**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. 8382 |  | Ефимова М.А. |
| Преподаватель |  | Ефремов М.А, |

Санкт-Петербург

2020

**Цель работы.**

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

**Ход выполнения.**

1. Для определения типа PC и версии системы были написаны тексты .COM и .EXE модулей (см. Приложение А и B). Тип PC). Тип PC определяется предпоследним байтом ROM B). Тип PCIOS (табл. 1).

Таблица 1 – Соответствие кода и типа

|  |  |
| --- | --- |
| PC | FF |
|  |  |
| PC/XT | FE, FB). Тип PC |
|  |  |
| AT | FC |
|  |  |
| PS2 модель 30 | FA |
|  |  |
| PS2 модель 50 или 60 | FC |
|  |  |
| PS2 модель 80 | F8 |
|  |  |
| PCjr | FD |
|  |  |
| PC Convertible | F9 |
|  |  |

Версия системы определяется значением регистров AL, AH, B). Тип PCH, B). Тип PCL:CX, полученных после выполнения функции 30H прерывания 21H. (в AL – номер основной версии, в AH – номер модификации, в B). Тип PCH – серийный номер OEM, в B). Тип PCL:CX – 24-битовый серийный номер пользователя).

Результат выполнения .COM модуля представлен на рис. 1. Результат выполнения «плохого» .EXE модуля, полученного из исходного текста для .COM модуля, представлен на рис. 2. Результат выполнения «хорошего» .EXE модуля представлен на рис. 3.

