

Саратовский государственный университет им. Н.Г.Чернышевского
Факультет компьютерных наук и информационных технологий

Машинно-зависимые языки программирования

Лабораторная работа №2

Выполнил
студент 211 группы
Коновалов Александр

Содержание

1	Задания	2
1.1	Задание 2.1.	2
1.2	Задание 2.2.	2
2	Программы на языке ассемблера	3
2.1	Задание 2.1.	3
2.2	Задание 2.2.	4
3	Запуск программы	6
4	Трассировки программ	8
5	Контрольные вопросы	9
5.1	В какой регистр надо поместить код выводимого символа? Какой код Dos-функции используется для вывода отдельного символа на экран?	9
5.2	Какая операция позволяет получить для цифры её код в кодовой таблице?	9
5.3	Объясните назначение процедуры. Как определяются начало и конец процедуры? .	9
5.4	Ваша программа состоит из главной процедуры и процедур-подпрограмм. Каким может быть взаимное расположение главной процедуры и подпрограмм?	10
5.5	Как процессор использует стек при работе с любой процедурой?	10
5.6	С помощью какой команды вызывается процедура? Как меняется значение регистра SP после вызова процедуры? Приведите пример из вашей таблицы трассировки. . .	10
5.7	После какой команды процедуры из стека извлекается адрес возврата?	11

1 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.**

2 Программы на языке ассемблера

2.1 Задание 2.1.

```
.model small ;Модель памяти SMALL использует сегменты размером не более 64Кб
.stack 100h ;Сегмент стека размером 100h (256 байт)
.data ;Начало сегмента данных
student db 'Коновалов Александр 211', 0Dh, 0Ah, '$'
.code ;Начало сегмента кода
start: ;Точка входа в программу start

;Предопределенная метка @data обозначает
;адрес сегмента данных в момент запуска программы,
mov AX, @data ;который сначала загрузим в AX,
mov DS, AX ;а затем перенесем из AX в DS

call print_student ;вызываем процедуру
;по условию нам заданы две цифры в регистрах ax, bx соответственно
mov AX, 7 ;1 цифра
mov BX, 3 ;2 цифра

add AX, 30h ;Преобразуем цифру по ASCII таблице(пример: по таблице 1 под кодом 31, т
mov DL,AL ;DL - выводимый символ
call print_num ;вызываем процедуру

mov DL,20h ;- выводимый символ(пробел по ascii)
mov ah,02h ;AH - номер функции
int 21h ;Инициализация прерывания

add BX, 30h ;Преобразуем цифру по ASCII таблице(пример: по таблице 1 под кодом 31, т
mov DL,BL ;DL - выводимый символ
call print_num ;вызываем

mov AX,4C00h
int 21h ;Инициализация прерывания

print_num proc
    mov AH,02h ;AH - номер функции
    int 21h ;Инициализация прерывания
    ret
print_num endp

print_student proc
    mov AH,09h ;AH - номер функции
    mov DX,offset student ;dx - выводимая строка
    int 21h ;Инициализация прерывания
    ret
print_student endp

end start ;Конец текста программы с точкой входа
```

