**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №1**

**по дисциплине «Операционные системы»**

**Тема: Исследование структур загрузочных модулей**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6381 |  | Вероха В. Н. |
| Преподаватель |  | Губкин А. Ф. |

Санкт-Петербург

2018

**Постановка задачи**

1. Цель работы:

Исследование различий в структурах исходных текстов модулей .COM и .EXE, структур файлов загрузочных модулей и способов их загрузки в основную память.

1. Описание функций:

|  |  |
| --- | --- |
| Название | Назначение |
| TETR\_TO\_HEX | перевод половины байта в 16-ую систему счисления |
| BYTE\_TO\_HEX | перевод байта регистра AL в 16-ую систему счисления в AX |
| WRD\_TO\_HEX | перевод двух байт регистра AX в 16-ую систему счисления в DI |
| BYTE\_TO\_DEC | перевод байта регистра AL в 10-ую систему счисления, помещая результат в SI |
| PC\_INFO | определяет тип IBM PC |
| SYSTEM\_INFO | определяет версию MS DOS |
| OEM\_INFO | определяет серийный номер OEM |
| USER\_INFO | определяет серийный номер пользователя |
| Write\_msg | вывод на экран |

1. Описание данных:

|  |  |
| --- | --- |
| Наименование | Назначение |
| TYPE\_PC | Тип IBM PC |
| SYS\_VER | Версия системы |
| OEM\_NUM | Номер OEM |
| USER\_NUM | Серийный номер пользователя |

1. Общие сведения:

Тип IBM PC хранится в байте по адресу 0F000:0FFFE, в предпоследнем байте ROM BIOS. Соответствие кода и типа в таблице:

PC FF

PC/XT FE,FB

AT FC

PS2 модель 30 FA

PS2 модель 50 или 60 FC

PS2 модель 80 F8

PCjr FD

PC Convertible F9

Для определения версии MS DOS следует воспользоваться функцией 30H прерывания 21H. Входным параметром является номер функции в AH:

MOV AH, 30h

INT 21h

Выходными параметрами являются:

AL – номер основной версии. Если 0, то <2.0;

AH – номер модификации;

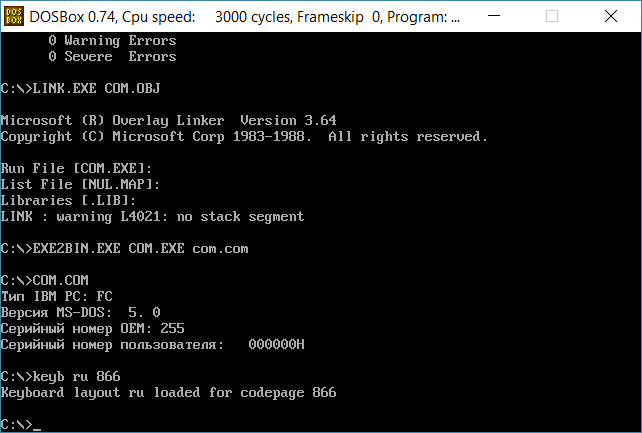
BH – серийный номер OEM (Original Equipment Manufacturer);

BL: CX – 24-битовый серийный номер пользователя.

**Ход работы**

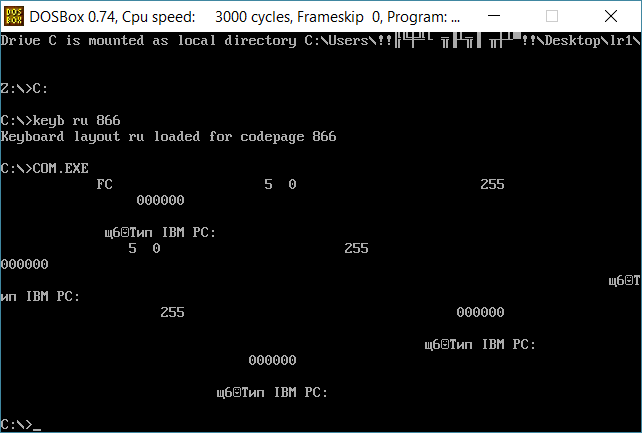
Шаг 1.

Запуск «хорошего» .COM модуля:



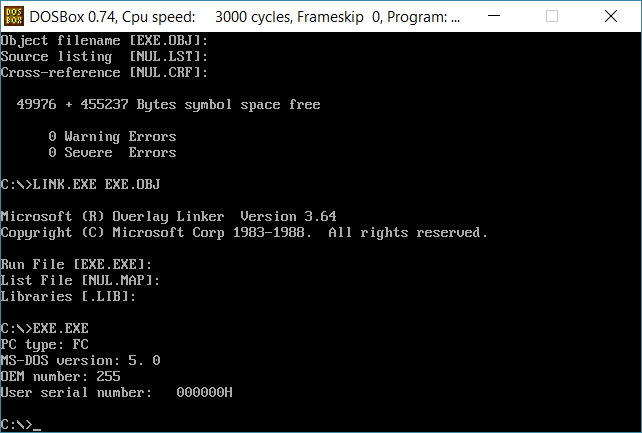
*Рисунок 1*

Запуск «плохого» .EXE модуля:



*Рисунок 2*

Шаг 2. Запуск «хорошего» .EXE модуля:



*Рисунок 3*

Шаг 3. Ответы на контрольные вопросы.

