БЕЛОРУССКИЙ  ГОСУДАРСТВЕННЫЙ  УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Кафедра технологий программирования

**﻿﻿﻿ QUEST**

Домашнее задание

Шибко Татьяны Александровны

студентки 4 курса 12 группы

специальность "прикладная информатика"

|  |  |
| --- | --- |
| Дата окончания работы 06.10.2024 г. | Преподаватель: |
|  | Доцент К.А. Зубович |

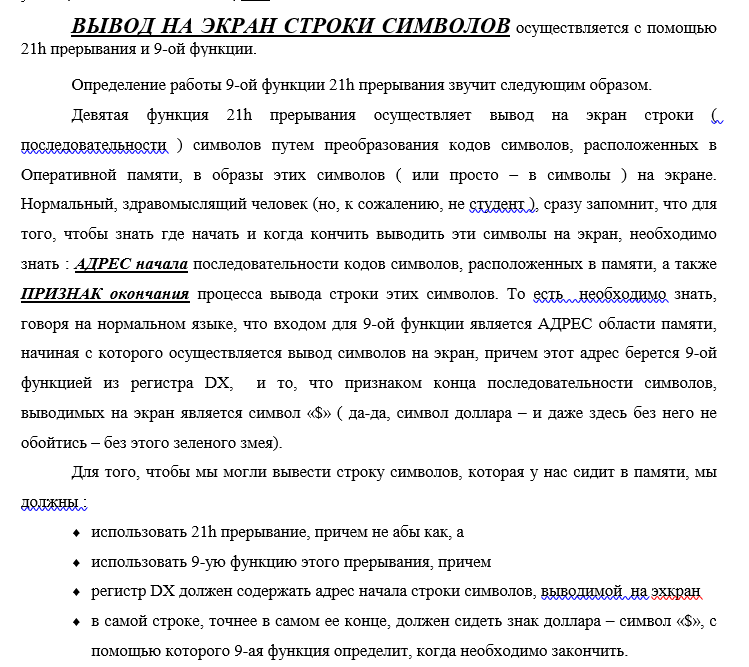
Минск, 2024

# Вопрос 1:

1.1. Дать (взяв его из Лекций Зубовича и указав номера лекций, страницы и сколько раз даётся это описание) описание функции 9 21h прерывания на примере, в котором область памяти Text определена следующим образом:

Text db ‘Mama’, 10, ‘Mila’, 13, ‘ramu’, ‘10’,13,”$”

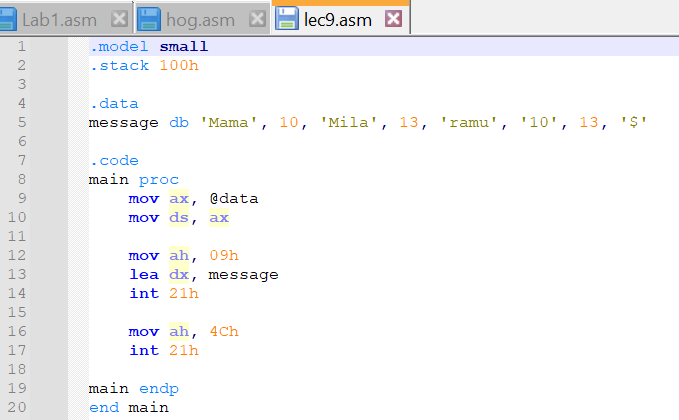
**Ответ:** Лекция 4. «Почти криминальное чтиво или символическая сноваа», страница 60



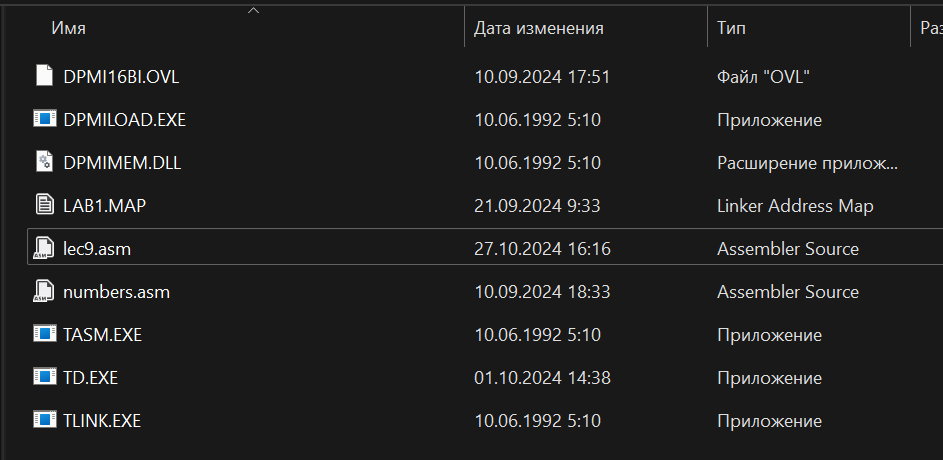
1.2. Ответить на вопрос: «Что будет выведено на экран»? Желательно использовать средства языка записи алгоритмов ассемблер, оформив соответствующий текст, протранслировав его, пролинковав объектный модуль, получив исполнимый модуль, загрузив его в ОП и отладив с помощью отладчика, выполнить полученную программу по выводу на экран указанной последовательности символов.

**Ответ:** Чтобы увидеть вывод программы, я создала отдельную папку lec9, куда перенесла нужные файлы для TASM, TLINK и TD.

Текст программы lec9.asm

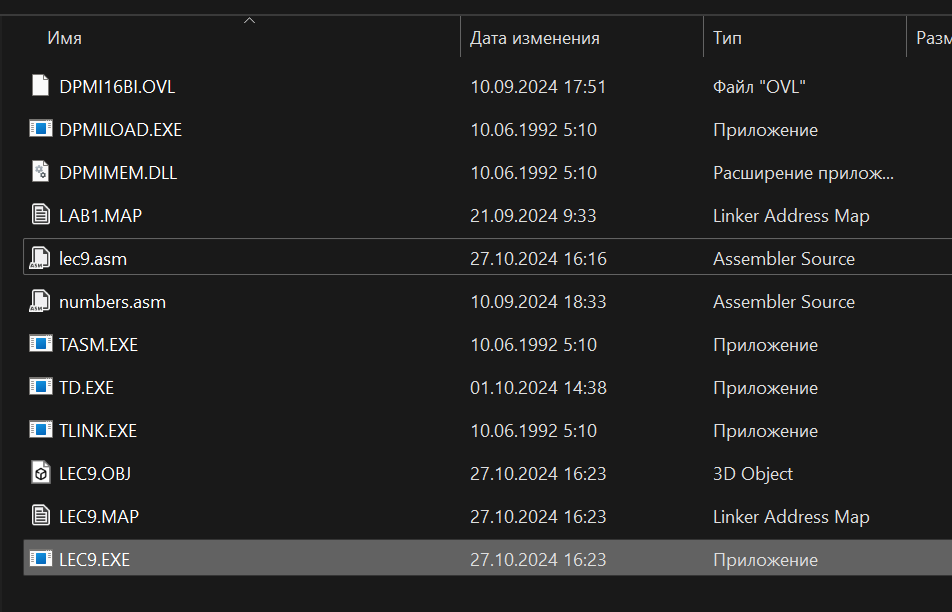


Папка с файлами. Все файлы я брала из папки с папой и рамой.

