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

Карасева Вадима Дмитриевича

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.

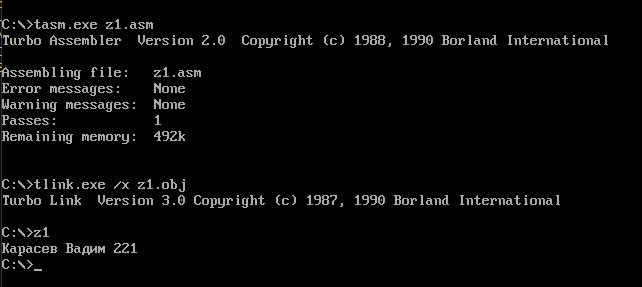
Для того, чтобы в DOSBox корректным образом отображались русские символы, необходимо сделать несколько операций.

Во-первых, необходимо убедиться, что исходный файл .asm пишется в кодировке Cyrillic OEM 866. Затем, перед запуском программы, нужно прописать в DOSBox следующую команду:

keyb ru 866

Ниже представлены коды всех 3-х программ с результатами вывода:

**Программа 1**



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

Hello db 'Карасев Вадим 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 ;Функция DOS 9h вывода на экран

mov DX,offset Hello ;Адрес начала строки 'Hello, World!' 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

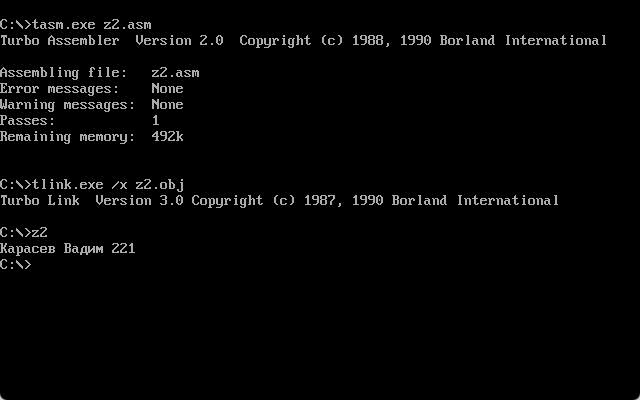
code ends ;Конец сегмента кода

end start ;Конец текста программы с точкой входа

z1.bat

cls   
tasm.exe %1.asm  
tlink.exe /x %1.obj   
%1

**Программа 2**



.model small ;Модель памяти SMALL использует сегменты

;размером не более 64Кб

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

.data ;Начало сегмента данных

Hello db 'Карасев Вадим 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

mov DX,offset Hello

int 21h

mov AX,4C00h

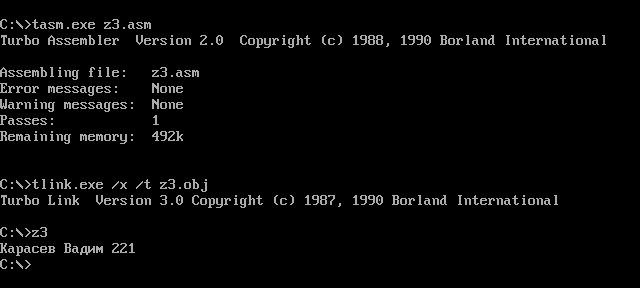
int 21h

end start

z2.bat

cls   
tasm.exe %1.asm  
tlink.exe /x %1.obj   
%1

**Программа 3**



.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

mov DX,offset Hello

int 21h

mov AX,4C00h

int 21h

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

Hello db 'Карасев Вадим 221$'