Отличия исходных текстов COM и EXE программ:

1. *Сколько сегментов должна содержать COM-программа?*

COM-программа может содержать только один сегмент (сегмент кода).

1. *EXE программа?*

EXE-программа должна содержать не менее одного сегмента (один сегмент и более). В таких программах могут быть сегменты и для кода, и для данных, и для стека.

1. *Какие директивы должны обязательно быть в тексте COM- программы?*

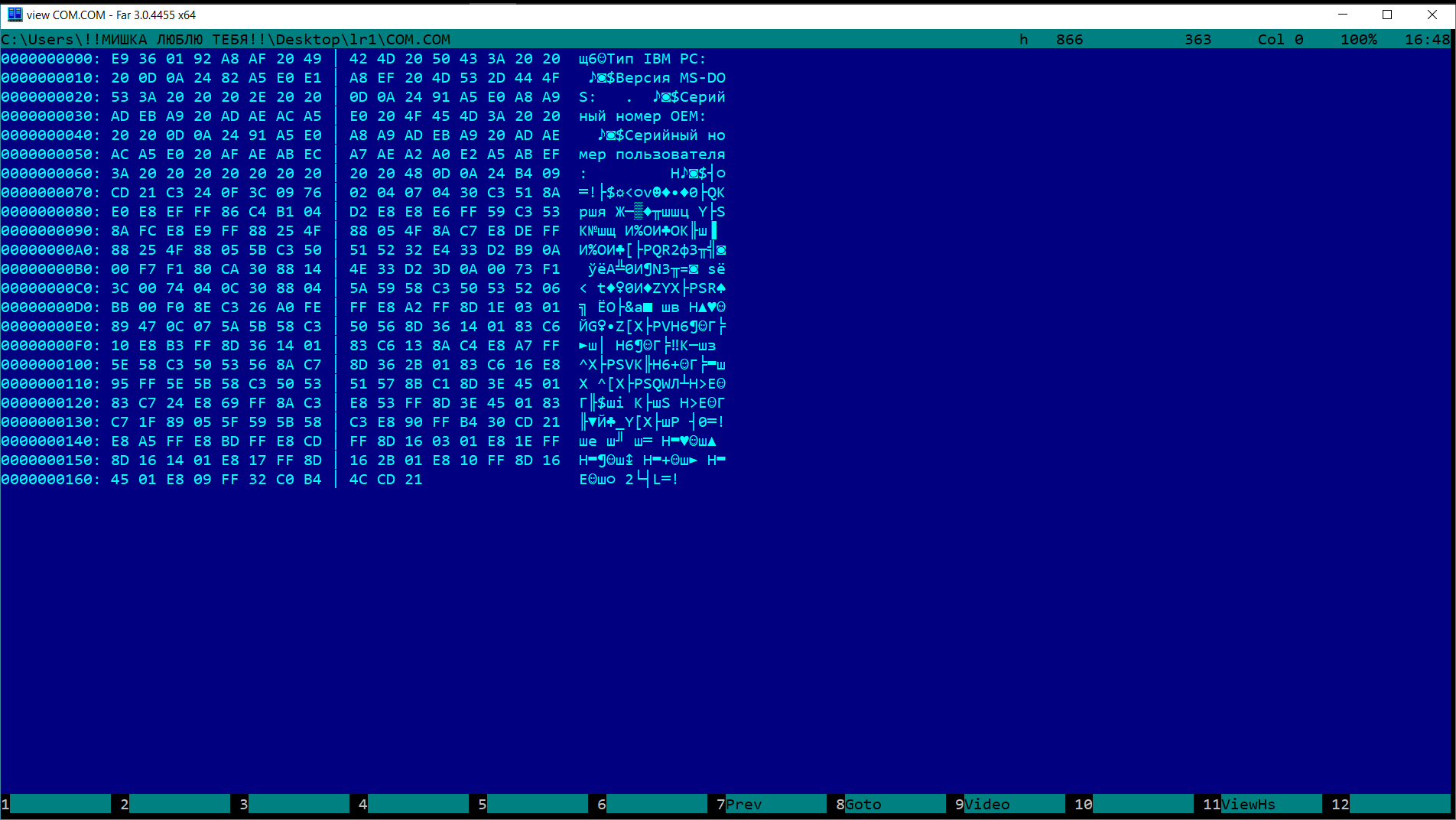
В тексте COM-программы должна обязательно быть директива ORG 100h (смещение 100h), так как при загрузке СОМ файла в память DOS занимает первые 256 байт (100h) блоком данных PSP, после него располагает код программы.