2

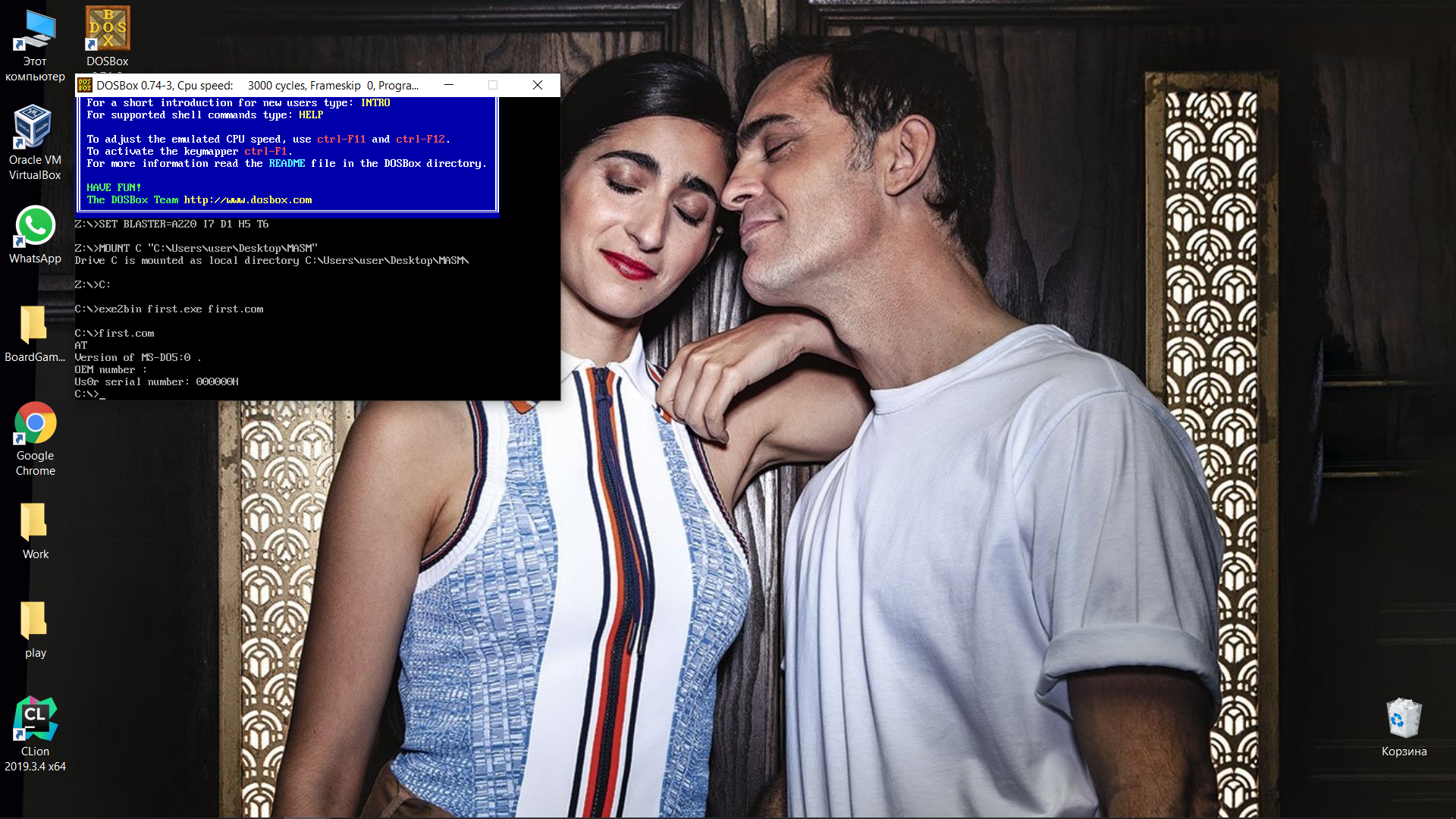


рис. 1 – результат выполнения .COM модуля

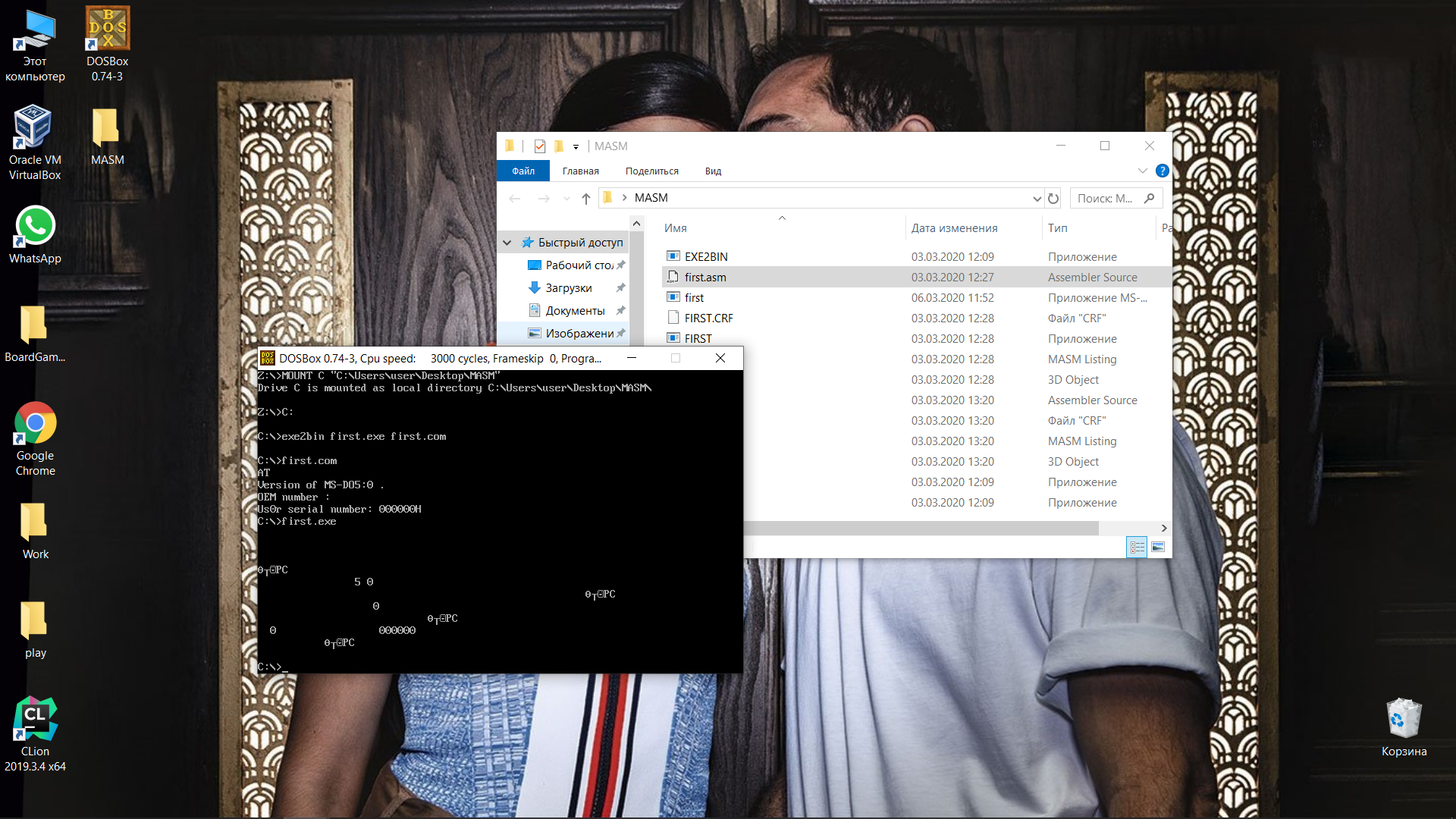


рис. 2 – результат выполнения «плохого» .EXE модуля