2.2 Задание 2.2.

```
.model small ;Модель памяти SMALL использует сегменты размером не более 64Кб
.stack 100h ;Сегмент стека размером 100h (256 байт)
.data ;Начало сегмента данных
student db 'Коновалов', 'Александр', 211$
newline db 0Dh, 0Ah, '$'
.code ;Начало сегмента кода
start: ;Точка входа в программу start

;Предопределенная метка @data обозначает
;адрес сегмента данных в момент запуска программы,
mov AX, @data ;который сначала загрузим в AX,
mov DS, AX ;а затем перенесем из AX в DS

call print_student ;вызываем процедуру
call new_line ;вызываем процедуру

;по условию нам заданы две цифры в регистрах ax, bx соответственно
mov AX, 7 ;1 цифра
mov BX, 3 ;2 цифра

add AX, 30h ;Преобразуем цифру по ASCII таблице(пример: по таблице 1 под кодом 31, т.е. '1')
push AX;сохранили значение, т.к. далее код символа попадает в младшую часть регистра
mov DL,AL ;DL - выводимый символ
call print_num ;вызываем процедуру

call space ;вызываем процедуру

add BX, 30h ;Преобразуем цифру по ASCII таблице(пример: по таблице 1 под кодом 31, т.е. '1')
mov DL,BX ;DL - выводимый символ
call print_num ;вызываем

call new_line ;вызываем процедуру

pop AX ;извлекаем из стека
XCHG AX, BX ;обмен данных между регистрами

mov DL,AL ;DL - выводимый символ
call print_num ;вызываем процедуру
call space ;вызываем процедуру
mov DL,BX ;DL - выводимый символ
call print_num ;вызываем

mov AX,4C00h ;Функция 4Ch завершения программы с кодом возврата 0
int 21h ;Инициализация прерывания

print_num proc
    mov AH,02h ;AH - номер функции
    int 21h ;Инициализация прерывания
    ret
```

```

print_num endp

print_student proc
    mov AH,09h ;AH - номер функции
    mov DX,offset student ;DX - выводимая строка
    int 21h ;Инициализация прерывания
    ret
print_student endp

new_line proc
    mov DX, offset newline ;DX - выводимая строка
    mov AH,09h ;AH - номер функции
    int 21h ;Инициализация прерывания
    ret
new_line endp

space proc
    mov DL,20h ;DL - выводимый символ
    mov ah,02h ; ;AH - номер функции
    int 21h ; ;Инициализация прерывания
    ret
space endp

end start ;Конец текста программы с точкой входа