1. *Все ли форматы команд можно использовать в COM-программе?*

Нельзя использовать команды, связанные с адресом сегмента, ибо до загрузки адрес сегмента неизвестен (так как в COM-программе не содержится таблица настроек (Relocation Table), которая содержит описание адресов, зависящих от размещения загрузочного модуля в ОП, поскольку подобные адреса в нем запрещены), поэтому загрузчик не сможет его определить. Также нельзя использовать оператор far: переход на метку возможен только в результате межсегментной передачи управления, а в COM-файле только один сегмент, следовательно, такие переходы невозможны.

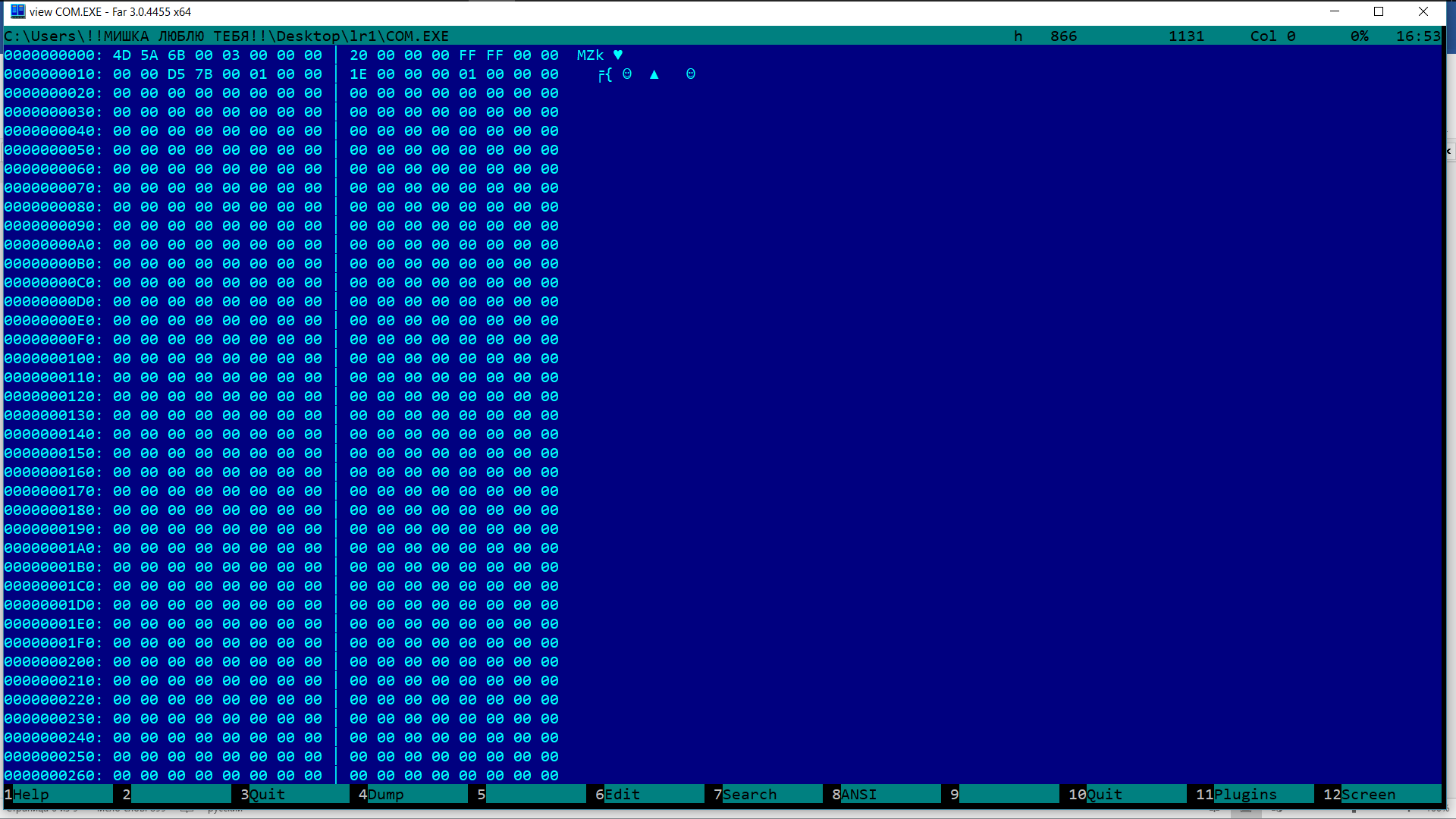
Шаг 4.