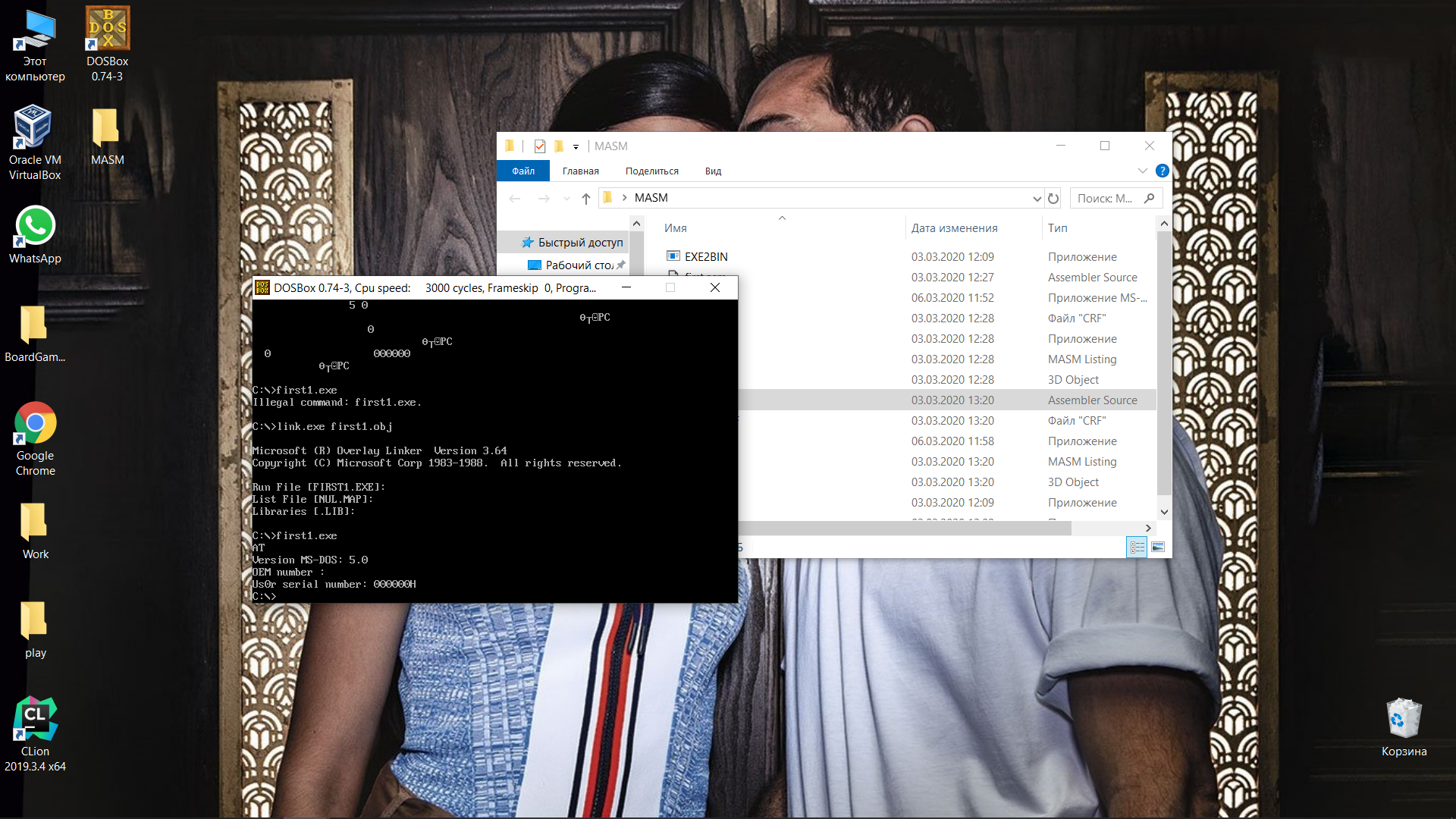


рис. 3 -результат выполнения «хорошего» .EXE модуля

3

**Отличия исходных текстов COM и EXE программ**

1. Сколько сегментов должна содержать COM-программа?