```

3 Запуск программы

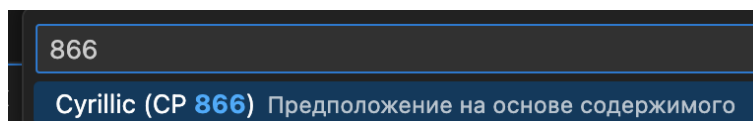


Рис. 1: Сохранение в нужной кодировке при помощи VS Code

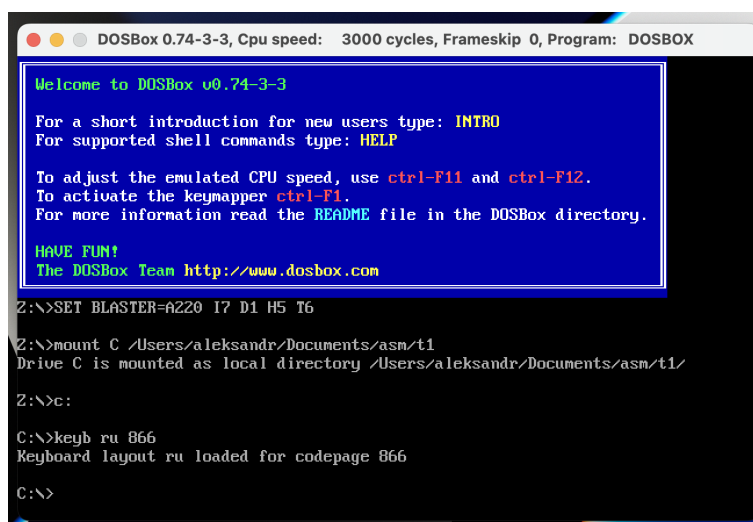


Рис. 2: Применение нужной кодировки в DosBox

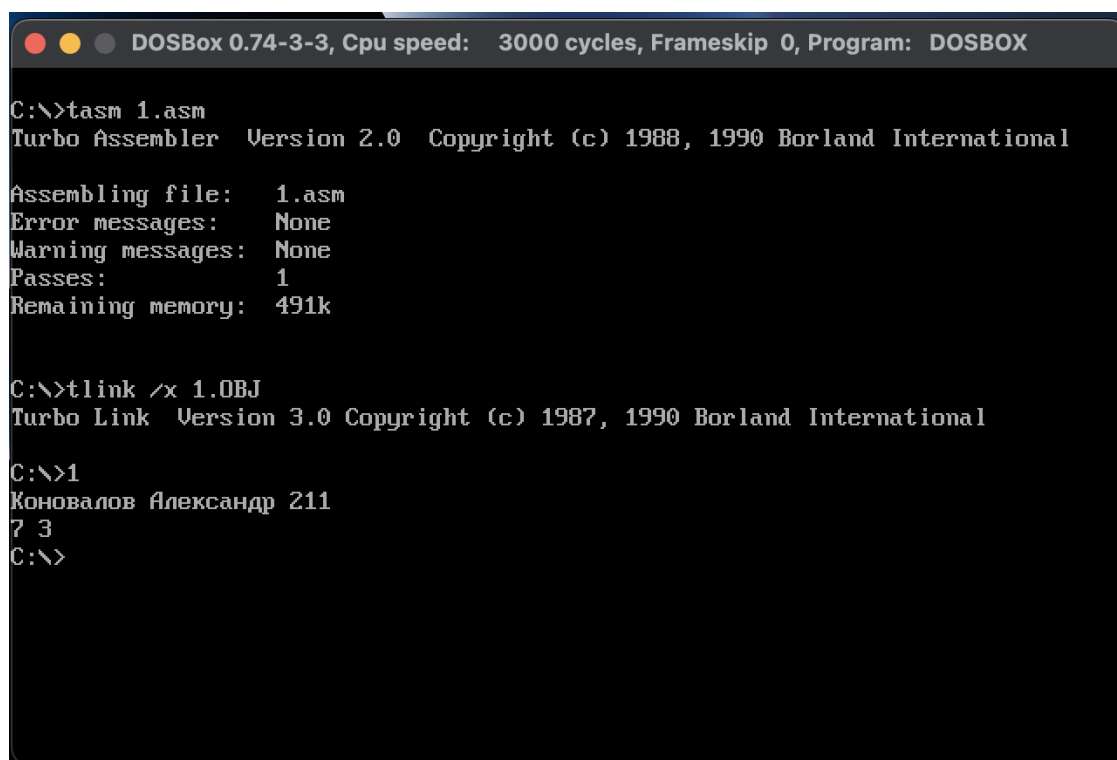
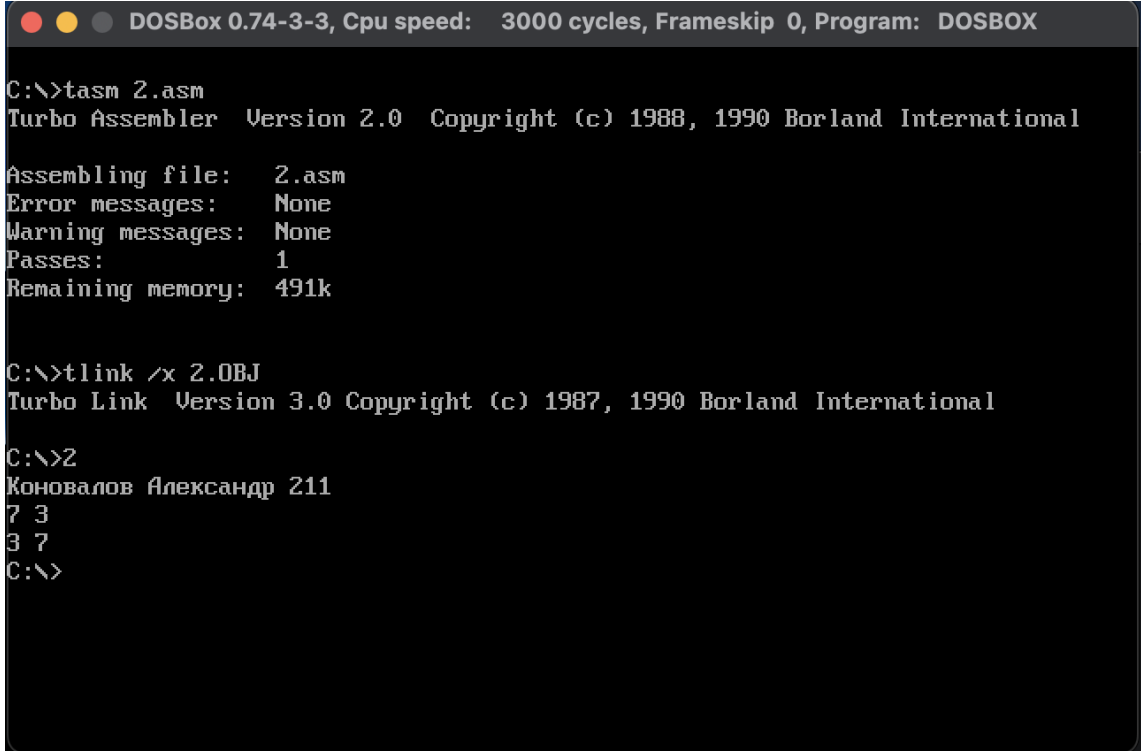


