МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования

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

Отчёт о практике

студента 2 курса 251 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Толстова Роберта Сергеевича

Проверено:

Старший преподаватель _____ Е. М. Черноусова

Саратов 2024

СОДЕРЖАНИЕ

1 Исходный код программ

1.1 Задание 1

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

Фото запуска первой программы

```
.model small
.stack 100h

.data
Names db 'Tolstov Robert 251', 0Dh, 0Ah, '$'
enter db 0Dh, 0Ah, '$'

.code

start:
mov AX, @data
mov DS, AX

;адрес начала строки
```

```
mov DX, offset Names
call cout

mov AL, 6 ; занесение первой цифры в регистр AL
mov BL, 9 ; занесение второй цифры в регистр BL

push AX ; добавление AX в стек

mov DL, AL  ; переносим AL в DL, чтобы вывести символ в AL
add DL, 30h ; перевод цифры в символьную форму с помощью кода ASCII
и команды ADD
call coutElem

mov DL, ' ' ; переносим AL в DL, чтобы вывести символ в AL
call coutElem

mov DL, BL  ; переносим BL в DL, чтобы вывести символ в BL
add DL, 30h ;перевод цифры в символьную форму с помощью кода ASCII
и команды ADD
call coutElem

pop AX ; удаление AX из стека

mov AX,4C00h ; функция завершения программы 4Ch с кодом возврата 0
int 21h      ; вызов функции DOS

; вывод строки
cout proc
    mov AH, 09h ;Функция DOS 09h вывода на экран
    int 21h ;вызов функции DOS 09h
    ret ; возврат в точку вызова
cout endp

; вывод символа
coutElem proc
    mov AH, 02h ; номер функции вывода одного символа
    int 21h ;вызов функции 02h
    ret ; возврат в точку вызова
coutElem endp

end start
```

Код первой программы

Шаг	Машинный код	Команда	Регистры									Флаги
			AX	BX	CX	DX	SP	DS	SS	CS	IP	CZSOPAID
1	B409	mov ah, 09	0000	0000	0000	0000	FFFE	489D	489D	489D	0100	00000010
2	BA3301	mov dx, 0133	0900	0000	0000	0000	FFFE	489D	489D	489D	0102	00000010
3	CD21	int 21	0900	0000	0000	0133	FFFE	489D	489D	489D	0105	00000010
4	E82100	call 012B	0900	0000	0000	0133	FFFE	489D	489D	489D	0107	00000010
5	BA4501	mov dx, 0145	0900	0000	0000	0133	FFFC	489D	489D	489D	012B	00000010
6	B409	mov ah, 09	0900	0000	0000	0145	FFFC	489D	489D	489D	012E	00000010
7	CD21	int 21	0900	0000	0000	0145	FFFC	489D	489D	489D	0130	00000010
8	C3	ret	0900	0000	0000	0145	FFFC	489D	489D	489D	0132	00000010
9	B80700	mov ax, 0007	0900	0000	0000	0145	FFFE	489D	489D	489D	010A	00000010
10	053000	add ax, 0030	0007	0000	0000	0145	FFFE	489D	489D	489D	010D	00000010
11	BB0500	mov bx,0005	0037	0000	0000	0145	FFFE	489D	489D	489D	0110	00000010
12	83C330	add bx, 0030	0037	0005	0000	0145	FFFE	489D	489D	489D	0113	00000010
13	8BD0	mov dx, ax	0037	0035	0000	0145	FFFE	489D	489D	489D	0116	00001010
14	B402	mov ah, 02	0037	0035	0000	0037	FFFE	489D	489D	489D	0118	00001010
15	CD21	int 21	0237	0035	0000	0037	FFFE	489D	489D	489D	011A	00001010
16	B200	mov dl, 00	0237	0035	0000	0037	FFFE	489D	489D	489D	011C	00001010
17	CD21	int 21	0237	0035	0000	0000	FFFE	489D	489D	489D	011E	00001010
18	8BD3	mov dx, bx	0200	0035	0000	0000	FFFE	489D	489D	489D	0120	00001010
19	B402	mov ah, 02	0200	0035	0000	0035	FFFE	489D	489D	489D	0122	00001010
20	CD21	int 21	0200	0035	0000	0035	FFFE	489D	489D	489D	0124	00001010
21	B8004C	mov ax, 4C00	0235	0035	0000	0035	FFFE	489D	489D	489D	0126	00001010
22	CD21	int 21	4C00	0035	0000	0035	FFFE	489D	489D	489D	0129	00001010
23	CD21	int 21	0192	2110	F670	0BB2	0106	2110	0192	0000	0000	10100011

Таблица трассировки первой программы

1.2 Задание 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 (командой XCHG).