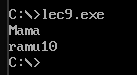


Теперь заходим в DosBox, где отработанными движениями получаем файл .EXE





Теперь мы можем просто запустить через lec9.exe и увидеть



# Вопрос 2. Какое макрорасширение будет сформировано ПРЕПРОЦЕССОРОМ по макрокоманде PutStr.

**Ответ:** При вызове макрокоманды PutStr будет сформировано расширение:

**mov ah,09h 🡪** запрашиваем девятую функцию

**mov dx, offset Text 🡪** кладём адрес нужной строки в DX

**int 21h 🡪** вызываем функцию

# Вопрос 3. Доказать или опровергнуть, что «первый совпадающий по маске файл» определяется «текущим признаком сортировки» файлов в папке (по имени, по Дате изменения, по типу, по размеру).

**Ответ:** Попытаюсь опровергнуть

1. **Совпадение по маске**:
   * Маска файла фильтрует файлы по заданным критериям (например, \*.txt), но не влияет на порядок их отображения.
2. **Признак сортировки**:
   * Сортировка (по имени, дате, типу, размеру) определяет порядок отображения файлов, но не изменяет сам процесс поиска по маске.
3. **Независимость поиска**:
   * При поиске по маске система рассматривает все соответствующие файлы, независимо от порядка. Например, файл может быть найден первым, даже если сортировка стоит по дате изменения.

# Вопрос 3.0. Где нарвал наврал ghj rjvfyle Ьщмыи?

BegScan:

push ds

pop es

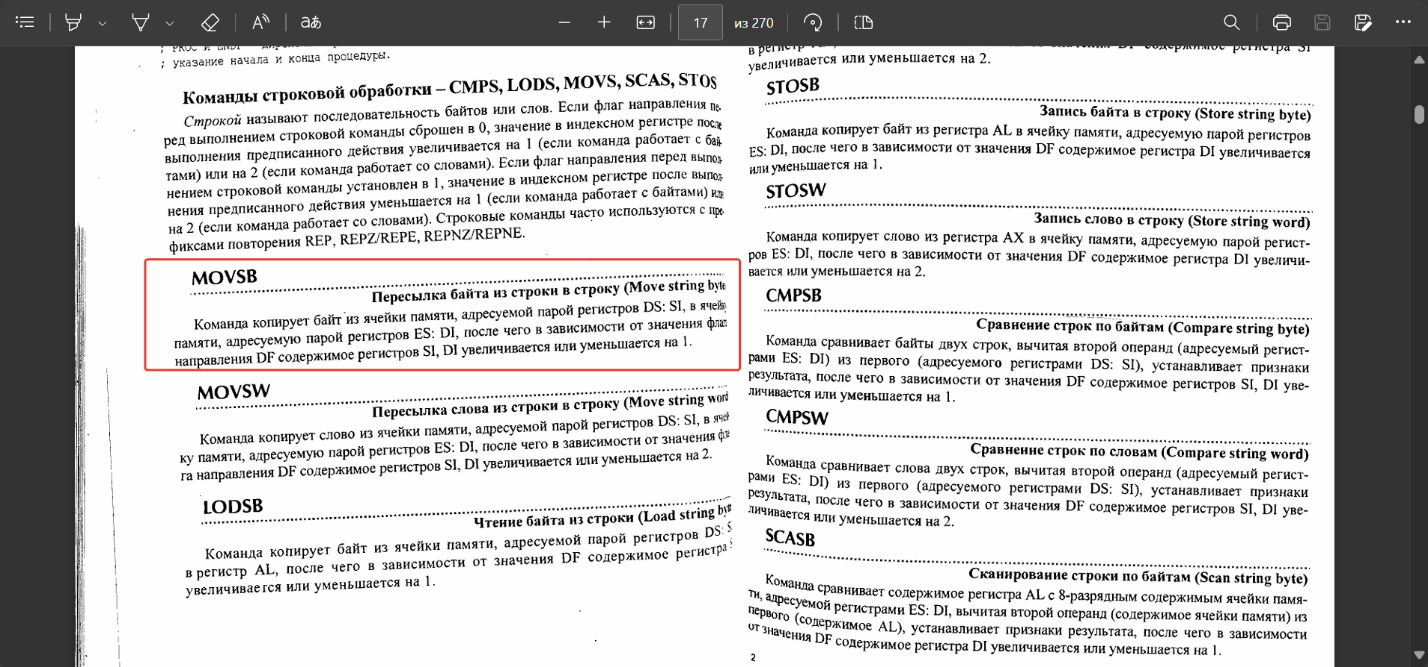
mov si,FN\_Ofs

mov di,offset FName

mov cx,13

rep movsb

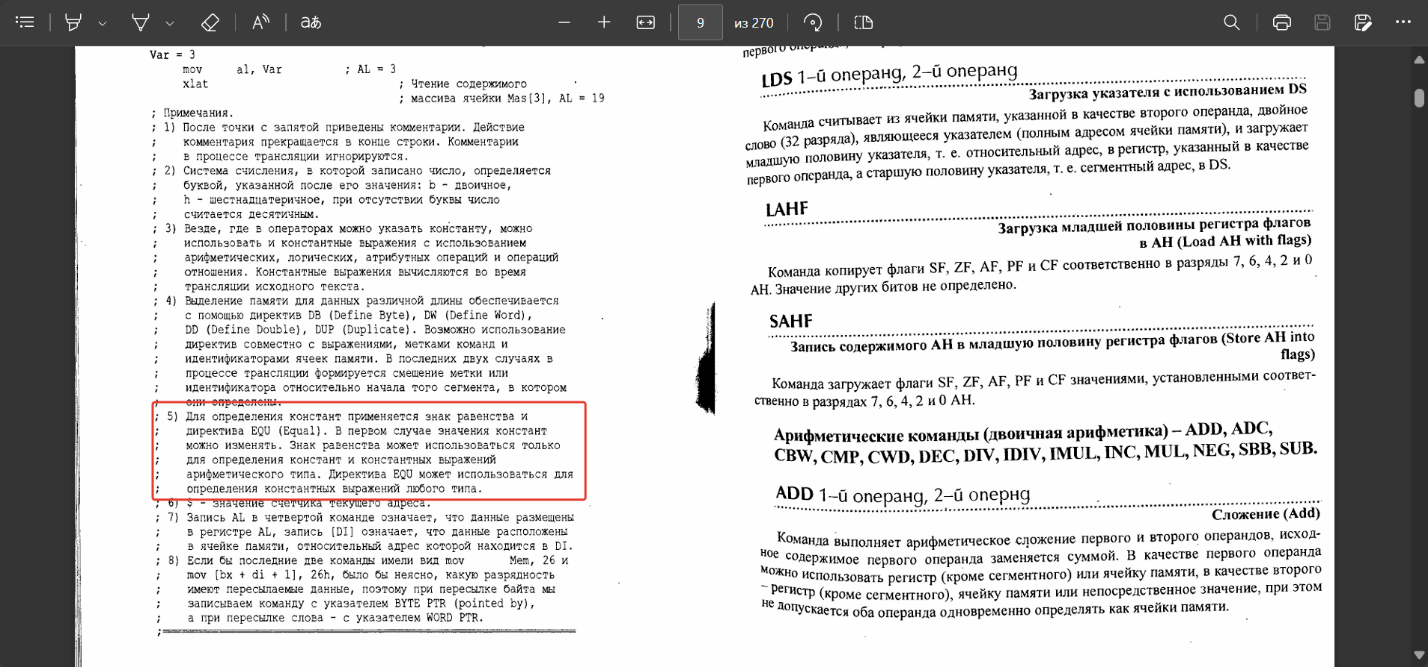
**Ответ:** ответ можно найти в полезной книге по ассемблеру.



# Вопрос 4: Адрес области памяти FN\_Ofs определен следующим образом:

FN\_Ofs equ offset DTA+1eh. Что такое Equ, что такое адрес, что такое DTA + 1eh и что означает offset.

**Ответ:** Так же можно найти часть ответа в книге



Адрес: Это место в памяти, где хранится значение.

DTA + 1eh: Указывает на смещение от начала области DTA, на 30 байт.

Offset: Указывает на адрес начала элемента в памяти.

# Вопрос 5: Нарисовать в шестнадцатеричном виде содержимое регистра СХ после выполнения команды MOV CX, -128. И что произойдёт, если вместо фрагмента:

mov si,FN\_Ofs

mov di,offset FName

mov cx,13

rep movsb

записать

mov si,FN\_Ofs

mov di,offset FName

mov cx,-128

rep movsb

Дома с помощью турбо-отладчика доказать, что как Вы написали на листочке, так и будет…

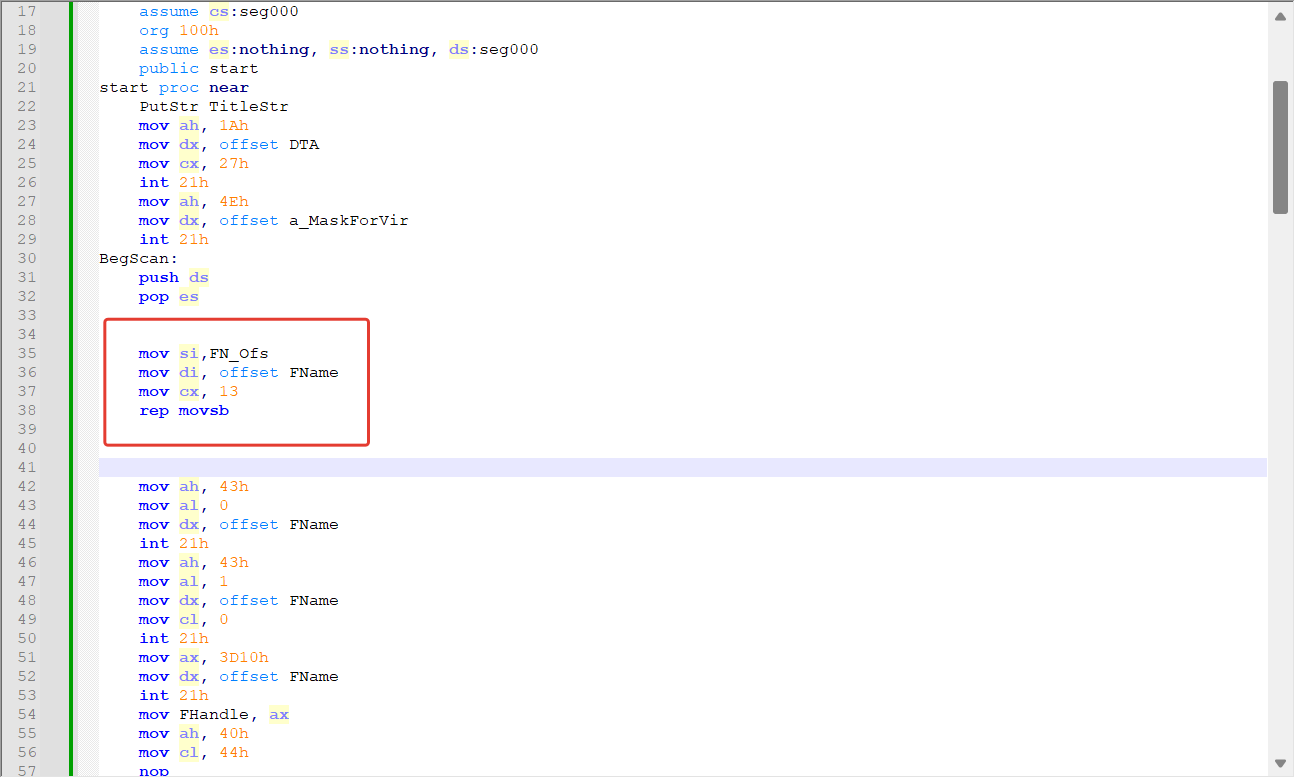
**Ответ:** Регистр CX состоит из двух байтов (CH и CL) и функционирует как счетчик. Его диапазон значений составляет от 0 до 65535. При попытке записать значение -128 в регистр, фактически получится:

216 - 128 = 65536 - 128 = 65408

В шестнадцатеричном формате это будет представлено как FF80h.