end start

z3.bat

cls   
tasm.exe %1.asm  
tlink.exe /x /t %1.obj   
%1

**Задание 1.2.** Заполните таблицы трассировки для 3-х программ.

**Программа 1**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Шаг | Машинный код | Команда | Регистры | | | | | | | | Флаги | |
|  | | | | | | | |  | |
| AX | BX | CX | DX | SP | DS | SS | CS | IP | CZSOPAID |
| 0 | B8BD48 | Mov, ax, 48BD | 0000 | 0000 | 0000 | 0000 | 0100 | 489D | 48AD | 48BF | 0000 | 00000010 |
| 1 | 8ED8 | Mov, ds, ax | 48BD | 0000 | 0000 | 0000 | 0100 | 489D | 48AD | 48BF | 0003 | 00000010 |
| 2 | B409 | Mov, ah, 09 | 48BD | 0000 | 0000 | 0000 | 0100 | 48BD | 48AD | 48BF | 0005 | 00000010 |
| 3 | BA0000 | Mov, dx, 0000 | 09BD | 0000 | 0000 | 0000 | 0100 | 48BD | 48AD | 48BF | 0007 | 00000010 |
| 4 | CD21 | Int, 21 | 09BD | 0000 | 0000 | 0000 | 0100 | 48BD | 48AD | 48BF | 000A | 00000010 |
| 5 | B8004C | Mov, ax, 4c00 | 09BD | 0000 | 0000 | 0000 | 0100 | 48BD | 48AD | 48BF | 000C | 00000010 |
| 6 | CD21 | Int, 21 | 4C00 | 0000 | 0000 | 0000 | 0100 | 48BD | 48AD | 48BF | 000F | 00000010 |
| 7 | Выход программы с кодом 0 | | 0192 | 000B | F715 | 098D | 0106 | 2110 | 0192 | 0000 | 0000 | 10100011 |

**Программа 2**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Шаг | Машинный код | Команда | Регистры | | | | | | | | | Флаги |
|  | | | | | | | | |  |
| AX | BX | CX | DX | SP | DS | SS | CS | IP | CZSOPAID |
| 0 | B8BD48 | Mov, ax, 48BD | 0000 | 0000 | 0000 | 0000 | 0100 | 489D | 48B1 | 48AD | 0000 | 00000010 |
| 1 | 8ED8 | Mov, ds, ax | 48AF | 0000 | 0000 | 0000 | 0100 | 489D | 48B1 | 48AD | 0003 | 00000010 |
| 2 | B409 | Mov, ah, 09 | 48AF | 0000 | 0000 | 0000 | 0100 | 48AF | 48B1 | 48AD | 0005 | 00000010 |
| 3 | BA0000 | Mov, dx, 0000 | 09AF | 0000 | 0000 | 0000 | 0100 | 48AF | 48B1 | 48AD | 0007 | 00000010 |
| 4 | CD21 | Int, 21 | 09AF | 0000 | 0000 | 0000 | 0100 | 48AF | 48B1 | 48AD | 000A | 00000010 |
| 5 | B8004C | Mov, ax, 4c00 | 09AF | 0000 | 0000 | 0000 | 0100 | 48AF | 48B1 | 48AD | 000C | 00000010 |
| 6 | CD21 | Int, 21 | 4C00 | 0000 | 0000 | 0000 | 0100 | 48AF | 48B1 | 48AD | 000F | 00000010 |
| 7 | Выход программы с кодом 0 | | 0192 | 000B | F715 | 098D | 0106 | 2110 | 0192 | 0000 | 0000 | 10100011 |

**Программа 3**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Шаг | Машинный код | Команда | Регистры | | | | | | | | Флаги | |
|  | | | | | | | |  | |
| AX | BX | CX | DX | SP | DS | SS | CS | IP | CZSOPAID |
| 0 | B409 | Int, ah,09 | 0000 | 0000 | 0000 | 0000 | FFFE | 489D | 489D | 489D | 0100 | 00000010 |
| 1 | BA0C01 | Mov, dx, 010C | 0900 | 0000 | 0000 | 0000 | FFFE | 489D | 489D | 489D | 0102 | 00000010 |
| 2 | CD21 | Int, 21 | 0900 | 0000 | 0000 | 010C | FFFE | 489D | 489D | 489D | 0105 | 00000010 |
| 3 | B8004C | Mov, ac, 4C00 | 0900 | 0000 | 0000 | 010C | FFFE | 489D | 489D | 489D | 0107 | 00000010 |
| 4 | CD21 | Int, 21 | 4C00 | 0000 | 0000 | 010C | FFFE | 489D | 489D | 489D | 010A | 00000010 |
| 5 | Выход программы с кодом 0 | | 0192 | 000B | F715 | 098D | 0106 | 2110 | 0192 | 0000 | 0000 | 10100011 |

**Задание 1.3.** Ответить на контрольные вопросы.

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

Сегментными адресами называются 16-ти битовые значения, хранящиеся в сегментных регистрах, таких как DS, CS, SS, ES. Исходя из этого, сегментным адресом является адрес начала сегмента.

