	0002	0000	0158	FFFE	489D	489D	489D	010F	00000010
14	83C330	add bx, 0030	0031	0002	0000	0158	FFFE	489D	489D	489D	0112	00000010
15	8BD0	mov dx, ax	0031	0032	0000	0158	FFFE	489D	489D	489D	0115	00000010
16	E81200	call 012C	0031	0032	0000	0031	FFFE	489D	489D	489D	0117	00000010
17	B402	mov ah, 02	0031	0032	0000	0031	FFFC	489D	489D	489D	012C	00000010

Шаг	Машинный код	Команда	Регистры									Флаги
			AX	BX	CX	DX	SP	DS	SS	CS	IP	CZSOPAID
18	CD21	int 21	0231	0032	0000	0031	FFFC	489D	489D	489D	012E	00000010
19	C3	ret	0231	0032	0000	0031	FFFC	489D	489D	489D	0130	00000010
20	E81400	call 0131	0231	0032	0000	0031	FFFE	489D	489D	489D	011A	00000010
21	B200	mov dl, 00	0231	0032	0000	0031	FFFC	489D	489D	489D	0131	00000010
22	B402	mov ah, 02	0231	0032	0000	0000	FFFC	489D	489D	489D	0133	00000010
23	CD21	int 21	0231	0032	0000	0000	FFFC	489D	489D	489D	0135	00000010
24	C3	ret	0200	0032	0000	0000	FFFC	489D	489D	489D	0137	00000010
25	8BD3	mov dx, bx	0200	0032	0000	0000	FFFE	489D	489D	489D	011D	00000010
26	E80A00	call 012C	0200	0032	0000	0032	FFFE	489D	489D	489D	011F	00000010
27	B402	mov ah, 02	0200	0032	0000	0032	FFFC	489D	489D	489D	012C	00000010
28	CD21	int 21	0200	0032	0000	0032	FFFC	489D	489D	489D	012E	00000010
29	C3	ret	0232	0032	0000	0032	FFFC	489D	489D	489D	0130	00000010
30	B8004C	mov ax, 4C00	0232	0032	0000	0032	FFFE	489D	489D	489D	0122	00000010
31	CD21	int 21	4C00	0032	0000	0032	FFFE	489D	489D	489D	0125	00000010

Таблица 2 – Таблица трассировки программы 2

Шаг	Машинный код	Команда	Регистры									Флаги
			AX	BX	CX	DX	SP	DS	SS	CS	IP	CZSOPAID
1	BA5301	mov dx, 0153	0000	0000	0000	0000	FFFE	489D	489D	489D	0100	00000010
2	E83400	call 013A	0000	0000	0000	0153	FFFE	489D	489D	489D	0103	00000010
3	B409	mov ah, 09	0000	0000	0000	0000	FFFC	489D	489D	489D	013A	00000010
4	CD21	int 21	0900	0000	0000	0000	FFFC	489D	489D	489D	013C	00000010
5	C3	ret	0900	0000	0000	0000	FFFC	489D	489D	489D	013E	00000010
6	E84200	call 014B	0900	0000	0000	0000	FFFE	489D	489D	489D	0106	00000010
7	BA6B01	mov dx, 016B	0900	0000	0000	0000	FFFC	489D	489D	489D	014B	00000010
8	B409	mov ah, 09	0900	0000	0000	016B	FFFC	489D	489D	489D	014E	00000010
9	CD21	int 21	0900	0000	0000	016B	FFFC	489D	489D	489D	0150	00000010
10	C3	ret	0900	0000	0000	016B	FFFC	489D	489D	489D	0152	00000010
11	B80100	mov ax, 0001	0900	0000	0000	016B	FFFE	489D	489D	489D	0109	00000010
12	BB0200	mov bx, 0002	0001	0000	0000	016B	FFFE	489D	489D	489D	010C	00000010
13	053000	add ax, 0030	0001	0002	0000	016B	FFFE	489D	489D	489D	010F	00000010
14	83C330	add bx, 0030	0031	0002	0000	016B	FFFE	489D	489D	489D	0112	00000010
15	50	push ax	0031	0032	0000	016B	FFFE	489D	489D	489D	0115	00000010
16	8BD0	mov dx, ax	0031	0032	0000	016B	FFFC	489D	489D	489D	0116	00000010
17	E82400	call 013F	0031	0032	0000	0031	FFFC	489D	489D	489D	0118	00000010
18	B402	mov ah, 02	0031	0032	0000	0031	FFFA	489D	489D	489D	013F	00000010
19	CD21	int 21	0231	0032	0000	0031	FFFA	489D	489D	489D	0141	00000010
20	C3	ret	0231	0032	0000	0031	FFFA	489D	489D	489D	0143	00000010
21	E82600	call 0144	0231	0032	0000	0031	FFFC	489D	489D	489D	011B	00000010
22	B200	mov dl, 00	0231	0032	0000	0031	FFFA	489D	489D	489D	0144	00000010
23	B402	mov ah, 02	0231	0032	0000	0000	FFFA	489D	489D	489D	0146	00000010
24	CD21	int 21	0231	0032	0000	0000	FFFA	489D	489D	489D	0148	00000010
25	C3	ret	0200	0032	0000	0000	FFFA	489D	489D	489D	014A	00000010
26	8BD3	mov dx, bx	0200	0032	0000	0000	FFFC	489D	489D	489D	011E	00000010
27	E81C00	call 013F	0200	0032	0000	0032	FFFC	489D	489D	489D	0120	00000010