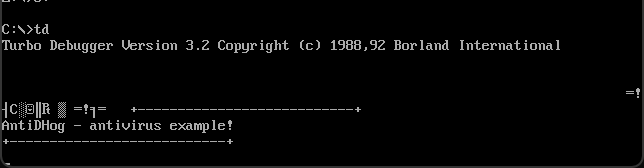
|  |  |
| --- | --- |
| FFh | 80h |
| байт | байт |
| CX | |

Исходный кусочек кода:

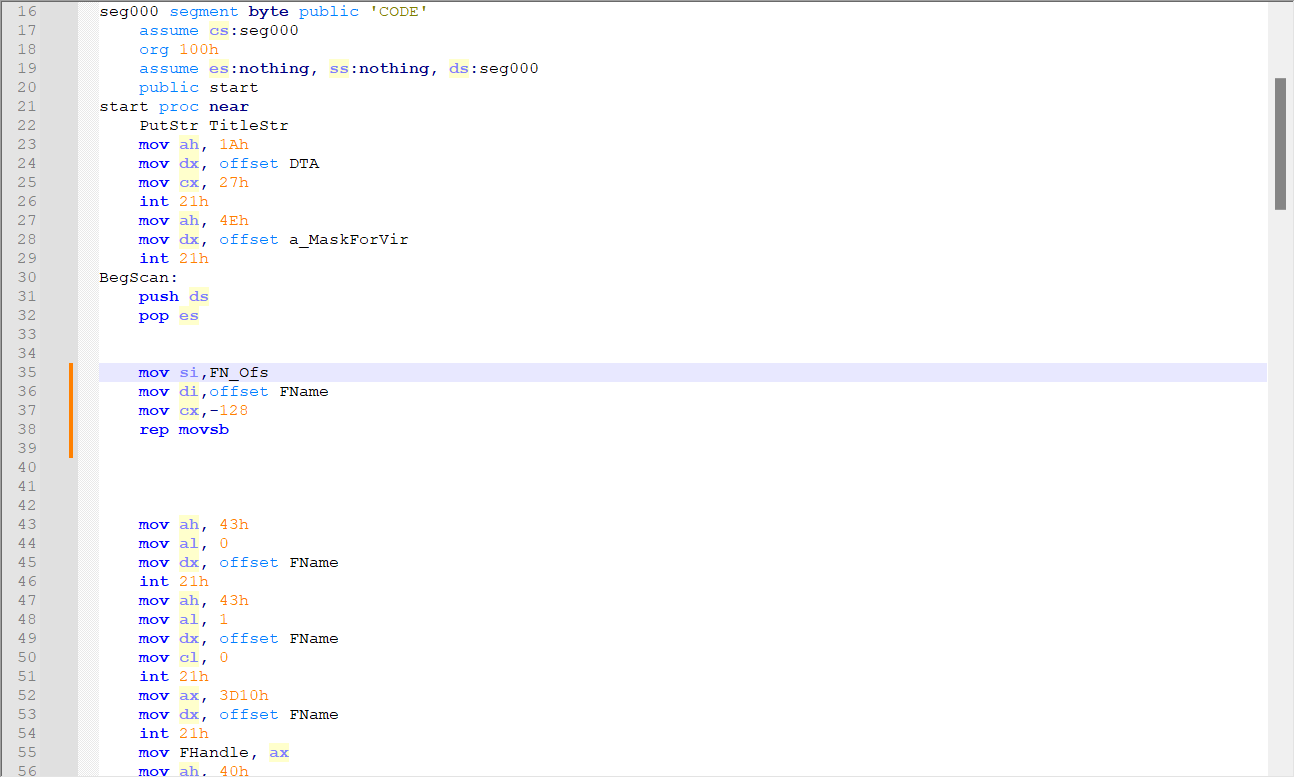


Получаем исполнимый файл через TASM, TLINK.

Запустили через TD.EXE



Теперь заменим кусок:



После пересборки и запуска программы в td, она висит какое-то время, а потом DosBox просто вылетает.

Проблема заключается в том, что инструкция rep movsb будет пытаться скопировать данные 65536 (216) раз, что намного превышает реальный объем доступной памяти. Это приводит к зацикливанию программы, отсутствию завершения работы и, в конечном итоге, к вылету из DosBox.

# Вопрос 6: Подготовить текст на языке записи алгоритмов ассемблер ДЛЯ ВЫВОДА НА ЭКРАН название файла и Handle файла, имя которого находится в FName. Отладить программу оформив соответствующий текст по аналогии с папой, и записать на листочке полный полученный текст и выведенный на экран номер – Handle файла. То есть нужно получить программу, в которой в области памяти FName записано имя файла, например, mrak.asm, открыть этот файл, получив Handle, и вывести на экран сначала имя файла, а затем ЧИСЛО-НОМЕР-Handle файла. Например так:

Mrak.asm 5

**Ответ:** Полный код программы

.model small

.stack 100h

.data

FName db 'q6.asm', 0 ; Имя файла

OblVvoda db 'Filename: $', 0 ; Буфер для вывода имени файла

HandleFile db 'Handle: $', 0 ; Буфер для вывода Handle файла

Handle db ? ; Хранение Handle файла (8-битное значение)

.code

start:

mov ax, @data

mov ds, ax

; Открыть файл

mov ah, 3Dh ; Функция для открытия файла

xor al, al ; Открыть для чтения

lea dx, FName ; Указатель на имя файла

int 21h ; Вызов DOS

mov Handle, al ; Сохранить Handle файла

; Вывести имя файла

mov ah, 9 ; Функция для вывода строки

lea dx, OblVvoda ; Указатель на строку "Filename: "

int 21h ; Вызов DOS

lea dx, FName ; Указатель на имя файла

int 21h ; Вызов DOS

; Вывести Handle файла

mov ah, 9 ; Функция для вывода строки

lea dx, HandleFile ; Указатель на строку "Handle: "

int 21h ; Вызов DOS

; Преобразование Handle в строку для вывода

mov al, Handle ; Получаем Handle

call PrintNumber ; Печатаем Handle

; Завершить программу

mov ah, 4Ch

xor al, al

int 21h

; Процедура вывода 8-битного числа в строковом формате

PrintNumber proc

push ax

push dx

push cx

; Проверяем, равен ли Handle нулю

cmp al, 0

je PrintZero

; Преобразуем AL в строку

mov cx, 10 ; Основание для деления

xor dx, dx ; Обнуляем остаток

mov bx, 0 ; Счетчик символов

; Деление на 10

PrintLoop:

xor dx, dx

div cx ; Делим AX на 10

push dx ; Сохраняем остаток

inc bx ; Увеличиваем счетчик

test al, al

jnz PrintLoop ; Повторяем, пока не станет нулем

; Печатаем символы в обратном порядке

PrintDigits:

pop dx ; Извлекаем остаток

add dl, '0' ; Преобразуем в символ

mov ah, 2

int 21h ; Выводим символ

dec bx ; Уменьшаем счетчик

jnz PrintDigits ; Печатаем следующий символ

pop cx

pop dx

pop ax

ret

PrintZero:

mov dl, '0' ; Если число ноль

mov ah, 2

int 21h ; Выводим символ '0'

pop cx

pop dx

pop ax

ret

PrintNumber endp

end start

Результат:



# Вопрос 7: Зачем здесь ЗАПИСАНЫ две команды Nop. Что будет, если их убрать?

***mov FHandle,ax***

***mov ah,40h***

***mov cl,44h***

***nop***

***nop***

**Ответ:**

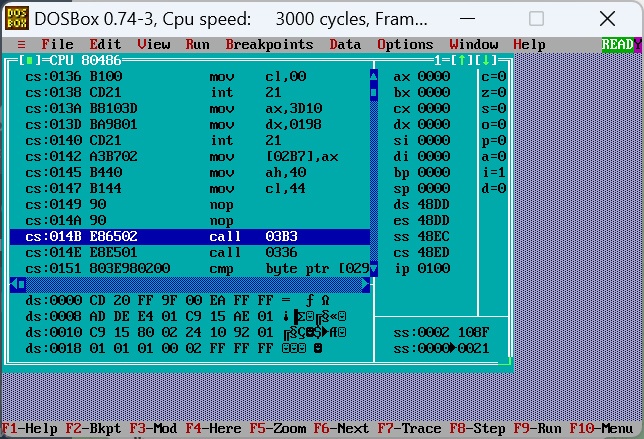
Команды nop нужны здесь для:

1. Выравнивание кода: обеспечивают оптимальную производительность процессора.
2. Заполнение: резервируют место для будущих изменений в коде.
3. Отладка: позволяют временно отключить инструкции для тестирования.

Удаление этих команд может изменить порядок выполнения и привести к ошибкам или неожиданным последствиям в работе программы.

# Вопрос 8: Далее осуществляется вызов процедуры InfoAboutFile. Что происходит при вызове процедуры, как передаётся управление процедуре и что происходит со стеком. Готовя отчёт к концу недели, используя отладчик показать, что находится в стеке до вызова процедуры и после выхода из процедуры, определить адрес ТОЧКИ ВОЗВРАТА из этой процедуры в головную программу. НАРИСОВАТЬ!!!!

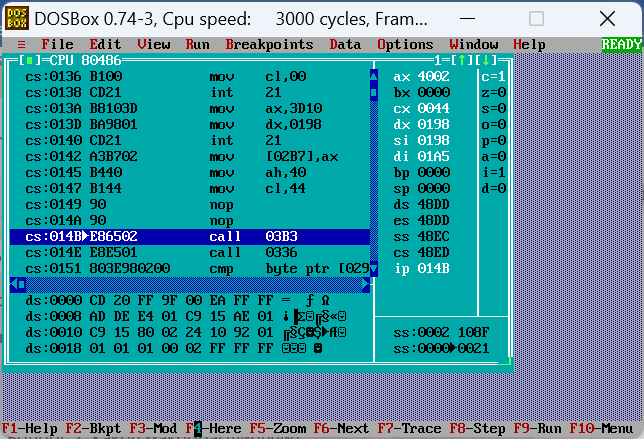
**Ответ:** **до вызова процедуры**



Адрес возврата 🡨 *Этот адрес (E86502) будет точкой возврата в главную программу.*

Другие данные

**После**



Адрес точки возврата: E86502

Следующий адрес 🡨 *Адрес, на который управление вернётся после выхода из процедуры.*

При вызове процедуры, текущий адрес возврата сохраняется в стеке, и управление передается на первую инструкцию вызываемой процедуры. Во время выполнения, процедура может использовать стек для хранения собственных данных. По завершении, адрес возврата извлекается из стека, и управление возвращается в основную программу.

# Вопрос 9. В чём разница между действиями: УБИТЬExecutable и УДАЛИТЬExecutable. Ответ типа: «Это соответствующий перевод слов Kill и Remove» - НЕ ПРИНИМАЕТСЯ…