**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.)**

Из файла .lst: code = 0011, data = 0012, stak = 0100. Из трассировки программы: DS = 48BD, CS = 48BF, SS = 48AD. Исходя из трассировки, сегменты программы расположились в памяти в следующем порядке: стек -> данные -> код.

Операции:

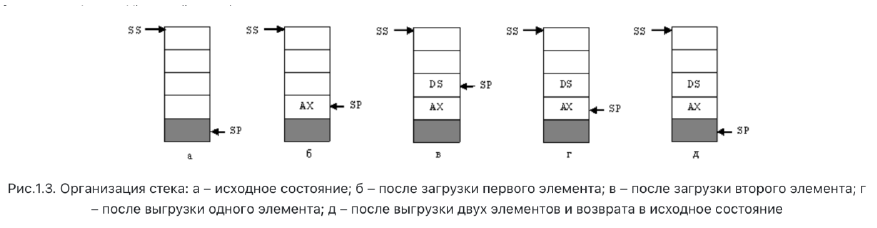
1. Для стека: представим базовый адрес в двоичном коде, расширим до физического и прибавим размер: 1000100010110100\_0000000000000000 + 1000000000000000 = 1000100011110100\_0000000000000000 = 297936, что в шестнадцатеричной системе даёт 48BD — адрес следующего сегмента.
2. Для данных: представим базовый адрес в двоичном коде, расширим до физического и прибавим размер: 1000100010111011\_0000000000000000 + 1000000000000000 = 1000100011111011\_0000000000000000 = 297954, что при округлении до делимого на 16 (297968) и при переводе в шестнадцатеричную даёт 48BF — адрес следующего сегмента.

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

Чтобы обосновать данное явление, обратимся к теории стеков:

**Стек** – это область памяти для временного хранения данных, в которую по специальным командам можно записывать отдельные слова (но не байты).

Элементы стека располагаются в памяти начиная со дна стека (т.е. с его максимального адреса), по последовательно уменьшающимся адресам.



Адрес верхнего, доступного элемента хранится в регистре-указателе стека SP. Как и любая другая область памяти программы, стек должен входить в какой-то сегмент или образовывать отдельный сегмент. Сегментный адрес этого сегмента помещается в регистр SS. Таким образом, пара регистров SS:SP описывает адрес доступной ячейки стека. В исходном состоянии указатель стека SP указывает на ячейку, лежащую под дном стека и не входящую в него (на рис.1.3. (а) данная ячейка обозначена серым цветом).

Для сохранения и восстановления различных 16-битовых данных в стеке используются команды PUSH (протолкнуть) и POP (вытолкнуть). За кодами операций PUSH и POP следует операнд, который необходимо поместить (извлечь) в (из) стек. В качестве операнда может выступать регистр или ячейка памяти.

Команда PUSH сначала уменьшает на 2 содержимое указателя стека, а затем помещает операнд по адресу, находящемуся в SP.

Исходя из вышесказанного, можно сделать вывод, что разница в 10h (16 байт) между DS и SS объясняется тем, что сегмент данных находится в памяти ниже стека.

**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?**

mov ax,48BD - B8BD48 - 3 байта

mov ah,09 - B409 – 2 байта (по две цифры на байт, 1 шестнадцатеричная цифра кодирует 4 бита)

Команда mov AX,data загружает 16-ти битное значение в регистр AX, а mov AH,09h 8 битное значение в регистр AH. Этот факт и объясняет разницу в размере двух этих команд. Так как команда для 16-битного регистра требует больше места, чтобы указать полное 16-битное значение, тогда как команда для 8-битного регистра занимает меньше места.

**Из таблицы трассировки ко второй программе выпишите базовые адреса сегментов (значение DS при этом 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.)**

- Базовый адрес DS (после инициализации) = 48AF;

- Базовый адрес SS = 48B1;

- Базовый адрес CS = 48AD.

