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

Лабораторная работа
№1

Выполнил
студент 211 группы
факультета КНиИТ
Коновалов Александр

2024г.

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

Предварительно, изменим программы и сохраним их в кодировке 866
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
Name db 'Коновалов Александр 211$'      ;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 Name              ;Адрес начала строки 'Коновалов Александр 211' 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
```

2)

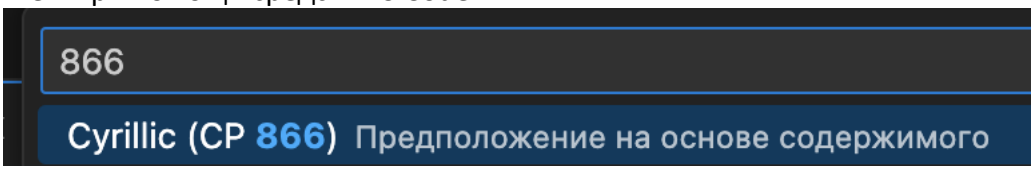
```
.model small                    ;Модель памяти SMALL использует сегменты
                                ;размером не более 64Кб
.stack 100h                     ;Сегмент стека размером 100h (256 байт)
.data                           ;Начало сегмента данных
Name db 'Коновалов Александр 211$'
.code                           ;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 Name
int 21h
mov AX,4C00h
int 21h
end start
```

3)

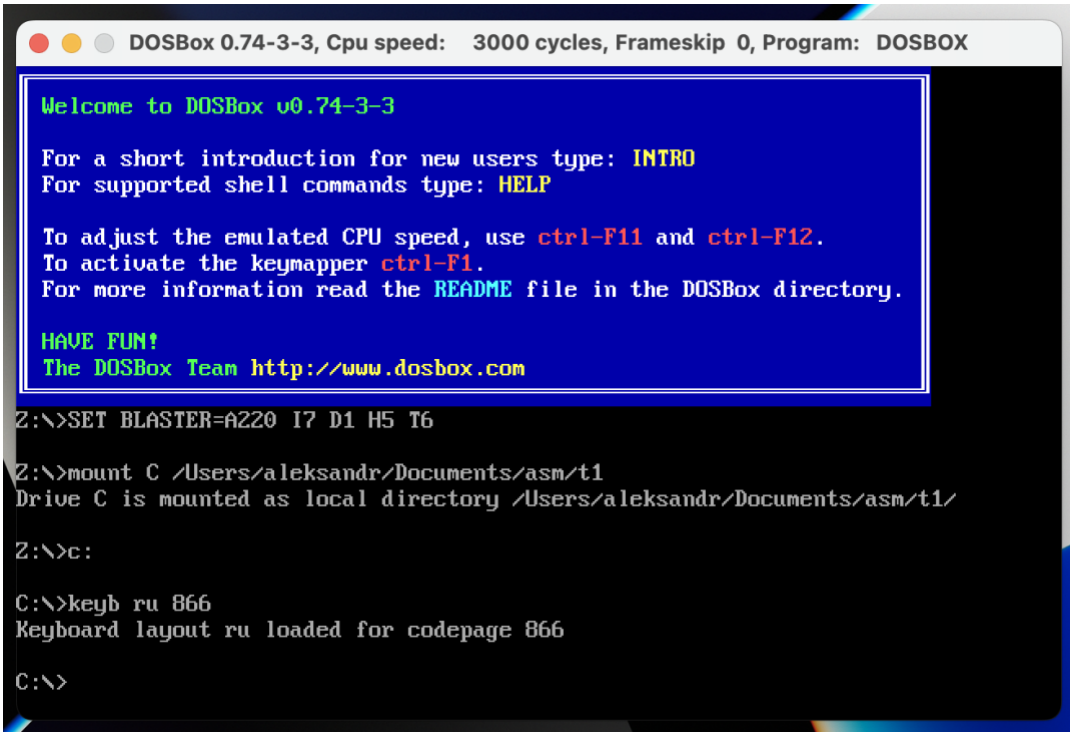
```
.model tiny            ;Модель памяти TINY, в которой код, данные и стек
                       ;размещаются в одном и том же сегменте размером до 64Кб
.code                  ;Начало сегмента кода
org 100h               ;Устанавливает значение программного счетчика в 100h
                       ;Начало необходимое для COM-программы,
                       ;которая загружается в память с адреса PSP:100h