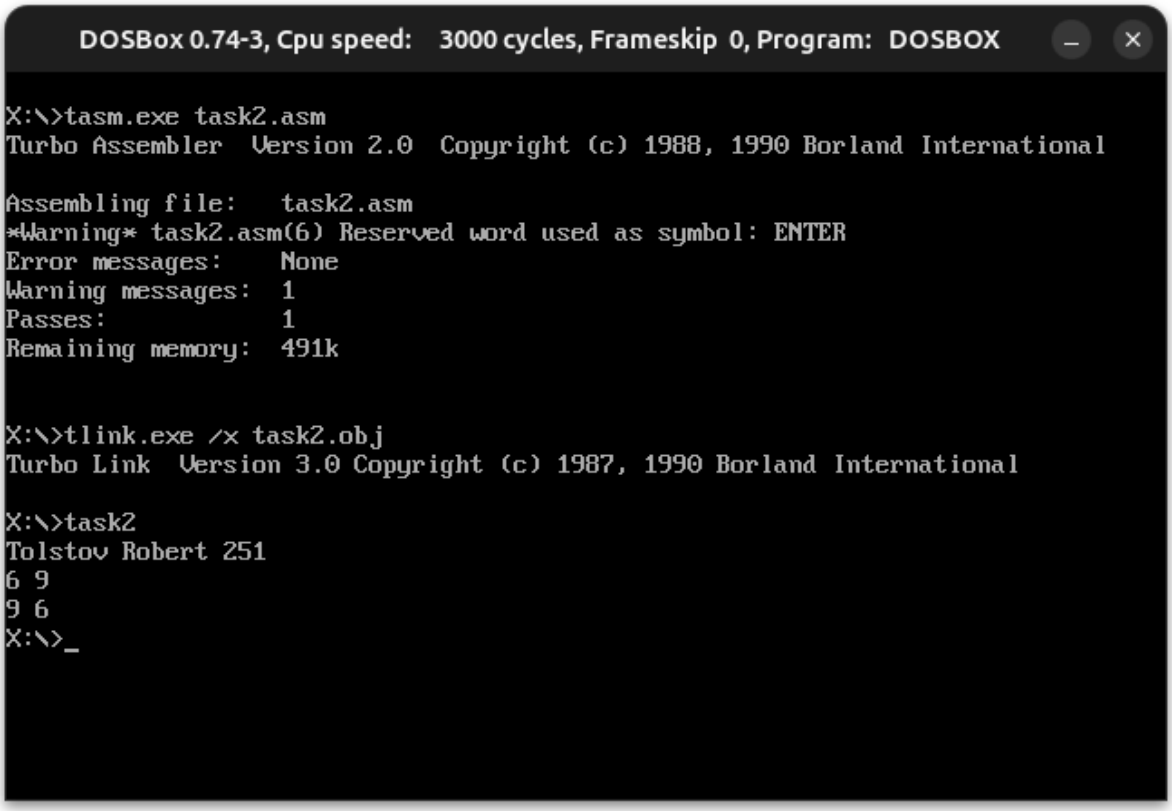

Фото запуска второй программы

```
.model small
.stack 100h

.data
Names db 'Tolstov Robert 251', 0Dh, 0Ah, '$'
enter db 0Dh, 0Ah, '$'

.code

start:
mov AX, @data
mov DS, AX

;адрес начала строки
mov DX, offset Names
call cout

mov AL, 6 ; занесение первой цифры в регистр AL
mov BL, 9 ; занесение второй цифры в регистр BL

push AX ; добавление AX в стек
```

```
mov DL, AL  ; переносим AL в DL, чтобы вывести символ в AL
add DL, 30h ; перевод цифры в символьную форму с помощью кода ASCII
и команды ADD
call coutElem

mov DL, ' ' ; переносим AL в DL, чтобы вывести символ в AL
call coutElem

mov DL, BL  ; переносим BL в DL, чтобы вывести символ в BL
add DL, 30h ;перевод цифры в символьную форму с помощью кода ASCII и
команды ADD
call coutElem

pop AX ; удаление AX из стека

XCHG AL, BL ; меняем местами AL и BL

mov DX, offset enter ;адрес начала строки
call cout

mov DL, AL  ; переносим AL в DL, чтобы вывести символ в AL
add DL, 30h ; перевод цифры в символьную форму с помощью кода ASCII
и команды ADD
call coutElem

mov DL, ' ' ; переносим AL в DL, чтобы вывести символ в AL
call coutElem

mov DL, BL  ; переносим BL в DL, чтобы вывести символ в BL
add DL, 30h ; перевод цифры в символьную форму с помощью кода ASCII
и команды ADD
call coutElem

mov AX,4C00h ; функция завершения программы 4Ch с кодом возврата 0
int 21h      ; вызов функции DOS

; вывод строки
cout proc
    mov AH, 09h ;Функция DOS 09h вывода на экран
    int 21h ;вызов функции DOS 09h
    ret ; возврат в точку вызова
cout endp

; вывод символа
```

```
coutElem proc
    mov AH, 02h ; номер функции вывода одного символа
    int 21h ;вызов функции 02h
    ret ; возврат в точку вызова
coutElem endp

end start
```

Код второй программы

Шаг	Машинный код	Команда	Регистры									Флаги
			AX	BX	CX	DX	SP	DS	SS	CS	IP	CZSOPAID
1	B8B348	mov ax, 48B3	0000	0000	0000	0000	0100	489D	48B5	48AD	0000	00000010
2	8ED8	mov dx, ax	48B3	0000	0000	0000	0100	489D	48B5	48AD	0003	00000010
3	BA0000	mov dx, 0000	48B3	0000	0000	0000	0100	48B3	48B5	48AD	0005	00000010
4	E83D00	call 0048	48B3	0000	0000	0000	0100	48B3	48B5	48AD	0008	00000010
5	B409	mov ah, 09	48B3	0000	0000	0000	00FE	48B3	48B5	48AD	0048	00000010