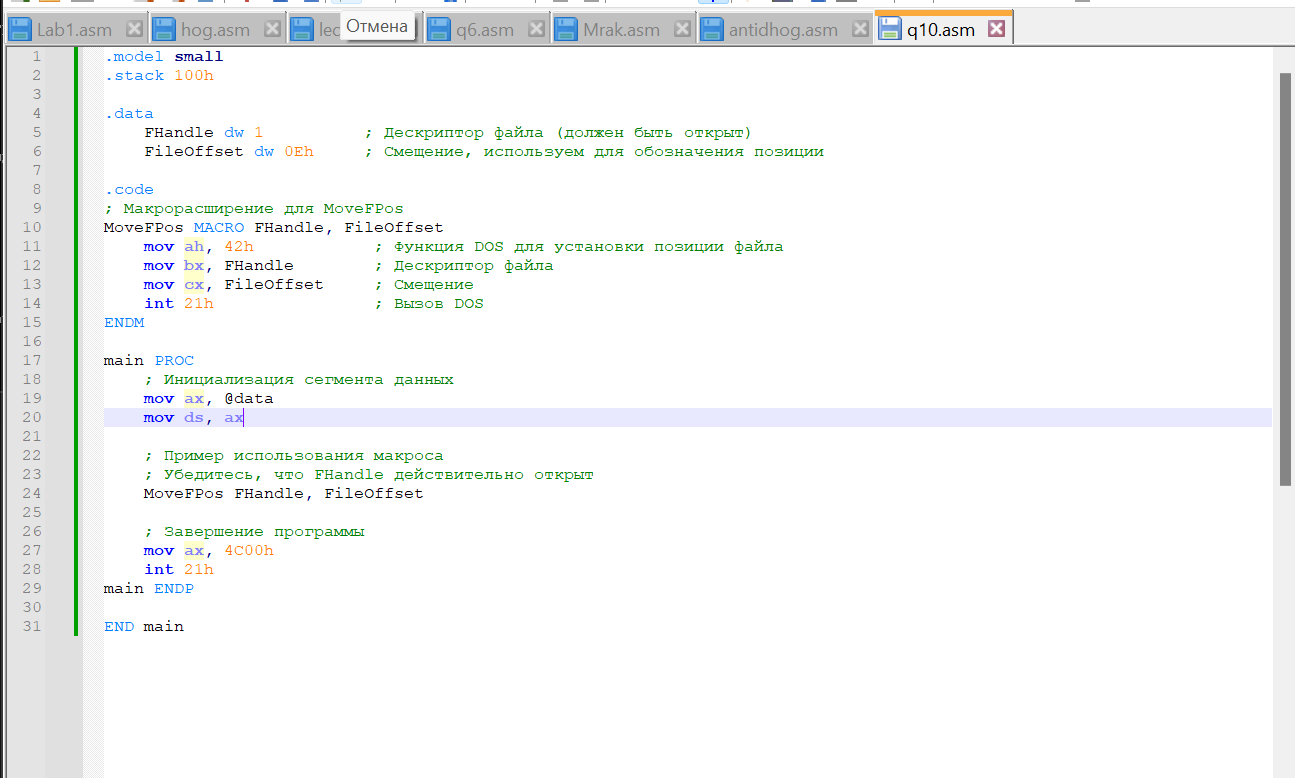
**Ответ:**

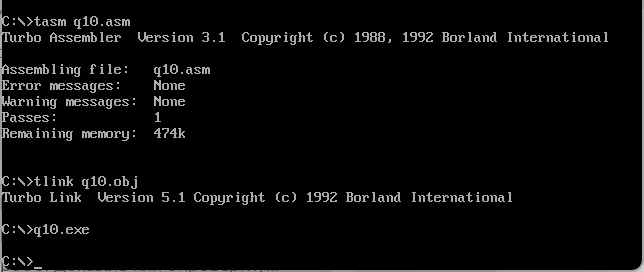
УБИТЬExecutable означает завершение его выполнения, включая остановку процессов и освобождение ресурсов.

УДАЛИТЬExecutable означает физическое удаление файла из файловой системы, что не влияет на уже запущенные процессы.

# Вопрос 10: Построить макрорасширение, соответствующее макрокоманде

**Ответ:** код программы





Макрорасширение MoveFPos используется для установки позиции указателя файла в DOS.

# Вопрос 11: Зачем ЗДЕСЬ ЗАПИСАНЫ эти команды Push и Pop. С какой целью и для какой из ниже следующих команд.

mov si,offset VirSignature

mov di,offset SignatureArray

mov cx,14

repe cmpsb

**Ответ:** Цели push и pop:

**Сохранение состояния**:

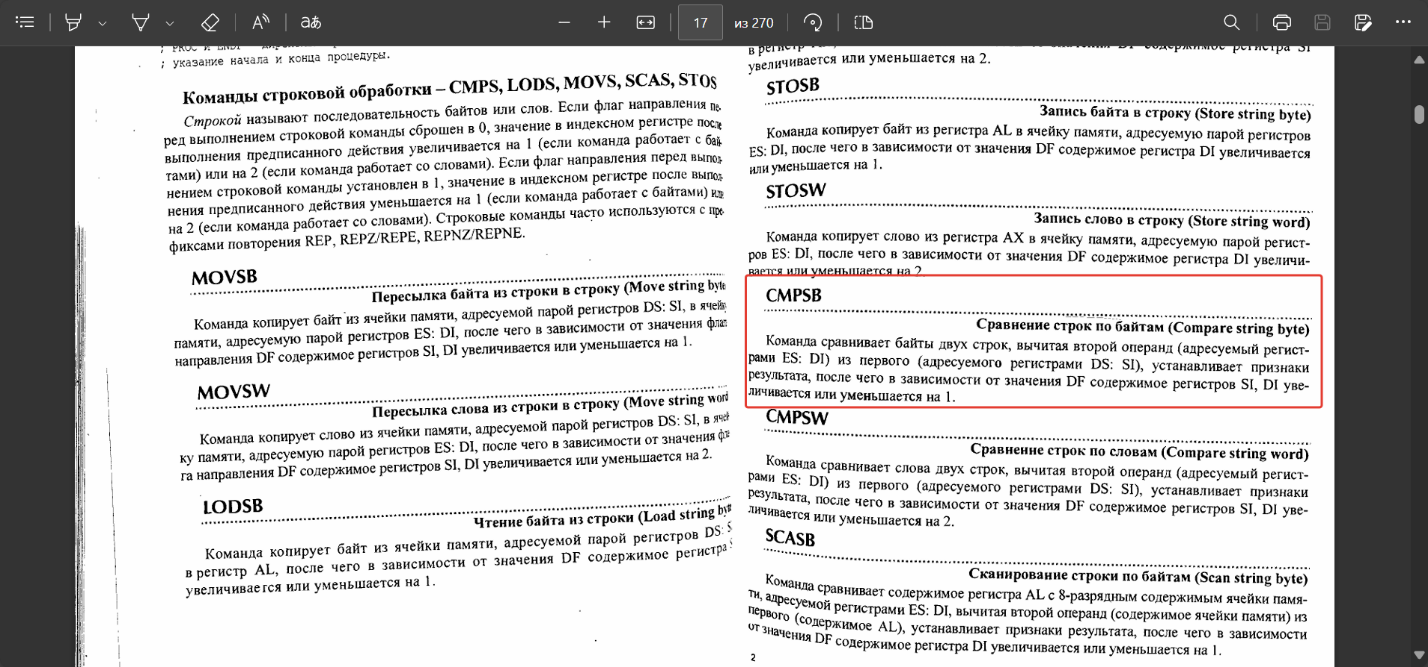
Если перед этими командами используются push, это может быть сделано для сохранения значений регистров (например, si, di, cx), чтобы они не были перезаписаны в дальнейшем коде. Это важно, если после выполнения этих команд будут другие операции, которые могут изменить эти регистры.