start:
mov AH,09h
mov DX,offset Name
int 21h
mov AX,4C00h
int 21h
;===== Data =====
Name db 'Коновалов Александр 211$'
end start
```

И сохраняем при помощи средств VS Code:



Запуск программ:
Для начала смонтируем образ к пути директории, в которой находятся файлы .asm
Укажем для DosBox нужную нам кодировку 866 (для вывода кириллицы)



Создадим командные файлы make.bat и makec.bat(для com-программ) по примеру из лекции

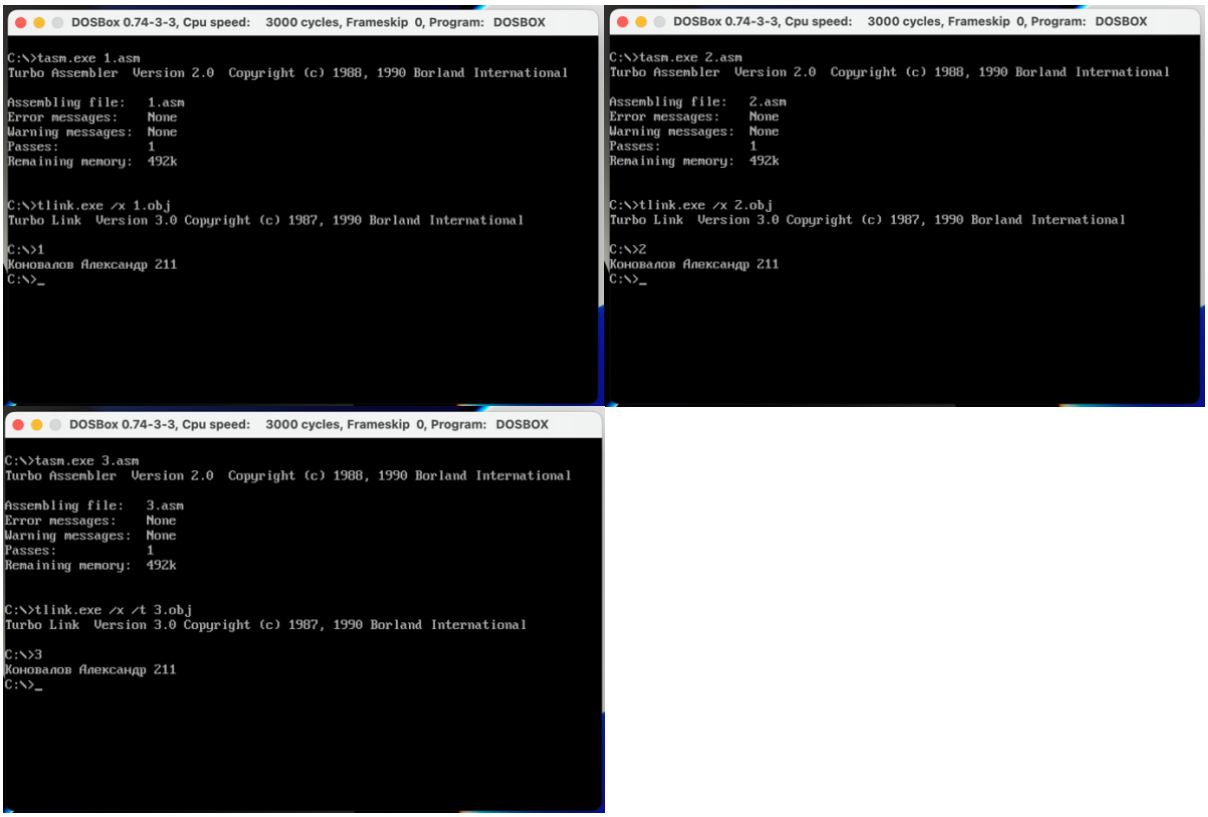
make.bat

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

makec.bat

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

Запустим полученные программы при помощи командных файлов. Получаем вывод:



Задание 1.2.

Пользуясь схематическим представлениемTD, заполним таблицы трассировки данными из отладчик пакета TASM (TD - Turbo Debugger).

<table>
<tr><td>Окно кода.
В нем отображаются команды, их адреса и машинные коды.</td><td>Окно значений регистров</td><td>Окно значений битов регистра FLAGS</td></tr>
<tr><td>Окно данных.
Отображает содержимое сегмента данных.</td><td colspan="2">Окно стека.
Отображает содержимое стека</td></tr>
</table>

Получаем следующие таблицы:

1			Регистры									Флаги
шаг	машинный код	команда	AX	BX	CX	DX	SP	DS	SS	CS	IP	CZSOPAID
	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
Выход из программы			0192	000B	F711	098D	0106	2110	0192	0000	0000	10100011

2			Регистры									Флаги
шаг	машинный код	команда	AX	BX	CX	DX	SP	DS	SS	CS	IP	CZSOPAID
	B8AF48	mov ax,48AF	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
Выход из программы			0192	000B	F711	098D	0106	2110	0192	0000	0000	10100011

3			Регистры									Флаги
шаг	машинный код	команда	AX	BX	CX	DX	SP	DS	SS	CS	IP	CZSOPAID
	B409	mov 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 ax,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
Выход из программы			0192	000B	F711	098D	0106	2110	0192	0000	0000	10100011

Контрольные вопросы

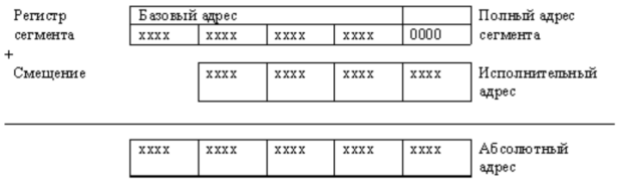
1. **Что такое сегментный (базовый) адрес?**