Так как CS имеет наименьшее числовое значение, то, следовательно, сегмент кода располагается в памяти первым.

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

Полученные значения можно объяснить тем, что во второй программе явно указывается, что под стек выделяется 256 байт (строка .stack 100h), в то время как в третьей программе это значение явно не указывается, так как в СОМ-программах обычно стек инициализируется на адресе, который равен верхней части сегмента (FFFF) и уменьшается/увеличивается на 2 по мере выполнения программы. Исходя из этого, значение указателя SP изначально равняется FFFE.

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

Директивами называются операторы, которые указывают программе ассемблера, каким образом следует объединять инструкции для создания модуля, который станет работающей программой. Примером директивы может служить слово ASSUME, которое позволяет явно указывать, какой сегментный регистр на какой сегмент указывает. Помимо этого, примером может быть слово END, которое явно указывает, в каком месте заканчивается конкретный модуль программы.

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

Метка, стоящая после кода псевдооперации END, определяет адрес, с которого должно начинаться выполнение программы. Этот адрес называется точкой входа в программу.

**Как числа размером в слово хранятся в памяти и как они заносятся в двухбайтовые регистры?**

Группа из восьми битов называется байтом и представляет собой наименьшую адресуемую единицу — ячейку памяти. Двухбайтовое поле образует шестнадцатиразрядное машинное слово, или Word. В слове биты нумеруются справа налево от 0 до 15, причем байт с меньшим адресом считается младшим. Символьная информация кодируется согласно кодировке ASCII, числовые данные — согласно двоичной арифметике, а отрицательные числа представляются в виде дополнительного кода. Для того чтобы занести слово в двухбайтовые регистры, можно воспользоваться командой MOV.

**Как инициализируются в программе выводимые на экран текстовые строки?**

Предположим, необходимо вывести на экран строку "Hello world!". Для этого в программе сначала нужно её инициализировать. Делается это так:

data segment 'data'

Hello db "Hello world!$"

data ends

Сначала идёт описание сегмента с данными (первая строка). Далее указывается имя переменной (в данном случае — `Hello`). Затем, через ключевое слово `db`, указывается значение этой переменной (в данном случае — строка "Hello world!").

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

Для начала, с помощью цепочки записей необходимо записать data (вместо data — актуальное название сегмента данных) в регистр DS. За вывод строки на экран отвечает DOS-команда `int 09h`. Её необходимо записать в регистр AH. Затем в регистр DX записывается адрес начала строки (offset Hello, где Hello — актуальное имя переменной). Все эти шаги необходимы, потому что в дальнейшем будет вызываться функция int 21h («Активирование команды DOS»), которая работает, опираясь на значения в регистрах DX и AH.

Для того чтобы DOS «понимал», где заканчивается строка, программисту необходимо явно указать знак доллара в конце строки.

**Программы, которые должны исполняться как .EXE и .COM, имеют существенные различия по размеру, сегментной структуре и механизму инициализации. Охарактеризуйте каждый из перечисленных пунктов.**

- Размер: Если размер имеет доминирующую важность в реализации программы, COM-программы будут предпочтительнее, так как они весят меньше. Это обусловлено отсутствием EXE-заголовка и сегмента стека в загрузочном модуле.

- Сегментная структура: EXE-программы содержат несколько программных сегментов, включая сегменты данных, регистра и стека. COM-программы, в свою очередь, содержат единственный сегмент (или, по крайней мере, не имеют явных ссылок на другие сегменты).

- Механизм инициализации: EXE-файл загружается с адреса PSP:0100h. В процессе загрузки считывается информация из EXE-заголовка файла с целью выполнения настройки ссылок на сегменты в загруженном модуле. Это позволяет учитывать факт, что программа загружена в произвольно выбранный сегмент. После того как ссылки были загружены, управление передается загрузочному модулю к адресу CS:IP, также извлеченному из заголовка EXE.

В случае COM-программ, образ COM-файла считывается с диска и помещается в память, начиная с адреса PSP:0100h. В связи с этим программа должна содержать в начале сегмента директиву, которая позволит реализовать такую загрузку (org 100h).