МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

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

Кафедра дискретной математики и информатики

ЛАБОРАТОРНАЯ РАБОТА №1

студента 2 курса 221 группы

направления 09.03.01 Информатика и вычислительная техника

факультета компьютерных наук и информационных технологий

Мусатова Федора Алексеевича

Саратов 2025

**Задание 1.1.** 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.

**Текст 1 программы**

stak segment stack 'stack' ;Начало сегмента стека

db 256 dup (?) ;Резервируем 256 байт для стека

stak ends ;Конец сегмента стека

data segment 'data' ;Начало сегмента данных

Names db 'Musatov Fedor, 221$' ;Строка для вывода

data ends ;Конец сегмента данных

code segment 'code' ;Начало сегмента кода

assume CS:code,DS:data,SS:stak ;Сегментный регистр CS будет указывать на сегмент команд,

;регистр DS - на сегмент данных, SS – на стек

start: ;Точка входа в программу start

;Обязательная инициализация регистра DS в начале программы

mov AX,data ;Адрес сегмента данных сначала загрузим в AX,

mov DS,AX ;а затем перенесем из AX в DS

mov AH,09h ;Выводит строку, адрес которой в DX

mov DX,offset Names ;Адрес начала строки (переменной Names) записывается в регистр DX

int 21h ;Вызов функции DOS

mov AX,4C00h ;Функция 4Ch завершения программы с кодом возврата 0

int 21h ;Вызов функции DOS

code ends ;Конец сегмента кода

end start ;Конец текста программы с точкой входа

**Текст 2 программы**

model small ;Модель памяти SMALL использует сегменты

;размером не более 64Кб

.stack 100h ;Сегмент стека размером 100h (256байт)

.data ;Начало сегмента данных

Names db 'Musatov Fedor, 221$'

.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

mov AH,09h ;Вывод строки из регистра DX

mov DX,offset Names ;Адрес начала строки (переменной Names) записывается в регистр DX

int 21h

mov AX,4C00h ;Функция 4Ch завершения программы с кодом возврата 0

int 21h ; Прерывание, которое вызывает функции

end start

**Текст 3 программы**

.model tiny ;Модель памяти TINY, в которой код, данные и стек

;размещаются в одном и том же сегменте размером до 64Кб

.code ;Начало сегмента кода

org 100h ;Устанавливает значение программного счетчика в 100h

;Начало необходимое для COM-программы,

;которая загружается в память с адреса PSP:100h

start:

mov AH,09h ;Вывод строки из регистра DX

mov DX,offset Names;Адрес начала строки (переменной Names) записывается в регистр DX

