

МИНОБРНАУКИ РОССИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования

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

**ЛАБОРАТОРНАЯ РАБОТА №1**

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 1.1

### 1.1 Текст задания

Измените программы из примеров 1, 2 и 3 так, чтобы они выводили на экран ваши фамилию, имя и номер группы. Используя командные файлы (с расширением bat), подготовьте к выполнению и запустите 3 программы. Убедитесь, что они выводят на экран нужный текст и успешно завершаются.

### 1.2 Тексты трёх программ на языке ассемблера с комментариями

```
1 stak segment stack 'stack'           ;Начало сегмента стека
2 db 256 dup (?)                       ;Резервируем 256 байт для стека
3 stak ends                             ;Конец сегмента стека
4 data segment 'data'                  ;Начало сегмента данных
5 MyName db 'Tyumentsev Radomir, 251$' ;Строка для вывода
6 data ends                             ;Конец сегмента данных
7 code segment 'code'                  ;Начало сегмента кода
8 assume CS:code,DS:data,SS:stak       ;Сегментный регистр CS будет указывать на
   сегмент команд,
9                                     ;регистр DS - на сегмент данных, SS – на
                                     стек
10 start:                              ;Точка входа в программу start
11 ;Обязательная инициализация регистра DS в начале программы
12 mov AX,data                          ;Адрес сегмента данных сначала загрузим в
   AX,
13 mov DS,AX                           ;а затем перенесем из AX в DS
14 mov AH,09h                          ;Функция DOS 9h вывода на экран
15 mov DX,offset MyName                ;Адрес начала строки записывается в регистр
   DX
16 int 21h                             ;Вызов функции DOS
17 mov AX,4C00h                        ;Функция 4Ch завершения программы с кодом
   возврата 0
18 int 21h                             ;Вызов функции DOS
19 code ends                           ;Конец сегмента кода
20 end start                           ;Конец текста программы с точкой входа
```

Текст программы 1

```

1 .model small           ;Модель памяти SMALL использует сегменты
2                       ;размером не более 64Кб
3 .stack 100h           ;Сегмент стека размером 100h (256 байт)
4 .data                 ;Начало сегмента данных
5 MyName db 'Tyumentsev Radomir, 251$'
6 .code                 ;Начало сегмента кода
7 start:               ;Точка входа в программу start
8                       ;Предопределенная метка @data обозначает
9                       ;адрес сегмента данных в момент запуска
                        программы,
10 mov AX, @data         ;который сначала загрузим в AX,
11 mov DS,AX             ;а затем перенесем из AX в DS
12 mov AH,09h
13 mov DX,offset MyName
14 int 21h
15 mov AX,4C00h
16 int 21h
17 end start