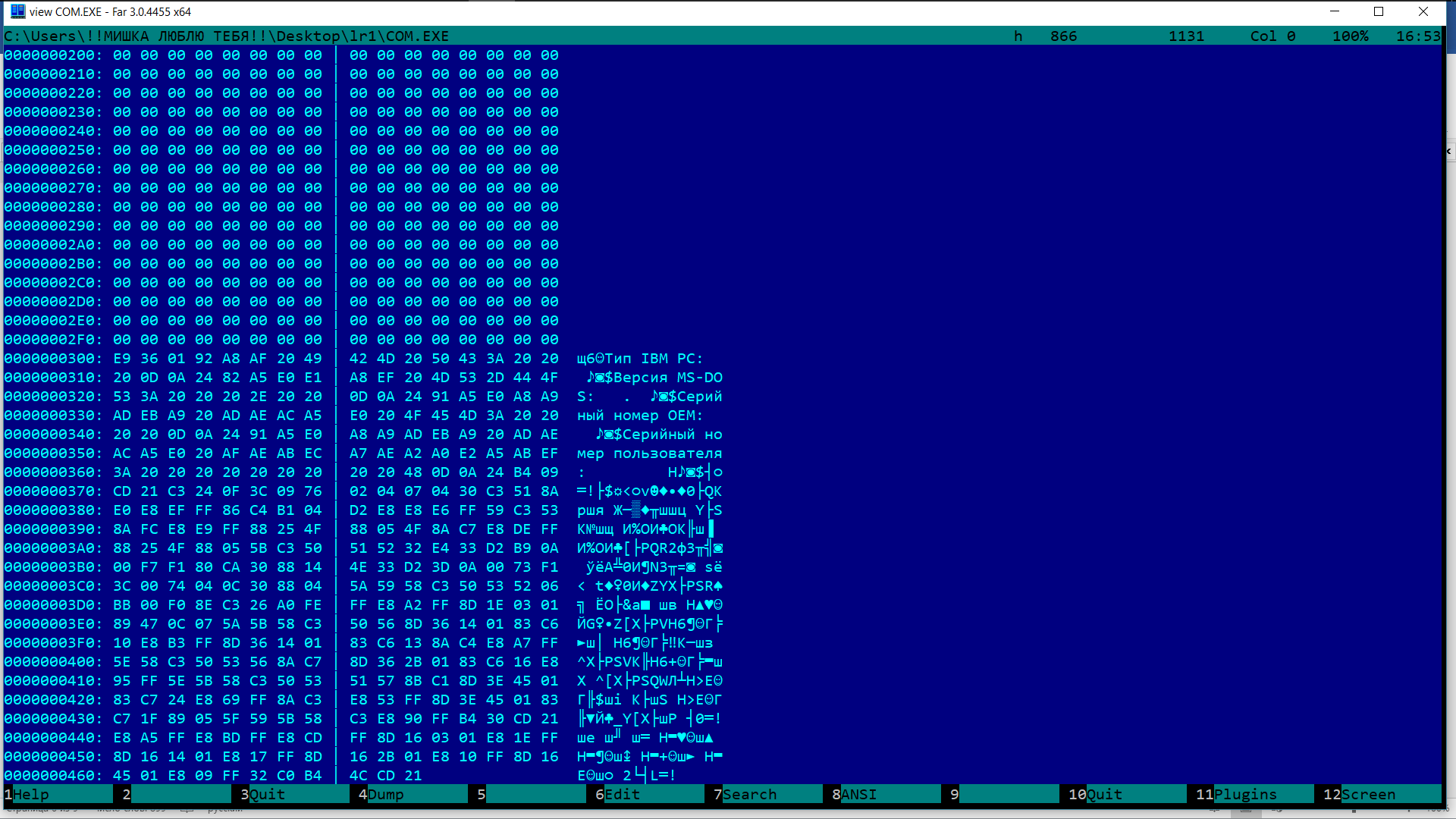
COM-модуль в 16-ричном виде:



*Рисунок 4*

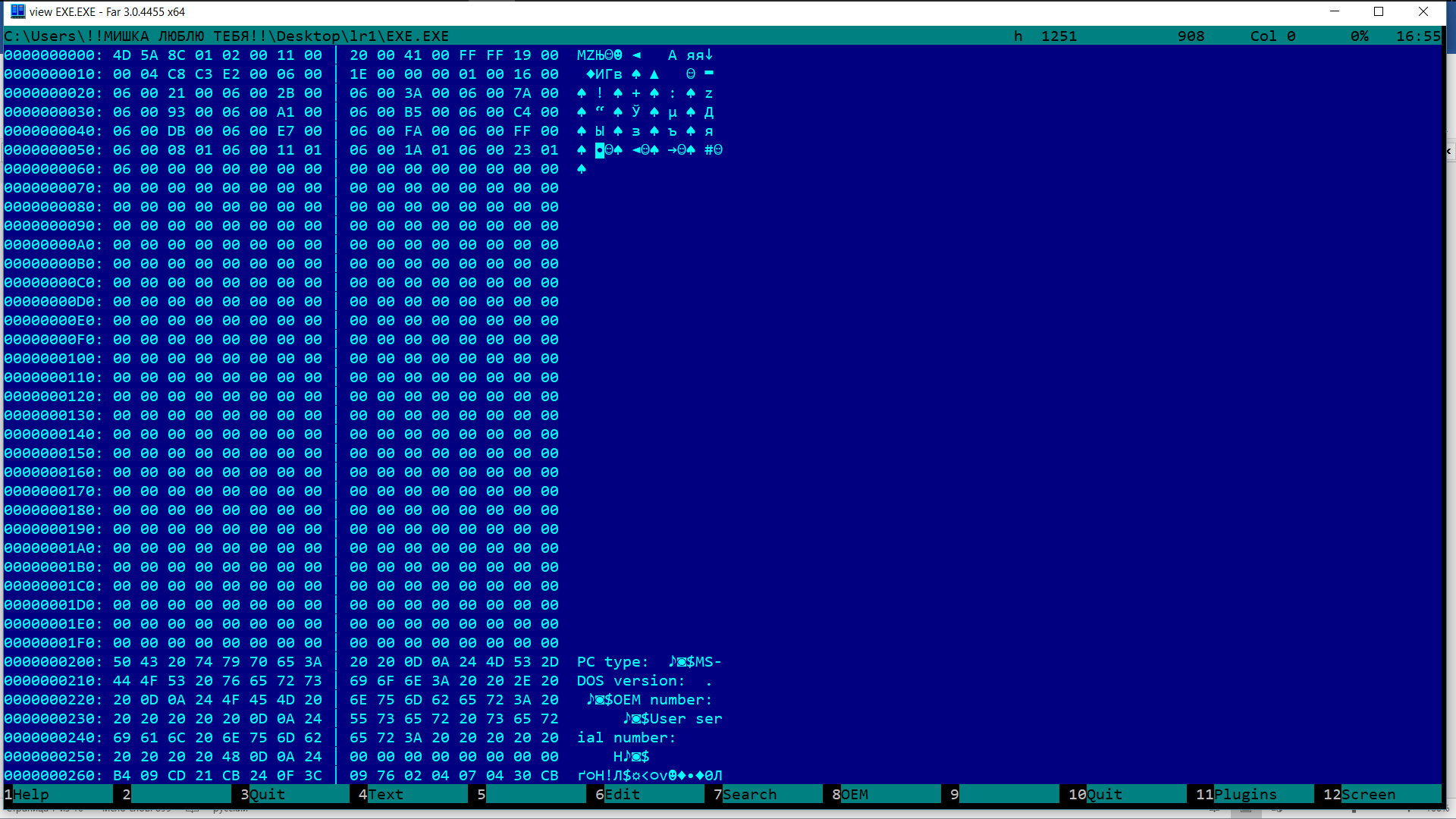
«Плохой» .EXE модуль в 16-ричном виде:

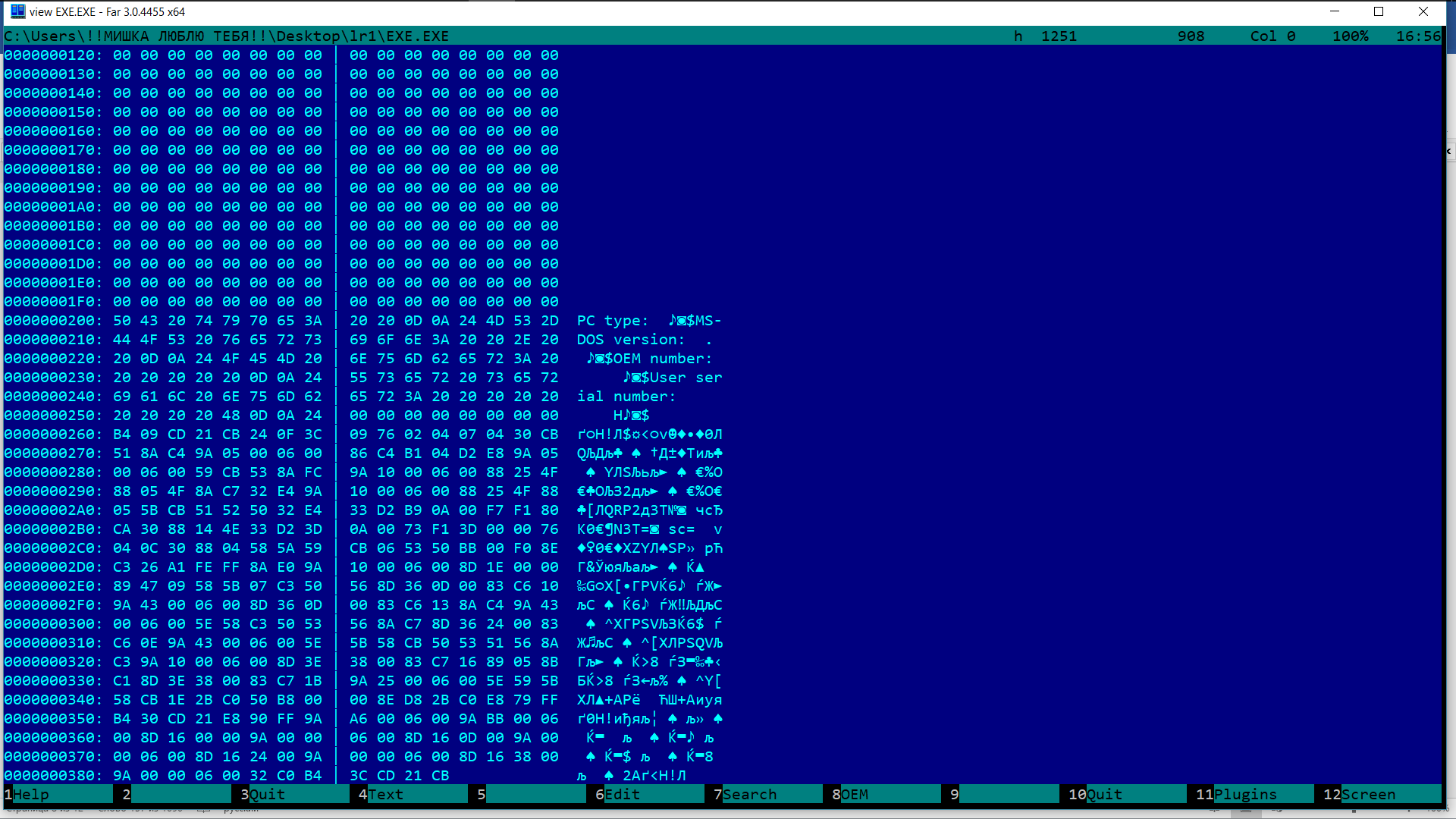




*Рисунок 5, 6*

«Хороший» .EXE модуль в 16-ричном виде:





*Рисунок 7, 8*

Ответы на контрольные вопросы.

Отличия форматов файлов COM и EXE программ:

1. *Какова структура файла COM? С какого адреса располагается код?*

COM-файл состоит из команд, процедур и данных, используемых в программе. Код начинается с адреса 0h, но при загрузке модуля устанавливается смещение в 100h.

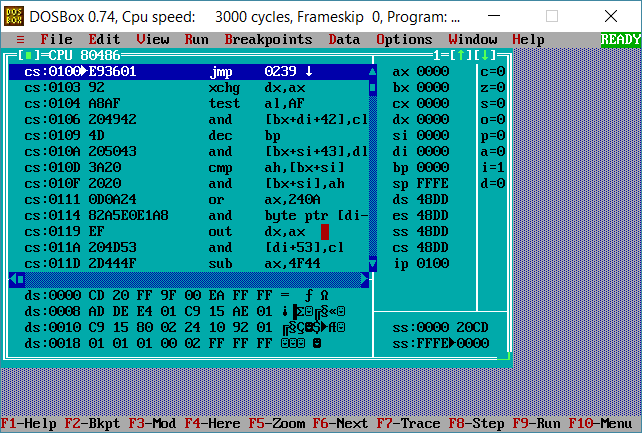
1. *Какова структура файла «плохого» EXE? С какого адреса располагается код? Что располагается с 0 адреса?*

В «плохом» EXE файле данные и код располагаются в одном сегменте. С адреса 0h располагается Relocation Table (таблица разметки), а код - с адреса 300h, так как присутствует директива org 100h, выделяющая место под PSP.

1. *Какова структура файла «хорошего» EXE? Чем он отличается от «плохого» EXE файла?*

В «хорошем» файле EXE содержится информация для загрузчика, сегмент стека, сегмент данных и сегмент кода, то есть 3 сегмента в отличии от одного в «плохом».EXE. Код располагается с адреса 200h, а не с 300h, так как нет директивы org 100h, выделяющей место под PSP.

Шаг 5. Загрузка COM модуля в основную память.



*Рисунок 9*

Ответы на контрольные вопросы.

Загрузка COM модуля в основную память:

1. *Какой формат загрузки COM модуля? С какого адреса располагается код?*

После загрузки COM-программы в память сегментные регистры указывают на начало PSP. Код располагается с адреса 100h.

1. *Что располагается с 0 адреса?*

С нулевого адреса располагается заголовок PSP.

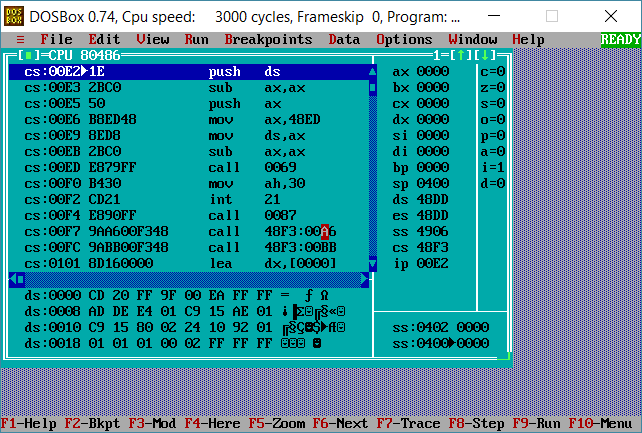
1. *Какие значения имеют сегментные регистры? На какие области памяти они указывают?*

48DD. Они указывают на начало PSP.

1. *Как определяется стек? Какую область памяти он занимает? Какие адреса?*

Стек определяется автоматически, указатель стека устанавливается на последнее слово сегмента (SP=FFFEh). Независимо от фактического размера программы, ей выделяется 64 Кбайт адресного пространства, всю нижнюю часть которого занимает стек.

Шаг 6. Загрузка «хорошего» EXE модуля в память.



*Рисунок 10*

Ответы на контрольные вопросы.