Рис. 3: Запуск 1 программы



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

```
C:\>tasm 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 /x 2.OBJ
Turbo Link Version 3.0 Copyright (c) 1987, 1990 Borland International

C:\>2
Коновалов Александр 211
7 3
3 7
C:\>
```

Рис. 4: Запуск 2 программы

4 Трассировки программ

Шаг	Машинный код	Команда	Регистры								ИП	Флаги
			AX	BX	CX	DX	SP	DS	SS	CS		
			0000	0000	0000	0000	0100	489D	48B3	48AD	0000	CZSOPAD
1	B8B148	mov ax,48B1	48B1	0000	0000	0000	0100	489D	48B3	48AD	0003	00000010
2	8ED8	mov ds,ax	48B1	0000	0000	0000	0100	48B1	48B3	48AD	0005	00000010
3(print_student)	E82600	call 002E	48B1	0000	0000	0000	00FE	48B1	48B3	48AD	002E	00000010
3.1	B409	mov ah,09	09B1	0000	0000	0000	00FE	48B1	48B3	48AD	0030	00000010
3.2	BA0000	mov dx,0000	09B1	0000	0000	0000	00FE	48B1	48B3	48AD	0033	00000010
3.3	CD21	int 21	09B1	0000	0000	0000	00FE	48B1	48B3	48AD	0035	00000010
3.4	C3	ret	09B1	0000	0000	0000	0100	48B1	48B3	48AD	0008	00000010
4	B80700	mov ax,0007	0007	0000	0000	0000	0100	48B1	48B3	48AD	000B	00000010
5	BB0300	mov bx,0003	0007	0003	0000	0000	0100	48B1	48B3	48AD	000E	00000010
6	053000	add ax,0030	0037	0003	0000	0000	0100	48B1	48B3	48AD	0011	00000010
7	8AD0	mov dl,al	0037	0003	0000	0037	0100	48B1	48B3	48AD	0013	00000010
8(print_num)	E81300	call 0029	0037	0003	0000	0037	00FE	48B1	48B3	48AD	0029	00000010
8.1	B402	mov ah,02	0237	0003	0000	0037	00FE	48B1	48B3	48AD	002B	00000010
8.2	CD21	int 21	0237	0003	0000	0037	00FE	48B1	48B3	48AD	002D	00000010
8.3	C3	ret	0237	0003	0000	0037	0100	48B1	48B3	48AD	0016	00000010
9	B220	mov dl,20	0237	0003	0000	0020	0100	48B1	48B3	48AD	0018	00000010
10	B402	ah,02	0237	0003	0000	0020	0100	48B1	48B3	48AD	001A	00000010
11	CD21	int 21	0220	0003	0000	0020	0100	48B1	48B3	48AD	001C	00000010
12	83C330	bx, 0030	0220	0033	0000	0020	0100	48B1	48B3	48AD	001F	00001010
13	8AD3	mov dl,bl	0220	0033	0000	0033	0100	48B1	48B3	48AD	0021	00001010
14(print_num)	E80500	call 0029	0220	0033	0000	0033	00FE	48B1	48B3	48AD	0029	00001010
14.1	B402	mov ah,02	0220	0033	0000	0033	00FE	48B1	48B3	48AD	002B	00001010
14.2	CD21	int 21	0233	0033	0000	0033	00FE	48B1	48B3	48AD	002D	00001010
14.3	C3	ret	0233	0033	0000	0033	0100	48B1	48B3	48AD	0024	00001010
15	B8004C	mov ax,4C00	4C00	0033	0000	0033	0100	48B1	48B3	48AD	0027	00001010
16	CD21	int 21	0192	000B	F711	098D	0106	2110	0192	0000	0000	10100011