Шаг	Машинный код	Команда	Регистры									Флаги
			AX	BX	CX	DX	SP	DS	SS	CS	IP	CZSOPAID
28	B402	mov ah, 02	0200	0032	0000	0032	FFFA	489D	489D	489D	013F	00000010
29	CD21	int 21	0200	0032	0000	0032	FFFA	489D	489D	489D	0141	00000010
30	C3	ret	0232	0032	0000	0032	FFFA	489D	489D	489D	0143	00000010
31	58	pop ax	0232	0032	0000	0032	FFFC	489D	489D	489D	0123	00000010
32	93	xchg bx, ax	0031	0032	0000	0032	FFFE	489D	489D	489D	0124	00000010
33	E82300	call 014B	0032	0031	0000	0032	FFFE	489D	489D	489D	0125	00000010
34	BA6B01	mov dx, 016B	0032	0031	0000	0032	FFFC	489D	489D	489D	014B	00000010
35	B409	mov ah, 09	0032	0031	0000	016B	FFFC	489D	489D	489D	014E	00000010
36	CD21	int 21	0932	0031	0000	016B	FFFC	489D	489D	489D	0150	00000010
37	C3	ret	0932	0031	0000	016B	FFFC	489D	489D	489D	0152	00000010
38	8BD0	mov dx, ax	0932	0031	0000	016B	FFFE	489D	489D	489D	0128	00000010
39	E81200	call 013F	0932	0031	0000	0932	FFFE	489D	489D	489D	012A	00000010
40	B402	mov ah, 02	0932	0031	0000	0932	FFFC	489D	489D	489D	013F	00000010
41	CD21	int 21	0232	0031	0000	0932	FFFC	489D	489D	489D	0141	00000010
42	C3	ret	0232	0031	0000	0932	FFFC	489D	489D	489D	0143	00000010
43	E81400	call 0144	0232	0031	0000	0932	FFFE	489D	489D	489D	012D	00000010
44	B200	mov dl, 00	0232	0031	0000	0932	FFFC	489D	489D	489D	0144	00000010
45	B402	mov ah, 02	0232	0031	0000	0900	FFFC	489D	489D	489D	0146	00000010
46	CD21	int 21	0232	0031	0000	0900	FFFC	489D	489D	489D	0148	00000010
47	C3	ret	0200	0031	0000	0900	FFFC	489D	489D	489D	014A	00000010
48	8BD3	mov dx, bx	0200	0031	0000	0900	FFFE	489D	489D	489D	0130	00000010
49	E80A00	call 013F	0200	0031	0000	0031	FFFE	489D	489D	489D	0132	00000010
50	B402	mov ah, 02	0200	0031	0000	0031	FFFC	489D	489D	489D	013F	00000010
51	CD21	int 21	0200	0031	0000	0031	FFFC	489D	489D	489D	0141	00000010
52	C3	ret	0231	0031	0000	0031	FFFC	489D	489D	489D	0143	00000010
53	B8004C	mov ax, 4C00	0231	0031	0000	0031	FFFE	489D	489D	489D	0135	00000010
54	CD21	int 21	4C00	0031	0000	0031	FFFE	489D	489D	489D	0138	00000010