Загрузка «хорошего» EXE модуля в память:

1. *Как загружается «хороший» EXE? Какие значения имеют сегментные регистры?*

В области памяти строится PSP, считывается стандартная часть заголовка в память, определяется длина тела загрузочного модуля, начальный сегмент, загрузочный модуль считывается в начальный сегмент, таблица настройки считывается в рабочую память, определяются значения сегментных регистров. DS и ES устанавливаются на начало PSP, SS - на начало стека, CS - на начало сегмента кода.

1. *На что указывают регистры DS и ES?*

Изначально DS и ES указывают на начало PSP.

1. *Как определяется стек?*

Регистры SS и SP принимают значения, указанные в заголовке, потом к SS прибавляется сегментный адрес начального сегмента. Стек определяется с помощью директивы «DW 512 DUP(?)» в описании сегмента стека.

1. *Как определяется точка входа?*

Смещение точки входа в программу загружается в указатель команд IP. IP, а именно адрес, с которого начинается выполнение программы, определяется операндом директивы END, который называется точкой входа.

**Заключение:**

В ходе работы было проведено исследование различий в структурах исходных текстов модулей .COM и .EXE, структур файлов загрузочных модулей и способов их загрузки в основную память.

Приложение А. Файл com.asm:

CODESEG SEGMENT

ASSUME CS:CODESEG, DS:CODESEG, ES:NOTHING, SS:NOTHING

ORG 100H

START: JMP BEGIN

; ДАННЫЕ

TYPE\_PC db 'Тип IBM PC: ',0DH,0AH,'$'

MSDOS\_VER db 'Версия MS-DOS: . ',0DH,0AH,'$'

OEM\_NUM db 'Серийный номер OEM: ',0DH,0AH,'$'

USER\_NUM db 'Серийный номер пользователя: H',0DH,0AH,'$'

;ПРОЦЕДУРЫ

; Процедура печати строки

WriteMsg PROC NEAR

mov AH,09h

int 21h ; Вызов функции DOS по прерыванию

ret

WriteMsg ENDP

;-----------------------------------------------------

TETR\_TO\_HEX PROC near

and AL,0Fh

cmp AL,09

jbe NEXT

add AL,07

NEXT: add AL,30h

ret

TETR\_TO\_HEX ENDP

;-------------------------------

BYTE\_TO\_HEX PROC near

; байт в AL переводится в два символа шестн. числа в AX

push CX

mov AH,AL

call TETR\_TO\_HEX

xchg AL,AH

mov CL,4

shr AL,CL

call TETR\_TO\_HEX ;в AL старшая цифра

pop CX ;в AH младшая

ret

BYTE\_TO\_HEX ENDP

;-------------------------------

WRD\_TO\_HEX PROC near

;перевод в 16 с/с 16-ти разрядного числа

; в AX - число, DI - адрес последнего символа

push BX

mov BH,AH

call BYTE\_TO\_HEX

mov [DI],AH

dec DI

mov [DI],AL

dec DI

mov AL,BH

call BYTE\_TO\_HEX

mov [DI],AH

dec DI

mov [DI],AL

pop BX

ret

WRD\_TO\_HEX ENDP

;--------------------------------------------------

BYTE\_TO\_DEC PROC near

; перевод байта в 10с/с, SI - адрес поля младшей цифры

; AL содержит исходный байт

push AX

push CX

push DX

xor AH,AH

xor DX,DX

mov CX,10

loop\_bd: div CX

or DL,30h

mov [SI],DL

dec SI

xor DX,DX

cmp AX,10

jae loop\_bd

cmp AL,00h

je end\_l

or AL,30h

mov [SI],AL

end\_l: pop DX

pop CX

pop AX

ret

BYTE\_TO\_DEC ENDP

;-------------------------------

; Процедура определения типа PC

PC\_INFO PROC NEAR

push AX

push BX

push DX

push ES

mov BX,0F000H

mov ES,BX

mov AL,ES:[0FFFEH]

call BYTE\_TO\_HEX

lea BX,TYPE\_PC

mov [BX+12],AX

pop ES

pop DX

pop BX

pop AX

ret

PC\_INFO ENDP

; Процедура определения версии MS-DOS

SYSTEM\_INFO PROC NEAR

push AX

push SI

lea SI,MSDOS\_VER

add SI,16

call BYTE\_TO\_DEC

lea SI,MSDOS\_VER

add SI,19

mov AL,AH

call BYTE\_TO\_DEC

pop SI

pop AX

ret

SYSTEM\_INFO ENDP

; Процедура определения номера OEM

OEM\_INFO PROC NEAR

push AX

push BX

push SI

mov AL,BH

lea SI,OEM\_NUM

add SI,22

call BYTE\_TO\_DEC

pop SI

pop BX

pop AX

ret

OEM\_INFO ENDP

; Процедура определения номера пользователя

USER\_INFO PROC NEAR

push AX

push BX

push CX

push DI

mov AX,CX

lea DI,USER\_NUM

add DI,36

call WRD\_TO\_HEX

mov AL,BL

call BYTE\_TO\_HEX

lea DI,USER\_NUM

add DI,31

mov [DI],AX

pop DI

pop CX

pop BX

pop AX

ret

USER\_INFO ENDP

; КОД

BEGIN:

call PC\_INFO

mov AH,30H

INT 21H

call SYSTEM\_INFO

call OEM\_INFO

call USER\_INFO

; Вывод версии PC

lea DX,TYPE\_PC

call WriteMsg

; Вывод версии MS-DOS

lea DX,MSDOS\_VER

call WriteMsg

; Вывод номена OEM

lea DX,OEM\_NUM

call WriteMsg

; Вывод номера пользователя

lea DX,USER\_NUM

call WriteMsg

; Выход в DOS

xor AL,AL

mov AH,4Ch

int 21H

CODESEG ENDS

END START ;конец модуля, START - точка входа

Приложение Б. Файл exe.asm:

EOL EQU '$'

DATA SEGMENT

TYPE\_PC db 'PC type: ',0dh,0ah,'$'

SYS\_VER db 'MS-DOS version: . ',0dh,0ah,'$'

OEM\_NUM db 'OEM number: ',0dh,0ah,'$'

USER\_NUM db 'User serial number: H',0dh,0ah,'$'

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA, SS:AStack

Write\_msg PROC FAR

mov ah,09h

int 21h

ret

Write\_msg ENDP

AStack SEGMENT STACK

DW 512 DUP(?)

AStack ENDS

TETR\_TO\_HEX PROC FAR

and al,0fh

cmp al,09

jbe NEXT

add al,07

NEXT: add al,30h

ret

TETR\_TO\_HEX ENDP

;---------------------------

BYTE\_TO\_HEX PROC FAR

; байт в AL переводится в два символа шестн. числа в AX

push cx

mov al,ah

call TETR\_TO\_HEX

xchg al,ah

mov cl,4

shr al,cl

call TETR\_TO\_HEX ; в AL старшая цифра

pop cx ; в AH младшая

ret

BYTE\_TO\_HEX ENDP

;--------------------------

WRD\_TO\_HEX PROC FAR

; первод в 16 с/с 16-ти разрядного числа

; в AX - число, DI - адрес последнего символа

push bx

mov bh,ah

call BYTE\_TO\_HEX

mov [di],ah

dec di

mov [di],al

dec di

mov al,bh

xor ah,ah

call BYTE\_TO\_HEX

mov [di],ah

dec di

mov [di],al

pop bx

ret

WRD\_TO\_HEX ENDP

;----------------------------

BYTE\_TO\_DEC PROC FAR

; перевод в 10 с/с, SI - адрес поля младшей цифры

; AL содержит исходный байт

push cx

push dx

push ax

xor ah,ah

xor dx,dx

mov cx,10

loop\_bd:div cx

or dl,30h

mov [si],dl

dec si

xor dx,dx

cmp ax,10

jae loop\_bd

cmp ax,00h

jbe end\_l

or al,30h

mov [si],al

end\_l: pop ax

pop dx

pop cx

ret

BYTE\_TO\_DEC ENDP

;---------------------------

; Процедура определения типа PC

PC\_INFO PROC NEAR

push es

push bx

push ax

mov bx,0f000h

mov es,bx

mov ax,es:[0fffeh]

mov ah,al

call BYTE\_TO\_HEX

lea bx,TYPE\_PC

mov [bx+9],ax

pop ax

pop bx

pop es

ret

PC\_INFO ENDP

;----------------------------

; Процедура определения версии MS-DOS

SYSTEM\_INFO PROC NEAR

push AX

push SI

lea SI,SYS\_VER

add SI,16

call BYTE\_TO\_DEC

lea SI,SYS\_VER

add SI,19

mov AL,AH

call BYTE\_TO\_DEC

pop SI

pop AX

ret

SYSTEM\_INFO ENDP

;---------------------------

OEM\_INFO PROC FAR

; функция определяющая OEM

push ax

push bx

push si

mov al,bh

lea si,OEM\_NUM

add si,14

call BYTE\_TO\_DEC

pop si

pop bx

pop ax

ret

OEM\_INFO ENDP

;--------------------------

USER\_INFO PROC FAR

push ax

push bx

push cx

push si

mov al,bl

call BYTE\_TO\_HEX

lea di,USER\_NUM

add di,22

mov [di],AX

mov ax,cx

lea di,USER\_NUM

add di,27

call WRD\_TO\_HEX

pop si

pop cx

pop bx

pop ax

ret

USER\_INFO ENDP

Main PROC FAR

push DS

sub AX,AX

push AX

mov AX,DATA

mov DS,AX

sub AX,AX

call PC\_INFO

mov ah,30h

int 21h

call SYSTEM\_INFO

call OEM\_INFO

call USER\_INFO

; выводим информацию

lea dx,TYPE\_PC

call Write\_msg

lea dx,SYS\_VER

call Write\_msg

lea dx,OEM\_NUM

call Write\_msg

lea dx,USER\_NUM

call Write\_msg

; выход в DOS

xor al,al

mov ah,3Ch

int 21h

ret

Main ENDP

CODE ENDS

END Main