int 21h

mov AX,4C00h;Функция 4Ch завершения программы с кодом возврата 0

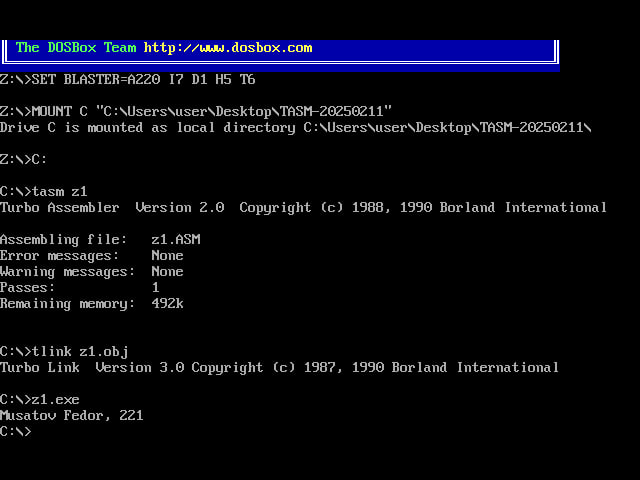
int 21h

;===== Data =====

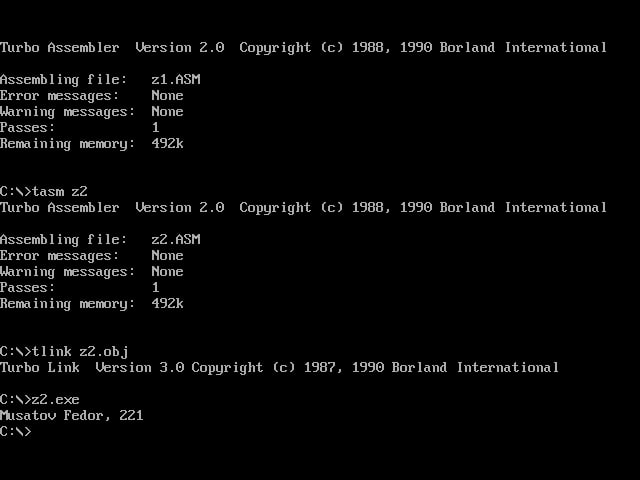
Names db 'Musatov Fedor, 221$'

end start

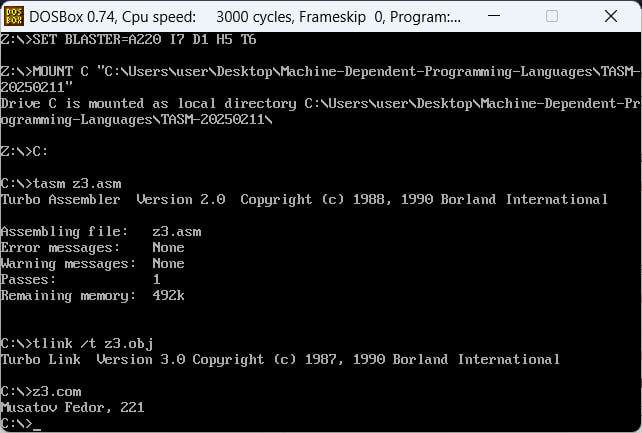
**Скриншот с запуска 1 программы**



**Скриншот с запуска 2 программы**



**Скриншот с запуска 3 программы**



**Тексты 2-х командных файлов (1 – для exe программы, 2 – для com программы)**

**1.**

cls

tasm.exe %1.asm

tlink.exe /x %1.obj

%1

**2.**

cls

tasm.exe %1.asm

tlink.exe /x /t %1.obj

%1

**Задание 2**

**Текст задания**

Заполните таблицы трассировки для 3-х программ.

**Таблица 1**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | 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 |
| 8 | 0192 | 000С | F715 | 098D | 0106 | 2110 | 0192 | 0000 | 0000 | | 10100011 |

**Таблица 2**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | Машинный код | Команда | Регистры | | | | | | | | | Флаги |
| AX | BX | CX | DX | SP | DS | SS | CS | IP |  |
| 1 | B8AF48 | 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) | 48BD | 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 |
| 8 | 0192 | 000C | F715 | 098D | 0106 | 2110 | 0192 | 0000 | 0000 | 10100011 |

**Таблица 3**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | 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 | BA0C01 | 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 |
| 6 | 0192 | 000C | F715 | 098D | 0106 | 2110 | 0192 | 0000 | 0000 | 10100011 |

**Ответы на контрольные вопросы**

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

Адрес начала сегмента (Содержащиеся в регистрах CS, DS, SS или ES 16-битовые значения)

1. **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 0011

DATA 0013

STAK 0100

Регистры после инициализации DS:

DS = 48BD

SS = 48AD

CS = 48BF

Сегменты программы в памяти расположились таким образом: SS, DS, CS

Так как физический адрес – это базовый адрес, умноженный на 10 и прибавленный к размеру, получим, что расширенный адрес сегмента

SS = 48AD \* 0010h + 0100h = 48BD0 (кратно 16) = 48BD = DS

DS = 48BD \* 0010h + 0015h = 48BE5 (не кратно 16) = 48BF = CS

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

При инициализации DS и SS, первоначально DS указывает на адрес начала префикса, размер которого 100h. После инициализации DS меняется и перемещается на место после сегмента стека. Таким образом физический адрес отличается на 100h, а базовый на 10h