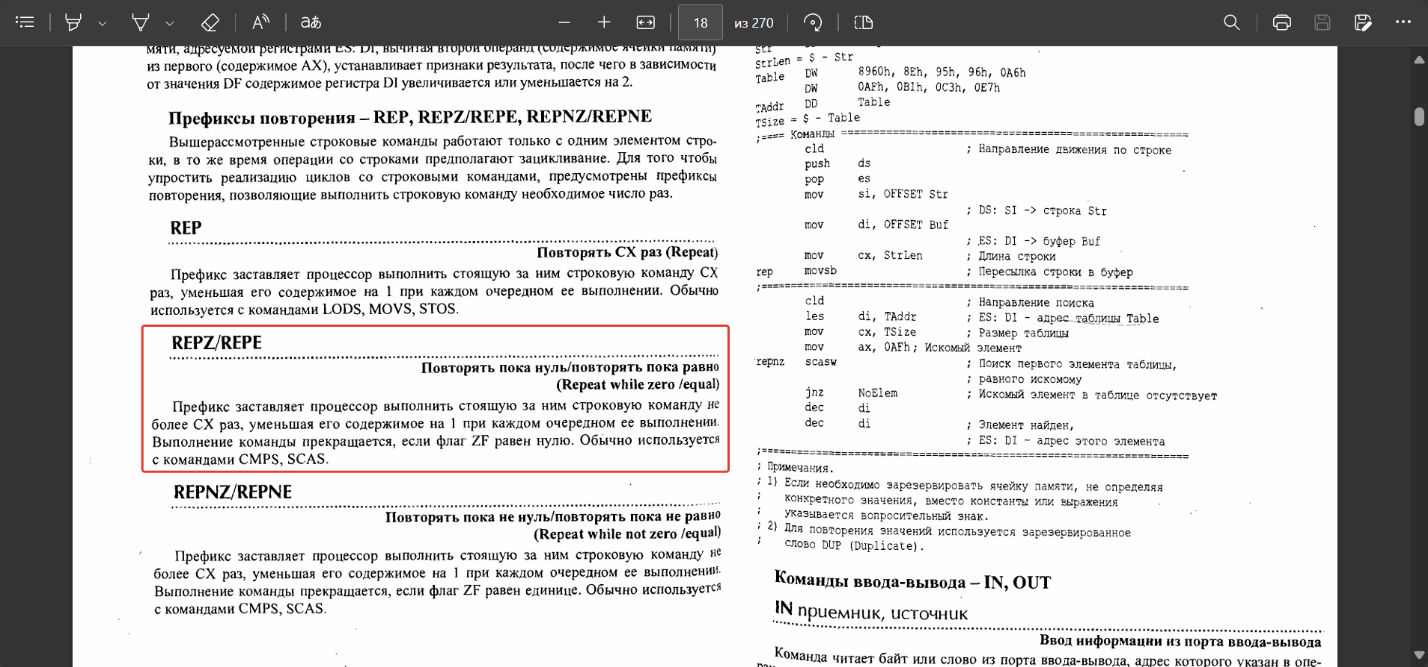
**Восстановление состояния**:

После выполнения команд, может следовать pop, чтобы восстановить ранее сохраненные значения регистров. Это гарантирует, что программа продолжит работать с теми же значениями регистров, что и до выполнения блока кода.

# Вопрос 12: Дать ПОЛНОЕ (нарисовать!!!!) ОПИСАНИЕ того, ЧТО ПРОИСХОДИТ в результате repe cmpsb, дать определение Префикса Repe и КОМАНДЫ Cmpsb!!!! Не помоечно!!

**Ответ:**

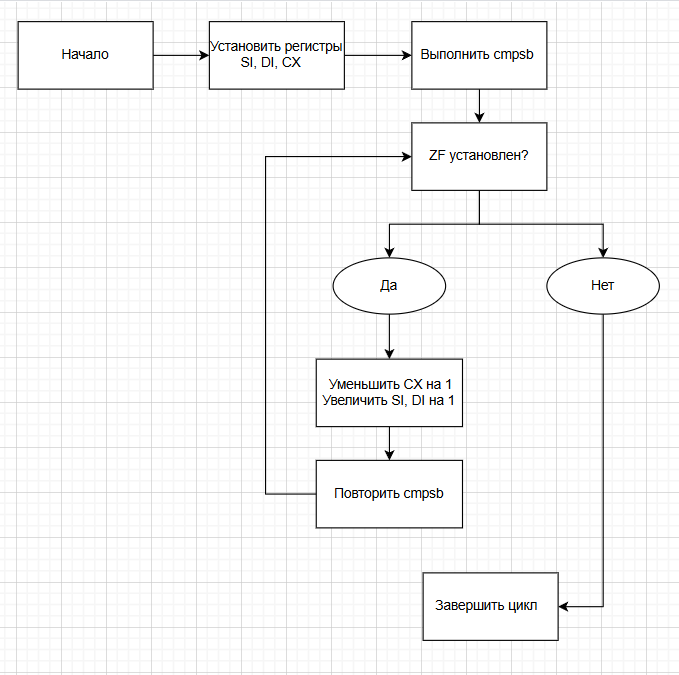




Что происходит при выполнении repe cmpsb

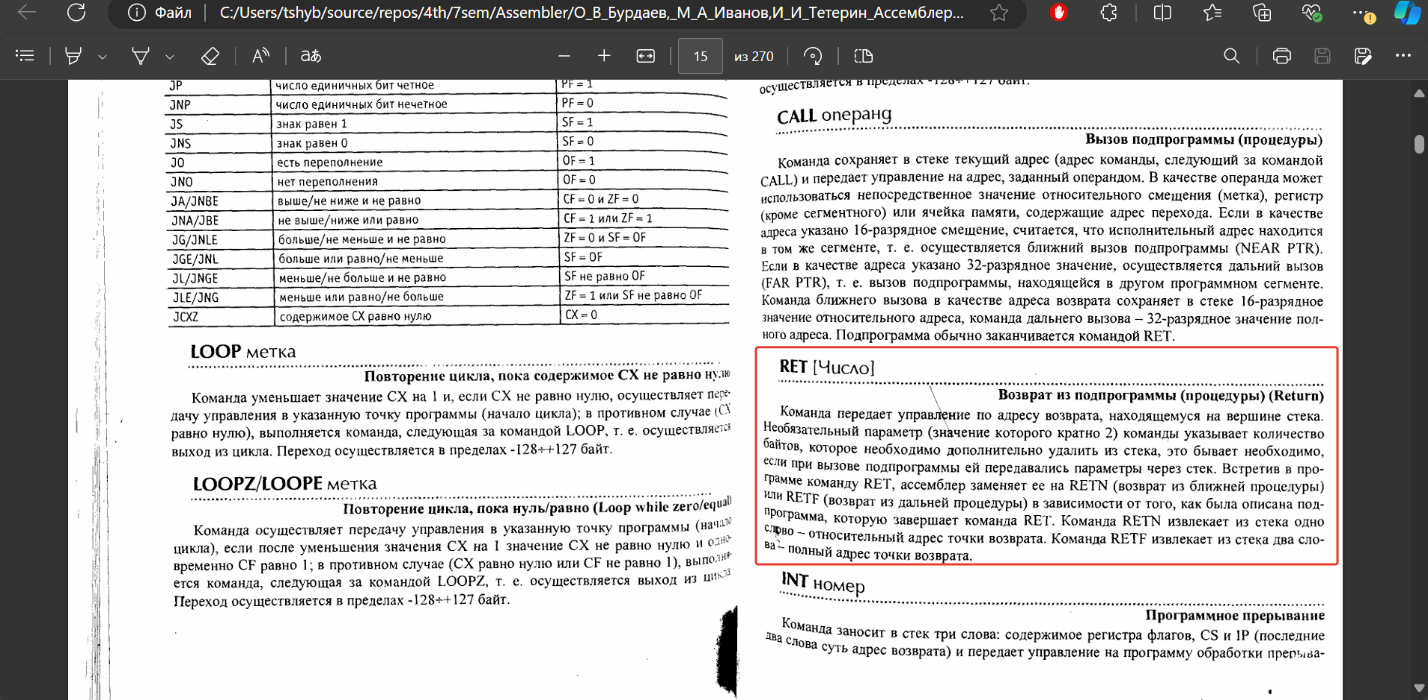
1. **Инициализация**:
   * Перед вызовом repe cmpsb должны быть установлены следующие регистры:
     + SI указывает на первый байт (например, VirSignature).
     + DI указывает на второй байт (например, SignatureArray).
     + CX содержит количество байт для сравнения.
2. **Работа команды cmpsb**:
   * Сравнивает байт, на который указывает SI, с байтом, на который указывает DI.
   * Обновляет флаги:
     + **ZF (флаг нуля)**: установлен, если байты равны.
     + **CF (флаг переноса)**: установлен, если первый байт меньше второго.
3. **Работа префикса repe**:
   * Если ZF установлен (байты равны) и CX не равно нулю:
     + Уменьшает CX на 1.
     + Увеличивает SI и DI на 1 (по умолчанию для cmpsb).
     + Повторяет cmpsb.
4. **Завершение**:
   * Если CX становится нулевым или ZF не установлен (байты не равны), выполнение завершается.