```

### Текст программы 2

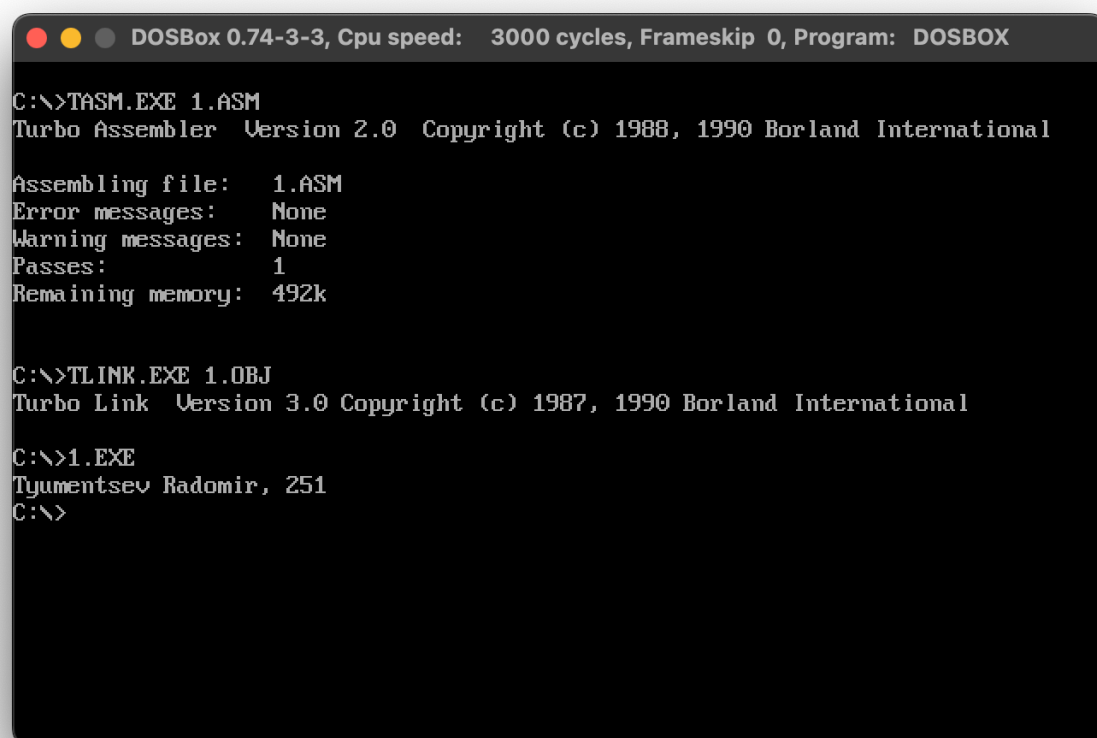
```

1 .model tiny           ;Модель памяти TINY, в которой код, данные и стек
2                       ;размещаются в одном и том же сегменте размером до
                        64Кб
3 .code                 ;Начало сегмента кода
4 org 100h              ;Устанавливает значение программного счетчика в 100h
5                       ;Начало необходимое для COM-программы,
6                       ;которая загружается в память с адреса PSP:100h
7
8 start:
9 mov AH,09h
10 mov DX,offset MyName
11 int 21h
12 mov AX,4C00h
13 int 21h
14 ;===== Data =====
15 MyName db 'Tyumentsev Radomir, 251$'
16 end start

```

### Текст программы 3

### 1.3 Скриншоты запуска трёх программ



```
DOSBox 0.74-3-3, 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: 492k

C:\>TLINK.EXE 1.OBJ
Turbo Link Version 3.0 Copyright (c) 1987, 1990 Borland International

C:\>1.EXE
Tyumentsev Radomir, 251
C:\>
```

Запуск программы 1

```
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: 492k

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

C:\>2.EXE
Tyumentsev Radomir, 251
C:\>
```

## Запуск программы 2

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

C:\>TASM.EXE 3.ASM
Turbo Assembler Version 2.0 Copyright (c) 1988, 1990 Borland International

Assembling file: 3.ASM
Error messages: None
Warning messages: None
Passes: 1
Remaining memory: 492k

C:\>TLINK.EXE /x /t 3.OBJ
Turbo Link Version 3.0 Copyright (c) 1987, 1990 Borland International

C:\>3.COM
Tyumentsev Radomir, 251
C:\>_
```

## **1.4 Тексты 2-х командных файлов (для exe-программ и для com-программы)**

```
1 cls
2 tasm.exe %1.asm
3 tlink.exe /x %1.obj
4 %1
```

Текст командного файла для exe-программ