2 Ответы на контрольные вопросы

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

В регистр DL заносится номер используемой для операции вывода функции. Вывод информации в ассемблерных программах осуществляется обычно при помощи сервисной функции DOS (прерывание 21h) (код int 02h)

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

Для вывода на экран цифры необходимо сначала преобразовать её в символьную форму. То есть, для получения символьной формы необходимо заменить цифру кодом ASCII её изображения. Для этой цели используем команду ADD арифметического сложения цифры, содержащейся в регистре, с числом 30h – шестнадцатеричным кодом 0.

```
1 add AX, 30h
```

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

Практически все современные программы состоят из одной главной программы и небольших частей, то есть подпрограмм (или **процедур**). (Главная программа вызывает эти процедуры на выполнение, передавая им управление процессором. После завершения работы процедуры возвращают управление главной программе и выполнение продолжается с команды, следующей за командой вызова подпрограммы) Достоинством такого метода является возможность разработки программ значительно большего объема небольшими функционально законченными частями.

Директива PROC процедуры MAIN имеет атрибут FAR, который связан с выполнением программы, а именно когда вы запрашиваете выполнение программы, загрузчик использует эту процедуру, как начальную точку для определения первой подлежащей исполнению команды.

Директива ENDP указывает на конец процедуры и содержит то же имя, что и предложение PROC, чтобы позволить ассемблеру соотнести конец процедуры и ее начало. Поскольку процедура должна полностью содержаться в одном сегменте, ENDP завершает процедуру перед тем, как ENDS определяет конец сегмента.

Пример:

```
1
2 MAIN PROC FAR ; название подпрограммы PROC тип (FAR или NEAR)
3 ;Запись адреса PSP в стек
4 PUSH DS
5 XOR AX,AX
6 PUSH AX
7 ...
8 RET ;Возврат к PSP
9 MAIN ENDP ; название процедуры endp
10
```

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

Язык программирования ассемблера поддерживает применение процедур двух типов – ближнего (near) и дальнего (far).

Процедуры ближнего типа должны находиться в том же сегменте, что и вызывающая программа. Дальний тип процедуры означает, что к ней можно обращаться из любого другого кодового сегмента.

В общем случае, размещать подпрограмму в теле программы можно где угодно, но принято размещать либо в конце сегмента кода, после команд завершения программы, либо в самом начале сегмента кода, перед точкой входа в программу (т.к. процедура не должна выполняться без её вызова). В больших программах подпрограммы часто размещают в отдельном кодовом сегменте.

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

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

Процедура вызывается с помощью команды call. При вызове процедуры в стеке сохраняется адрес возврата в вызывающую программу.

Пример из трассировки первой программы: до вызова процедуры `out_string` $SP = FFFE$, а после вызова $SP = FFFC$, после завершения процедуры SP становится вновь равен $FFFE$. SP уменьшается ровно на 2 байта во время вызова процедуры из-за сохранения адреса возврата в стеке.

7. После какой команды процедуры из стека извлекается адрес возврата?

Адрес возврата извлекается из стека после завершения работы процедуры командой `ret`.