Общая схема:



# Вопрос 13: Нарисовать, что происходит ПО КОМАНДЕ Ret. А В отчёте до конца недели ПОКАЗАТЬ, ЧТО НАХОДИТСЯ В СТЕКЕ ДО ВЫПОЛНЕНИЯ КОМАНДЫ RET и ПОСЛЕ ВЫПОЛНЕНИЯ КОМАНДЫ RET! Используя ОТЛАДЧИК!

**Ответ:**

****

# Вопрос 14: Сколько байт и каких (нарисовать содержимое этих байтов в 16 с.с.) ЗАПИСЫВАЕТСЯ (посредством использования какой функции и В КАКОЙ ФАЙЛ) при вызове процедуры KillExecutable.

**Ответ:**

При вызове процедуры KillExecutable записывается **2 байта**.

Содержимое байтов в шестнадцатеричной системе:

|  |  |
| --- | --- |
| 0CDh | 20h |

**Функция:** int 21h, ah = 40h (функция записи в файл).

**Файл:** файл, указанный дескриптором FHandle.

# **Вопрос 15: Составить текст на языке записи алгоритмов ассемблер (на любом другом языке записи алгоритмов), который после трансляции (компиляции) и линковки ПРЕВРАЩАЕТСЯ в исполнимый модуль, который в свою очередь при загрузке его средствами операционной системы в оперативную память ПРЕВРАЩАЕТСЯ в ПРОГРАММУ, которая ИЗМЕНЯЕТ САМА СЕБЯ, т.е. МЕНЯЕТ СВОЙ ФУНКЦИОНАЛ В ПРОЦЕССЕ ВЫПОЛНЕНИЯ ПРОГРАММЫ! Например, запись в исходном тексте предполагает вывод на экран квадрата введенной с клавиатуры цифры, а в процессе выполнения всё (что-то) меняется и на экран выводится просто само значение введённой цифры. Напоминаю, что под программой понимается не исходный текст на каком-либо языке записи алгоритмов типа ассемблер, гадюки, иди, кофе, острова, марки сигарет или мотоцикла, а последовательность инструкций или команд, которая будучи помещенной в оперативную память МОЖЕТ БЫТЬ ДЕКОДИРОВАНА самой ЭВМ и МОЖЕТ ЗАСТАВИТЬ эту ЭВМ выполнить определённую последовательность действий без вмешательства человека и каких-либо не относящихся к программной части операционной системы программных продуктов. Полученный исходный текст ЗАПИСАТЬ на ЛИСТОЧКАХ, которые должны быть сданы в конце сегодняшних занятий.**

**Ответ:**

.model small

.stack 100h

.data

prompt db 'Enter a number (0-9): $'

square\_msg db 'Square: $'

input db 2 dup(0) ; Buffer for input

square db 0 ; Variable to store the square

newline db 0Dh, 0Ah, '$' ; New line

change\_msg db 'Changed output: $'

.code

main proc

; Выводим сообщение с просьбой ввести число

mov ah, 09h

lea dx, prompt

int 21h

; Читаем число с клавиатуры

mov ah, 0Ah

lea dx, input

int 21h

; Преобразуем символ в число

sub byte [input + 1], '0' ; Преобразуем ASCII в число

mov al, [input + 1] ; Загружаем число в AL

mov ah, 0 ; Очищаем AH для умножения

imul ax ; Умножаем AX на AX, результат в AX

mov [square], al ; Сохраняем результат

; Выводим новую строку

mov ah, 09h

lea dx, newline

int 21h

; Выводим сообщение о квадрате

mov ah, 09h

lea dx, square\_msg

int 21h

; Выводим квадрат

mov al, [square]

add al, '0' ; Преобразуем обратно в ASCII

mov dl, al

mov ah, 02h ; Функция для вывода символа

int 21h

; Изменяем функционал: выводим просто оригинальное число

mov ah, 09h

lea dx, newline

int 21h

; Выводим сообщение о изменении

mov ah, 09h

lea dx, change\_msg

int 21h

; Выводим оригинальное число

mov al, [input + 1]

add al, '0' ; Преобразуем обратно в ASCII

mov dl, al

mov ah, 02h

int 21h

; Завершаем программу

mov ax, 4C00h

int 21h

main endp

end main

# Вопрос 16: Какие действия осуществляются транслятором при трансляции ДИРЕКТИВЫ End start. Нарисовать.

**Ответ:**

Действия транслятора при трансляции директивы end start

1. **Завершение обработки исходного файла**:
   * Директива end указывает транслятору, что это конец исходного файла. Все последующие строки после этой директивы игнорируются.
2. **Определение точки входа**:
   * Параметр start указывает точку входа в программу. Это означает, что именно с этого места будет начинаться выполнение программы после загрузки в оперативную память. Транслятор запоминает адрес метки start как начальную точку.
3. **Формирование таблицы символов**:
   * Транслятор обновляет таблицу символов, добавляя в нее информацию о метке start, ее адресе и типе (например, это точка входа).
4. **Подготовка к генерации исполняемого файла**:
   * После обработки директивы end, транслятор начинает подготовку к генерации исполняемого файла, собирая все секции кода, данных и другие элементы, определенные в исходном коде.