```
1 cls
2 tasm.exe %1.asm
3 tlink.exe /x /t %1.obj
4 %1
```

Текст командного файла для com-программ

## 2 Задание 1.2

### 2.1 Текст задания

Заполните таблицы трассировки для 3-х программ.

### 2.2 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	B8BD48	mov ax, 48BD	0000	0000	0000	0000	0100	489D	48AD	48BF	0000	00000010
2	8ED8	mov ds, ax	48BD	0000	0000	0000	0100	489D	48AD	48BF	0003	00000010
3	B409	mov ah, 09	48BD	0000	0000	0000	0100	48BD	48AD	48BF	0005	00000010
4	BA0000	mov dx, 0000	09BD	0000	0000	0000	0100	48BD	48AD	48BF	0007	00000010
5	CD21	int 21	09BD	0000	0000	0000	0100	48BD	48AD	48BF	000A	00000010
6	B8004C	mov ax, 4C00	09BD	0000	0000	0000	0100	48BD	48AD	48BF	000C	00000010
7	CD21	int 21	4C00	0000	0000	0000	0100	48BD	48AD	48BF	000F	00000010

Таблица 2 – Таблица трассировки программы 2

Шаг	Машинный код	Команда	Регистры									Флаги
			AX	BX	CX	DX	SP	DS	SS	CS	IP	
1	B8BD48	mov ax, 48AF	0000	0000	0000	0000	0100	489D	48B1	48AD	0000	00000010
2	8ED8	mov ds, ax	48AF	0000	0000	0000	0100	489D	48B1	48AD	0003	00000010
3	B409	mov ah, 09	48AF	0000	0000	0000	0100	48AF	48B1	48AD	0005	00000010
4	BA0000	mov dx, 0000	09AF	0000	0000	0000	0100	48AF	48B1	48AD	0007	00000010
5	CD21	int 21	09AF	0000	0000	0000	0100	48AF	48B1	48AD	000A	00000010
6	B8004C	mov ax, 4C00	09AF	0000	0000	0000	0100	48AF	48B1	48AD	000C	00000010
7	CD21	int 21	4C00	0000	0000	0000	0100	48AF	48B1	48AD	000F	00000010

Таблица 3 – Таблица трассировки программы 3

Шаг	Машинный код	Команда	Регистры									Флаги
			AX	BX	CX	DX	SP	DS	SS	CS	IP	
1	B409	mov ah, 09	0000	0000	0000	0000	FFFE	489D	489D	489D	0100	00000010
2	BA0000	mov dx, 010C	0900	0000	0000	0000	FFFE	489D	489D	489D	0102	00000010
3	CD21	int 21	0900	0000	0000	010C	FFFE	489D	489D	489D	0105	00000010
4	B8004C	mov ax, 4C00	0900	0000	0000	010C	FFFE	489D	489D	489D	0107	00000010
5	CD21	int 21	4C00	0000	0000	010C	FFFE	489D	489D	489D	010A	00000010

### 3 Ответы на контрольные вопросы

#### 1. Что такое сегментный (базовый) адрес?

В микропроцессоре есть несколько 16-битных сегментных регистров: CS, DS, SS и ES. В них хранятся 16-битные значения, которые называются базовым адресом сегмента.

Микропроцессор объединяет 16-битный исполнительный адрес и 16-битный базовый адрес следующим образом: он расширяет содержимое сегментного регистра (базовый адрес) четырьмя нулями в младших разрядах, делая его 20-битным (полный адрес сегмента), и прибавляет смещение (исполнительный адрес). В результате получается 20-битный адрес, который является физическим или абсолютным адресом ячейки памяти. Физический адрес нулевого смещения есть начало сегмента в виртуальной памяти.

**2. Сделайте листинг для первой программы (файл с расширением lst), выпишите из него размеры сегментов. Из таблицы трассировки к этой программе выпишите базовые адреса сегментов (значение DS при этом нам нужно взять после инициализации адресом сегмента данных). В каком порядке расположились сегменты программы в памяти? Расширяя базовый адрес сегмента до физического адреса, прибавляя размер этого сегмента и округляя до кратного 16 значения, мы можем получить физический адрес следующего за ним сегмента. Сделайте это для первых 2-х сегментов. (Если данные не совпали, значит, неверно заполнена таблица трассировки.)**

CODE Size 0011, DATA Size 0018, STAK Size 0100.