Рис. 5: Трассировка первой программы

Шаг	Машинный код	Команда	Регистры								ИП	Флаги
			AX	BX	CX	DX	SP	DS	SS	CS		
			0000	0000	0000	0000	0100	489D	48B5	48AD	0000	CZSOPAD
1	B8B348	mov ax,48B3	48B3	0000	0000	0000	0100	489D	48B5	48AD	0003	00000010
2	8ED8	mov ds,ax	48B3	0000	0000	0000	0100	48B3	48B5	48AD	0005	00000010
3(print_stud)	E83900	call 0041	48B3	0000	0000	0000	00FE	48B3	48B5	48AD	0041	00000010
3.1	B409	mov ah,09	09B3	0000	0000	0000	00FE	48B3	48B5	48AD	0043	00000010
3.2	BA0000	mov dx,0000	09B3	0000	0000	0000	00FE	48B3	48B5	48AD	0046	00000010
3.3	CD21	int 21	09B3	0000	0000	0000	00FE	48B3	48B5	48AD	0048	00000010
3.4	C3	ret	09B3	0000	0000	0000	0100	48B3	48B5	48AD	0008	00000010
4(new_line)	E83900	call 0049	09B3	0000	0000	0000	00FE	48B3	48B5	48AD	0049	00000010
4.1	BA1800	mov dx,0018	09B3	0000	0000	0018	00FE	48B3	48B5	48AD	004C	00000010
4.2	B409	mov ah,09	09B3	0000	0000	0018	00FE	48B3	48B5	48AD	004E	00000010
4.3	CD21	int 21	09B3	0000	0000	0018	00FE	48B3	48B5	48AD	0050	00000010
4.4	C3	ret	09B3	0000	0000	0018	0100	48B3	48B5	48AD	000B	00000010
5	B80700	mov ax,0007	0007	0000	0000	0018	0100	48B3	48B5	48AD	000E	00000010
6	BB0300	mov bx,0003	0007	0003	0000	0018	0100	48B3	48B5	48AD	0011	00000010
7	053000	add ax,0030	0037	0003	0000	0018	0100	48B3	48B5	48AD	0014	00000010
8	50	push ax	0037	0003	0000	0018	00FE	48B3	48B5	48AD	0015	00000010
9	8AD0	mov dl,al	0037	0003	0000	0037	00FE	48B3	48B5	48AD	0017	00000010
10(print_num)	E82200	call 003C	0037	0003	0000	0037	00FC	48B3	48B5	48AD	003C	00000010
10.1	B402	mov ah,02	0237	0003	0000	0037	00FC	48B3	48B5	48AD	003E	00000010
10.2	CD21	int 21	0237	0003	0000	0037	00FC	48B3	48B5	48AD	0040	00000010
10.3	C3	ret	0237	0003	0000	0037	00FE	48B3	48B5	48AD	001A	00000010
11(space)	E83400	call 0051	0237	0003	0000	0037	00FC	48B3	48B5	48AD	0051	00000010
11.1	B220	mov dl,20	0237	0003	0000	0020	00FC	48B3	48B5	48AD	0053	00000010
11.2	B402	mov ah,02	0237	0003	0000	0020	00FC	48B3	48B5	48AD	0055	00000010
11.3	CD21	int 21	0220	0003	0000	0020	00FC	48B3	48B5	48AD	0057	00000010
11.4	C3	ret	0220	0003	0000	0020	00FE	48B3	48B5	48AD	001D	00000010
12	83C330	add bx,0030	0220	0033	0000	0020	00FE	48B3	48B5	48AD	0020	00001010
13	8AD3	mov dl,bl	0220	0033	0000	0033	00FE	48B3	48B5	48AD	0022	00001010
14							(print_num)					
15							(new_line)					
16	58	pop ax	0037	0033	0000	0018	0100	48B3	48B5	48AD	0029	00001010
17	93	xchg bx,ax	0033	0037	0000	0018	0100	48B3	48B5	48AD	002A	00001010
18	8AD0	mov dl,al	0033	0037	0000	0033	0100	48B3	48B5	48AD	002C	00001010
19							(print_num)					
20							(space)					
21	8AD3	mov dl,bl	0220	0037	0000	0020	0100	48B3	48B5	48AD	0032	00001010
22							(print_num)					
23	B8004C	mov ax, 4C00	4C00	0037	0000	0037	0100	48B3	48B5	48AD	003A	00001010
24	CD21	int 21	0192	000B	F711	098D	0106	2110	0192	0000	0000	10100011