У файла типа .COM есть один сегмент команд.

1. Сколько сегментов должна содержать EXE-программа?

У файла типа .EXE может содержать различные сегменты. Он может содержать ряд сегментов, которые динамически перемещаются в пределах программной области.

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

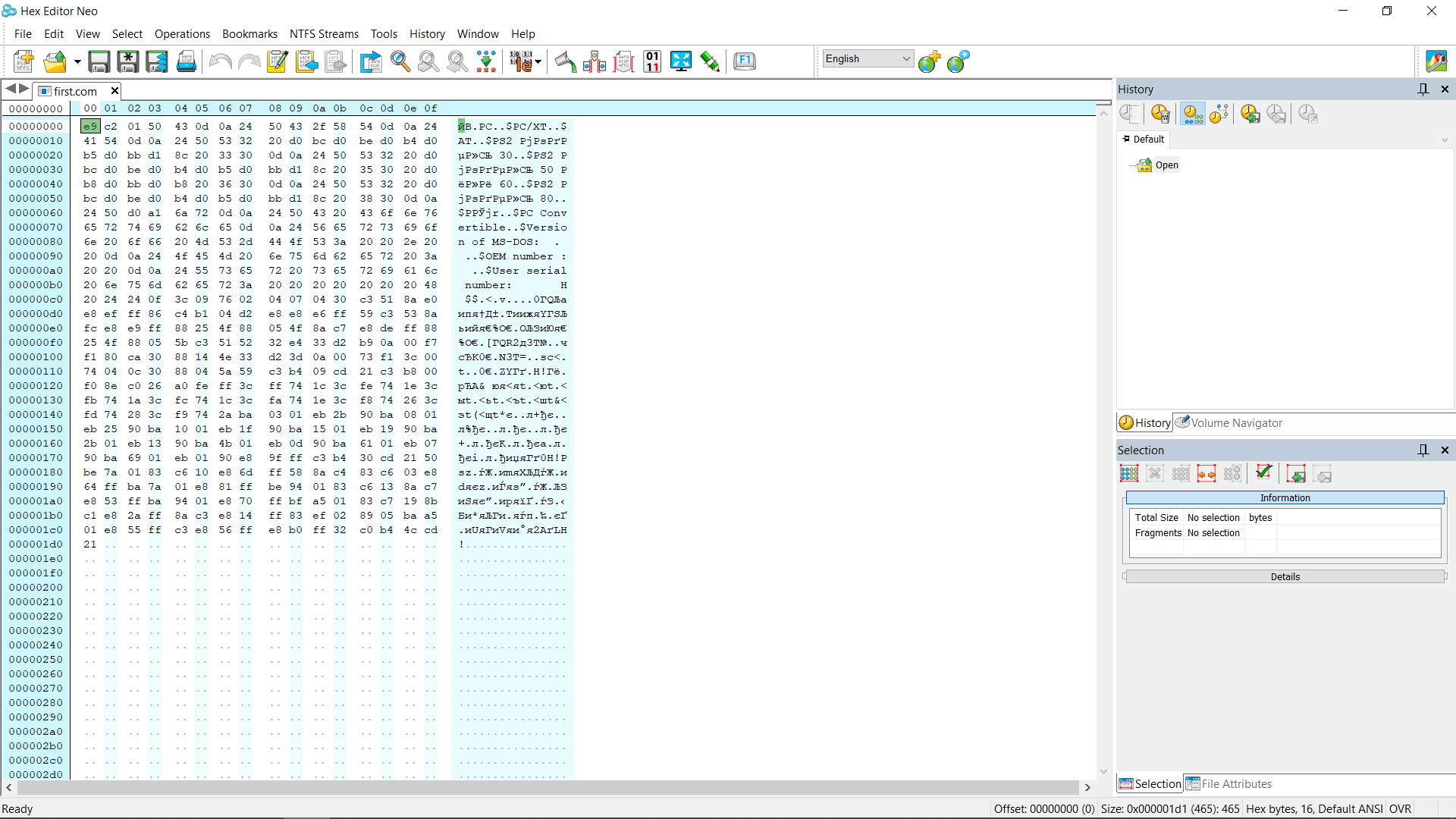
DOS передает управление в сегмент памяти, отведенный для команд, в точку со смещением 100H