   Сегментные регистры имеют размер в 16 разрядов, а содержащиеся в этих регистрах (CS , DS , SS или ES) 16-битовые значения называются **базовым адресом сегмента**.

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.)**

   Groups & Segments Size

   CODE 0011
   DATA 0018
   STAK 0100

   CS 48BF – 3 порядок
   DS 48BD – 2 порядок
   SS 48AD – 1 порядок



   Пользуясь данным правилом, выполним проверку(в 16 СС):

   SS:
   48AD * 10 = 48AD0 (сдвинули на 4 бита влево)
   48AD0 + 100 = 48BD0
   DS:
   48BD * 10 = 48BD0
   48BD0 + 18 = 48BE8 округлим (48BF0)

   По итогам расчётов, можем сделать вывод, что таблица трассировки заполнена верно.

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)**

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

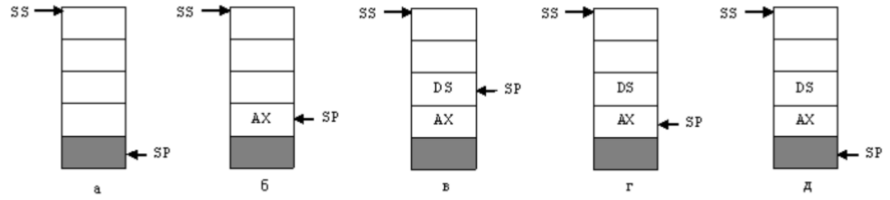


Рис.1.3. Организация стека: а – исходное состояние; б – после загрузки первого элемента; в – после загрузки второго элемента; г – после выгрузки одного элемента; д – после выгрузки двух элементов и возврата в исходное состояние

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 эта 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, делаем вывод, что разница в 10h (16 байт) между DS и SS объясняется тем, что сегмент данных находится в памяти ниже стека.

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?**

   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.

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.)**

   CS 48AD - 1

DS 48AF - 2
SS 48B1 – 3

Groups & Segments	Size
STACK	0100
_DATA	0018
_TEXT	0011

Проверка
CS:
48AD * 10 = 48AD0
48AD0 + 11 = 48AE1(округл до 48AF0)
DS:
48AF * 10 = 48AF0
48AF0 + 18 = 48B08 (округл до 48B10)

Вывод: таблица заполнена верно, действительно сегмент кода CS располагается в памяти первым.

6. **Сравните содержимое регистра SP в таблицах трассировки для программах 2 и 3. Объясните, почему получены эти значения.**
   2) 0100
   3) FFFE
   В данном случае мы рассматриваем два разных типа программ: EXE и COM
   Обращаясь к теории, можно заметить, что
   Для EXE
   В самом начале мы резервируем память для стека:

```
.model small                ;Модель памяти SMALL использует сегменты
                            ;размером не более 64Кб
.stack 100h                 ;Сегмент стека размером 100h (256 байт)
```

   “Сегмент стека в приведенной структуре описан с параметром STACK, поэтому в самой программе нет необходимости загружать сегментный регистр SS. Сегментный регистр CS тоже нет необходимости загружать, как уже отмечалось ранее. В связи с этим в начале программы загружается лишь регистр DS.”

   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).

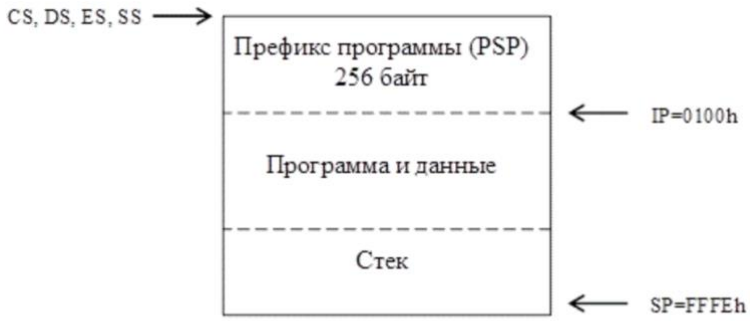


Рис.1.6. Образ памяти программы .COM.

Получается, что для разных типов программ память распределяется разным образом.

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

Программа на языке ассемблера состоит из операторов и директив. Операторы – это инструкции, управляющие работой процессора, а директивы указывают программе ассемблеру, каким образом следует объединять инструкции для создания модуля, который и станет работающей программой.
Структура директивы аналогична структуре инструкции. Второе поле – код псевдооперации определяет смысловое содержание директивы. Как и у инструкции, у директивы есть операнды, причем их может быть один или несколько и они отделяются друг от друга запятыми. Допустимое число операндов в директиве определяется кодом псевдооперации
Директива может быть помечена символическим именем и содержать поле комментария. Символическое имя, стоящее в начале директивы распределения памяти, называется ***переменной***. В отличие от метки команды после символического имени директивы двоеточие не ставиться. Комментарий записывается также как и в случае команды и может занимать целую строку.

Комментарии вносятся в программу как поясняющий текст и могут содержать любые знаки до ближайшего символа конца строки. Для создания комментариев, занимающих несколько строк, может быть использована директива COMMENT.

Каждый сегмент начинается директивой начала сегмента – SEGMENT и заканчивается директивой конца сегмента – ENDS . В начале директив ставится имя сегмента.

Программа ассемблер должна знать заранее, через какие сегментные регистры будут адресоваться ячейки программы и чтобы сообщить ей об этом используется директива ASSUME (« assume » – предположить). С помощью директивы ASSUME ассемблеру сообщается информация о соответствии между сегментными регистрами, и программными сегментами.

Модель памяти задается директивой .MODEL, после установления которой, вступают в силу упрощенные директивы определения сегментов, объединяющие действия директив SEGMENT и ASSUME.

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***.

   Директива end указывает ассемблеру, что это конец исходного файла. После этой директивы ассемблер не будет обрабатывать дальнейший код.

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

   Адресную шину можно представить в виде 20 проводников, в каждом из которых может либо протекать напряжение заданного уровня (сигнал 1), либо отсутствовать (сигнал 0).
   Таким образом, микропроцессор оперирует с двоичной системой счисления (двоичной системой представления данных). Символьная информация кодируется в соответствии с кодом ASCII (Американский стандартный код для обмена информацией).
   Числовые данные кодируются в соответствии с двоичной арифметикой. Отрицательные числа представляются в дополнительном коде.
   Минимальная единица информации, соответствующая двоичному разряду, называется **бит** (**Bit**).
   Группа из восьми битов называется **байтом** (**Byte**) и представляет собой наименьшую адресуемую единицу – **ячейку памяти**.
   Биты в байте нумеруют справа налево цифрами 0…7.

   Двухбайтовое поле образует **шестнадцатиразрядное машинное слово** (**Word**), биты в котором нумеруются от 0 до 15 справа налево. Байт с меньшим адресом считается младшим.

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

    Адрес строки Hello загружается в регистр DX с помощью оператора OFFSET (смещение):
    `OFFSET имя`
    Значением оператора является смещение указанного имени, отсчитанное от начала того сегмента, в котором оно описано. Имя может быть именем переменой, или меткой (именем команды). При трансляции ассемблер ставит в соответствие имени переменной Hello, описанной в директиве DB, смещение первого байта строки Hello относительно начала сегмента DS. С помощью оператора OFFSET мы записываем это смещение в DX.

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 номером 21h (33 – в десятичной системе счисления) 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. 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 '$'.

12. **Программы, которые должны исполняться как .EXE и .COM, имеют существенные различия по:**
    - размеру,
    - сегментной структуре,
    - механизму инициализации.

Охарактеризуйте каждый из перечисленных пунктов.

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

Модель памяти TINY, в которой код, данные и стек размещаются в одном и том же сегменте размером до 64Кб

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).

COM-программы предпочтительнее EXE-программ, когда дело касается небольших ассемблерных утилит. Они быстрее загружаются, ибо не требуется перемещения сегментов, и занимают меньше места на диске, поскольку EXE-заголовок и сегмент стека отсутствуют в загрузочном модуле.