Рис. 6: Трассировка второй программы

5 Контрольные вопросы

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

Пересылка кода символа выполняется в регистр DL.

Вывод информации в ассемблерных программах осуществляется обычно при помощи сервисных функций DOS (прерывание 21h). Процесс вывода состоит в следующем:

- определенные регистры микропроцессора загружаются выводимой информацией или адресом буфера, содержащего выводимую информацию;
- в регистр AH заносится номер используемой для операции вывода функции;
- инициируется прерывание.

Функция 02h

Вывод на дисплей.

Вход: AH=02h

DL=выводимый символ

Выход: нет

Описание: Посылает символ из DL на стандартный вывод. Обработывает символ Backspace (ASCII 8), перемещая курсор влево на одну позицию и оставляя его в новой позиции.

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

Для вывода на экран цифры необходимо сначала преобразовать ее в символьную форму. Действительно, если послать на экран, например, код 5, мы увидим изображение трефового туза, чтобы получить на экране цифру 5, надо послать код ASCII этого символа, т.е. число 35h. Таким образом, для получения символьной формы необходимо заменить цифру кодом ASCII ее изображения. Используйте для этой цели команду ADD арифметического сложения цифры, содержащейся в регистре BX, с числом 30h — шестнадцатеричным кодом 0.

5.3 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. 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. 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 64К, то при разработке программ с объемом кода более 64К, просто не обойтись без модульного принципа.

Описание процедуры имеет следующий синтаксис:

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

5.4 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 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.



Рис. 7: Варианты размещения подпрограммы в теле программы.

5.5 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.

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

Для работы с подпрограммами в систему команд процессора включены специальные команды, это вызов подпрограммы CALL и возврат управления RET.

Все команды вызова CALL безусловны. Внутрисегментный вызов NEAR CALL используется для передачи управления процедуре, находящейся в том же сегменте. Он указывает новое значение регистра IP и сохраняет в стеке адрес возврата, т.е. IP команды, следующей за командой CALL.

Межсегментный вызов FAR CALL используется для передачи управления процедуре, находящейся в другом сегменте или даже программном модуле. Он задает новые значения сегмента CS и смещения IP для дальнейшего выполнения программы и сохраняет в стеке как регистр IP, так и регистр CS.

			SP
2	8ED8	mov ds,ax	0100
3(print_student)	E82600	call 002E	00FE
3.4	C3	ret	0100

Рис. 8: Изменение значения регистра SP после вызова процедуры print student в программе 1

5.7 После какой команды процедуры из стека извлекается адрес возврата?

Все возвраты RET являются косвенными переходами, поскольку извлекают адрес перехода из вершины стека. 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.