6	CD21	int 21	09B3	0000	0000	0000	00FE	48B3	48B5	48AD	004A	00000010
7	C3	ret	09B3	0000	0000	0000	00FE	48B3	48B5	48AD	004C	00000010
8	B007	mov al, 07	09B3	0000	0000	0000	0100	48B3	48B5	48AD	000B	00000010
9	B305	mov bl, 05	0907	0000	0000	0000	0100	48B3	48B5	48AD	000D	00000010
10	50	push ax	0907	0005	0000	0000	0100	48B3	48B5	48AD	000F	00000010
11	8AD0	mov dl, al	0907	0005	0000	0000	00FE	48B3	48B5	48AD	0010	00000010
12	80C230	add dl, 30	0907	0005	0000	0007	00FE	48B3	48B5	48AD	0012	00000010
13	E83500	call 004D	0907	0005	0000	0037	00FE	48B3	48B5	48AD	0015	00000010
14	B402	mov ah, 02	0907	0005	0000	0037	00FC	48B3	48B5	48AD	004D	00000010
15	CD21	int 21	0207	0005	0000	0037	00FC	48B3	48B5	48AD	004F	00000010
16	C3	ret	0237	0005	0000	0037	00FC	48B3	48B5	48AD	0051	00000010
17	B220	mov dl, 20	0237	0005	0000	0037	00FE	48B3	48B5	48AD	0018	00000010
18	E83000	call 004D	0237	0005	0000	0020	00FE	48B3	48B5	48AD	001A	00000010
19	8AD3	mov dl, bl	0220	0005	0000	0020	00FE	48B3	48B5	48AD	001D	00000010
20	80C230	add dl, 30	0220	0005	0000	0005	00FE	48B3	48B5	48AD	001F	00000010
21	E82800	call 004D	0220	0005	0000	0035	00FE	48B3	48B5	48AD	0022	00001010
22	58	pop ax	0235	0005	0000	0035	00FE	48B3	48B5	48AD	0025	00001010
23	86C3	xchg bl, al	0907	0005	0000	0035	0100	48B3	48B5	48AD	0026	00001010
24	BA1400	mov dx, 0014	0905	0007	0000	0035	0100	48B3	48B5	48AD	0028	00001010
25	E81A00	call 0048	0905	0007	0000	0014	0100	48B3	48B5	48AD	002B	00001010
26	8AD0	mov dl, al	0905	0007	0000	0014	0100	48B3	48B5	48AD	002E	00001010
27	80C230	add dl, 30	0905	0007	0000	0005	0100	48B3	48B5	48AD	0030	00001010
28	E81700	call 004D	0905	0007	0000	0035	0100	48B3	48B5	48AD	0033	00001010
29	B220	mov dl, 20	0235	0007	0000	0035	0100	48B3	48B5	48AD	0036	00001010
30	E81200	call 004D	0235	0007	0000	0020	0100	48B3	48B5	48AD	0038	00001010
31	8AD3	mov dl, bl	0220	0007	0000	0020	0100	48B3	48B5	48AD	003B	00001010
32	80C230	add dl, 30	0220	0007	0000	0007	0100	48B3	48B5	48AD	003D	00001010
33	E80A00	call 004D	0220	0007	0000	0037	0100	48B3	48B5	48AD	0040	00000010
34	B8004C	mov ax, 4C00	0237	0007	0000	0037	0100	48B3	48B5	48AD	0043	00000010
35	CD21	int 21	4C00	0007	0000	0037	0100	48B3	48B5	48AD	0046	00000010
36	CD21	int 21	0192	2110	F670	0BB2	0106	2110	0192	0000	0000	10100011