48AD – 489D = 10h – искомое смещение

1. **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?**

Машинный код команды mov AX,data: B8 BD 48

Машинный код команды mov AH,09h: B409

Первая команда занимает в памяти 3 байта

Вторая команда занимает в памяти 2 байта

Операнд - константа 09h, всегда занимает 1 байт, а вся команда занимает фиксированное количество памяти – 2 байта. Команда mov AX,data использует более сложный формат записи и операнд, который может занимать больше места, в то время как команда mov AH,09h использует более простой формат записи, операнд. Все это приводит к тому, что вторая команда занимает меньше места в памяти.

1. **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 расположились в следующем порядке: CS, DS, SS. CS = 48AD ⋅ 10ℎ + 0011(code size) = 48𝐴𝐷0 + 0011 = 48𝐴𝐸1 = 48𝐴𝐹0 (мы округлили, так как было не кратно 16, теперь установили кратность и значит можно отбросить ноль в конце) = 48𝐴𝐹 = 𝐷𝑆.

𝐷𝑆 = 48𝐴𝐹 ⋅ 10ℎ + 0015 = 48𝐴𝐹0 + 0012 = 48𝐵02 = 48𝐵1 = 𝑆𝑆, в чём и следовало убедиться.

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

В программе 2, которая является моделью SMALL, память под стек выделяется командой “.stack 0100h”, поэтому его регистр будет 0100h. В 3 программе (модель TINY) сегменты стека, данных и кода располагаются в одном сегменте размером до 64 Кб, в этом случае он указывает на вершину (самый большой адрес), то есть на FFFE (он инициализируется максимальным возможным значением, поскольку он будет сдвигаться при помещении в него данных.)

1. **Какие операторы называют директивами ассемблера? Приведите примеры директив.**

Указывают программе ассемблеру, каким образом следует объединять инструкции для создания модуля, который и станет работающей программой.

Пример: директива END, SEGMENT

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

Метка, стоящая после кода псевдооперации END, определяет адрес, с которого должно начаться выполнение программы и называется ***точкой входа в программу***

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

|  |  |
| --- | --- |
| 9B | 00 |

Числа размером в слово хранятся в памяти в "перевернутом" виде: младшие (правые) 8 битов числа размещаются в первом байте слова, а старшие 8 битов - во втором байте (в 16-ричной системе: две правые цифры - в первом байте, две левые цифры - во втором байте). Например, число 155 (=009Вh) в виде слова хранится в памяти так:

Однако в регистрах числа хранятся в обычном виде:

|  |  |  |
| --- | --- | --- |
| AX | 00 | 9B |
|  | AH | AL |

1. **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 $.

**mov** **DX,** offset Name (где Name – название метки строки)

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

Для того, чтобы обратиться к DOS для вывода строки на экран, нужно сделать **прерывание.** Прерывания под номером 21h называются функциями DOS, у них нет названий, а только номера. Номер прерывания и его параметры передаются в регистрах процессора, при этом номер должен находиться в регистре AH. Адрес строки Name загружается в регистр DX с помощью оператора offset (смещение):

**Offset Name**



Так, например, прерывание INT 21h, с помощью которого на экран выводится строка символов, управляется двумя параметрами: в регистре AH должно быть число 9, а в регистре DX – адрес строки символов, оканчивающейся знаком '$'.

1. **Программы, которые должны исполняться как .EXE и .COM, имеют существенные различия по:**
   1. размеру,
   2. сегментной структуре,
   3. механизму инициализации.
2. **Размер:**

COM-программа меньше по размеру, так как в отличие от EXE-программы она не содержит заголовок и сегмент стека (это касается малых программ).

1. **Сегментная структура:**

COM -программы в отличие от EXE - программы, которая содержит отдельные сегменты данных, стека и кода, содержит лишь один основной сегмент (сегмент кода), в котором размещаются код, данные и стек (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).

1. **Механизм инициализации:**

COM-программа просто загружается в память, а с EXE-программы сначала считывается заголовок, 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.