Регистры после инициализации DS:

- DS = 48BD
- SS = 48AD
- CS = 48BF

В памяти программа разделена на сегменты, которые расположены в порядке увеличения размера: SS, DS, CS. Физический адрес определяется как базовый адрес, умноженный на 10h, плюс размер сегмента.

Расчёт для SS:  $48ADh \cdot 10h + 0100h = 48AD0h + 0100h = 48BD0h$ . Адрес кратен 16, то есть сегмент DS начинается с адреса 48BD0h.

Расчёт для DS:  $48BDh \cdot 10h + 0018h = 48BD0h + 0018h = 48BE8h$ . Адрес не кратен 16, то есть сегмент CS начинается с адреса 48BF0h.



**3. Почему перед началом выполнения первой программы содержимое регистра DS в точности на 10h меньше содержимого регистра SS? (Сравниваются данные из первой строки таблицы трассировки)**

Потому что сегменты расположены последовательно и выровнены по 0x10, сегмент стека весит 0x100 байт, а в сегментных регистрах адреса памяти хранятся без разрядов, отвечающих за смещение.

**4. Из таблицы трассировки к первой программе выпишите машинные коды команд `mov AX,data` и `mov AH,09h`. Сколько места в памяти в байтах они занимают? Почему у них разный размер?**

1 `mov AX, data` - B8BD48

Размер машинного кода:

0003 - 0000 = 3 байта

1 `mov AH, 09h` - B409

Размер машинного кода:

0005 - 0003 = 2 байта

Размер команд разный, потому что в первом случае адрес сегмента данных, а во втором - код функции DOS.

**5. Из таблицы трассировки ко второй программе выпишите базовые адреса сегментов (значение DS при этом нам нужно взять после инициализации). При использовании модели `small` сегмент кода располагается в памяти первым. Убедитесь в этом. (Если это не так, значит, вы неверно заполнили таблицу трассировки.)**

— DS = 48AF

— SS = 48B1

— CS = 48AD

Сегменты в модели `small` расположены в следующем порядке: CS, DS, SS.

CS:  $48ADh \cdot 16 + 0011h = 48AD0h + 0011h = 48AE1h$ . Адрес не кратен 16, округляем вверх: 48AF0h. Значит сегмент DS начинается с адреса 48AFh.

DS:  $48AFh \cdot 16 + 0018h = 48AF0h + 0018h = 48B08h$ . Адрес не кратен 16, округляем вверх: 48B10h, значит сегмент SS начинается с адреса 48B1h.

**6. Сравните содержимое регистра SP в таблицах трассировки для программ 2 и 3. Объясните, почему получены эти значения.**

В программе 2 регистр SP равен 0100, а в программе 3 - FFFE.

В программе 2 (.model small) регистр SP равен 0100h, потому что была явно указана директива .stack 100h, и загрузчик установил указатель стека в верхнюю часть сегмента стека (размер 256 байт -> SP = 0100h).

В программе 3 (.model tiny) регистр SP равен FFFh, потому что это COM-программа, где стек размещается в самом низу, инициализируясь максимальным значением, и имеет адрес (FFFh).

### **7. Какие операторы называют директивами ассемблера? Приведите примеры директив.**

Директивы передают ассемблеру метаданные, необходимые для создания объектного и исполняемого файлов или же листинга.

Директива ASSUME передаёт ассемблеру информацию о соответствии между адресами сегментных регистров и программными сегментами. Директива имеет следующий формат:

```
1 assume <пара>[[, <пара>]]  
2 assume nothing
```

где <пара> - это <сегментный регистр> :<имя сегмента> либо <сегментный регистр> :NOTHING Например, директива

```
1 assume es:a, ds:b, cs:c
```

сообщает ассемблеру, что для сегментирования адресов из сегмента А выбирается регистр ES, для адресов из сегмента В – регистр DS, а для адресов из сегмента С – регистр CS.

### **8. Зачем в последнем предложении end указывают метку, помечающую первую команду программы?**

Программа на языке ассемблера состоит из программных модулей, содержащихся в различных файлах. Каждый модуль, в свою очередь, состоит из инструкций процессора или директив ассемблера и заканчивается директивой END. Метка, стоящая после кода псевдооперации END, определяет адрес, с которого должно начаться выполнение программы и называется **точкой входа в программу**.

Каждый модуль также разбивается на отдельные части **директивами сегментации**, определяющими начало и конец сегмента. Каждый сегмент начинается директивой начала сегмента – SEGMENT и заканчивается директивой конца сегмента – ENDS . В начале директив ставится имя сегмента.

Таким образом, метка, указанная в END определяет начальный адрес, с которого процессор должен начать выполнение команды.

### 9. Как числа размером в слово хранятся в памяти и как они заносятся в 2-ух байтовые регистры?

В зависимости от архитектуры процессора, применяется прямой или обратный порядок байт. Почти во всех современных процессорах байты с меньшим адресом считаются младшими, такой порядок называется Little Endian. То есть 2 байта ложатся в 2 байта и сначала байт с младшими битами числа, затем байт со старшими. В случае с Big Endian (прямым порядком байт) ситуация обратная, сначала идут старшие разряды: так, как мы привыкли записывать числа на бумаге.

### 10. Как инициализируются в программе выводимые на экран текстовые строки?

Выводимые на экран текстовые строки инициализируются в секции .data с помощью db, строка должна оканчиваться знаком \$.

Прежде чем делать int 21h нужно в DX положить адрес начала строки

```
1 mov dx, offset имя_метки_с_которой_начинается_строка
```

### 11. Что нужно сделать, чтобы обратиться к DOS для вывода строки на экран? Как DOS определит, где строка закончилась?

Вывод на экран строки текста и выход из программы осуществляются путем вызова стандартных процедур DOS, называемых **прерываниями**. Прерывания под кодом 21h (33 – в десятичной системе счисления) называются **функциями DOS**, у них нет названий, а только идентификаторы. Номер прерывания и его параметры передаются в регистрах процессора, при этом номер должен находиться в регистре АХ. Так, например, прерывание INT 21h, с помощью которого на экран выводится строка символов, управляется двумя параметрами: в регистре АХ должно быть число 9, а в регистре DX – адрес строки символов, оканчивающейся знаком '\$': это спецсимвол языка ассемблера, которым обозначается нулевой байт (так называемый NUL-символ). Адрес строки Hello загружается в регистр DX с помощью оператора OFFSET (смещение):

```
1 offset имя
```

Выход из программы осуществляется через функцию DOS с номером 4Ch. Эта функция предполагает, что в регистре AL находится код завершения программы, который она передаст DOS. Если программа завершилась успешно,

код завершения должен быть равен 0, поэтому в примере загружаем регистры AH и AL с помощью одной команды `MOV ax,4C00h`, после чего вызываем прерывание 21h.

**12. Программы, которые должны исполняться как .EXE и .COM, имеют существенные различия по:**

- размеру
- сегментной структуре
- механизму инициализации

EXE-программы содержат несколько программных сегментов, включая сегмент кода, данных и стека. EXE-файл загружается, начиная с адреса `PSP:0100h`. В процессе загрузки считывается информация EXE-заголовка в начале файла, при помощи которого загрузчик выполняет настройку ссылок на сегменты в загруженном модуле, чтобы учесть тот факт, что программа была загружена в произвольно выбранный сегмент. После настройки ссылок управление передается загрузочному модулю к адресу `CS:IP`, извлеченному из заголовка EXE.

COM-программы содержат единственный сегмент (или, во всяком случае, не содержат явных ссылок на другие сегменты). Образ COM-файла считывается с диска и помещается в память, начиная с `PSP:0100h`, в связи с этим COM-программа должна содержать в начале сегмента кода директиву позволяющую осуществить такую загрузку (`ORG 100h`). Они быстрее загружаются, ибо не требуется перемещения сегментов, и занимают меньше места на диске, поскольку EXE-заголовок и сегмент стека отсутствуют в загрузочном модуле.