Таблица трассировки второй программы

2 Ответы на контрольные вопросы

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

Код выводимого символа помещается в регистр `DL`. Это стандартный регистр, который используется для передачи данных в функцию вывода символов. Код DOS-функции для вывода отдельного символа на экран – это `02h`. При вызове этой функции через прерывание `int 21h` содержимое регистра `DL` интерпретируется как символ, который будет выведен на экран.

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

Для получения кода цифры в кодовой таблице ASCII используется операция сложения с символом `'0'`. Например, если у вас есть число в регистре, вы можете добавить к нему значение ASCII символа `'0'` (которое равно 48 в десятичной системе). Это преобразует число от 0 до 9 в соответствующий ASCII-код:

```
add ax, '0' ; Преобразование числа в ASCII
```

Ну а если нам нужно получить именно цифру как символ, то используем команду ADD арифметического сложения цифры, содержащейся в регистре BX, с числом 30h – шестнадцатеричным кодом 0:

```
add ax, '30h' ; Преобразование цифры в её символьное представление
```

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

Назначение процедуры заключается в том, чтобы выполнять определённые действия или задачи, которые могут быть повторно использованы в программе. Процедуры помогают структурировать код, делают его более читаемым и облегчают отладку.

Начало процедуры определяется ключевым словом proc, за которым следует имя процедуры. Конец процедуры обозначается ключевым словом endp. Например:

```
myProcedure proc
    ; тело процедуры
myProcedure endp
```

2.4 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.

Принято размещать подпрограммы 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.

Кратко: процедуры могут быть:

- в начале кода перед входом в программу;
- в сегменте кода внутри программы;
- в конце кода после завершения программы.

Глобально, главная процедура может располагаться как перед подпрограммами, так и после них. Это зависит от предпочтений программиста и структуры программы. Важно лишь правильно организовать вызовы процедур, чтобы компилятор мог их найти.

2.5 Как процессор использует стек при работе с любой процедурой?

При вызове процедуры информация о точке возврата помещается в стек. Это позволяет процессору вернуться к правильному месту после завершения выполнения процедуры. Стек работает по принципу “последний пришёл — первый вышел” (LIFO). Когда процедура завершается, адрес возврата извлекается из стека, и управление передаётся обратно в программу.

Важно понимать, что при вызове ближней процедуры в стек записывается слово, содержащее смещение точки возврата относительно текущего кодового сегмента, а при вызове дальней – слово, содержащее адрес сегмента, в котором расположена точка возврата, и слово, содержащее смещение точки возврата в этом сегменте.

2.6 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 выполнении этой команды значение регистра SP (Stack Pointer) уменьшается на размер адреса возврата (обычно на 2 байта для 16-битных адресов). Например:

```
call myProcedure ; SP уменьшается на 2
```

Если перед вызовом SP равен 0x1000, то после вызова он станет 0x0FFE.

2.7 После какой команды процедуры из стека извлекается адрес возврата?

Адрес возврата извлекается из стека после выполнения команды ret. Эта команда не только возвращает управление обратно по адресу, который был сохранён в стеке, но также увеличивает указатель стека SP на размер адреса возврата:

```
ret ; Извлечение адреса возврата из стека
```