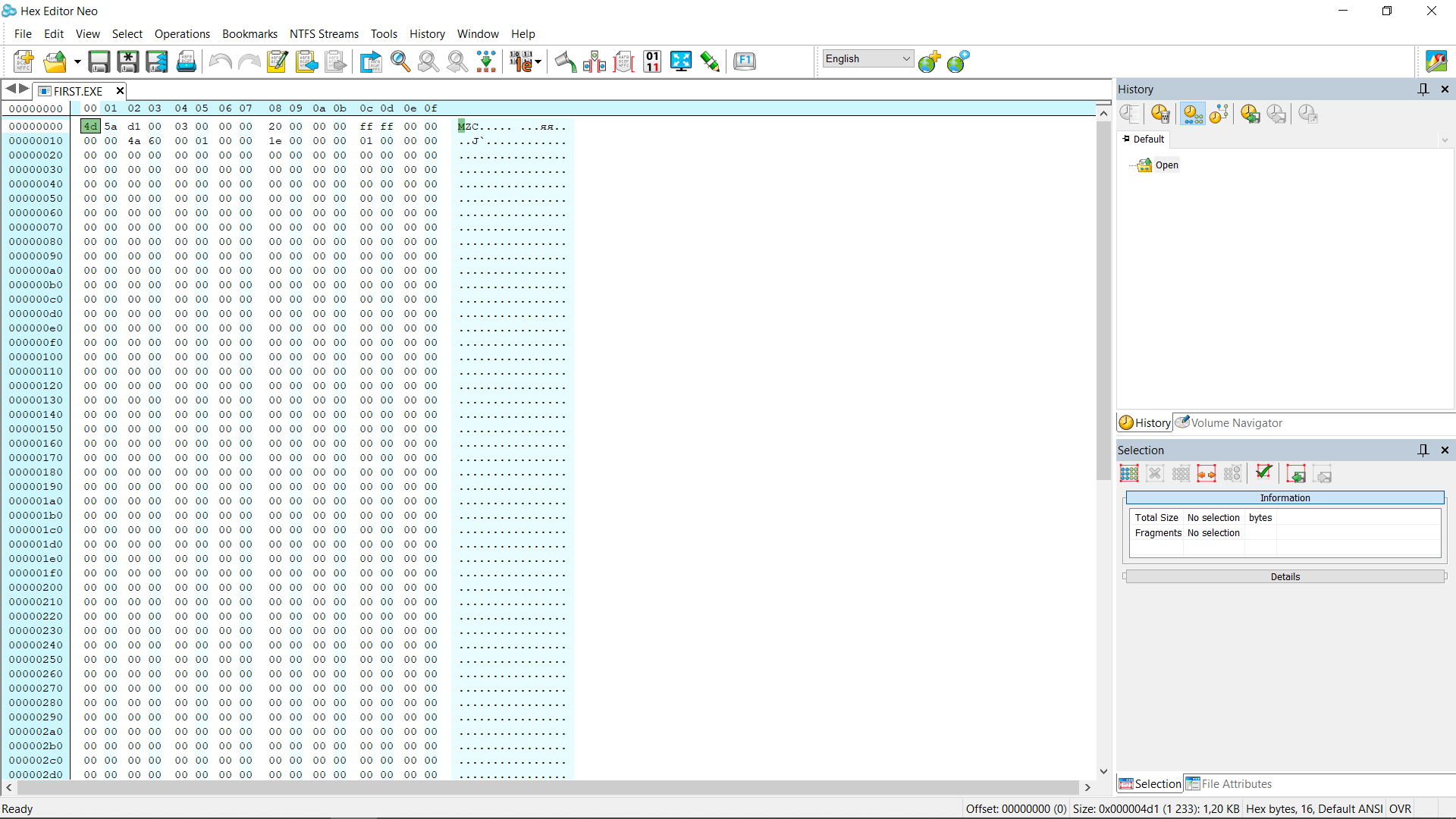
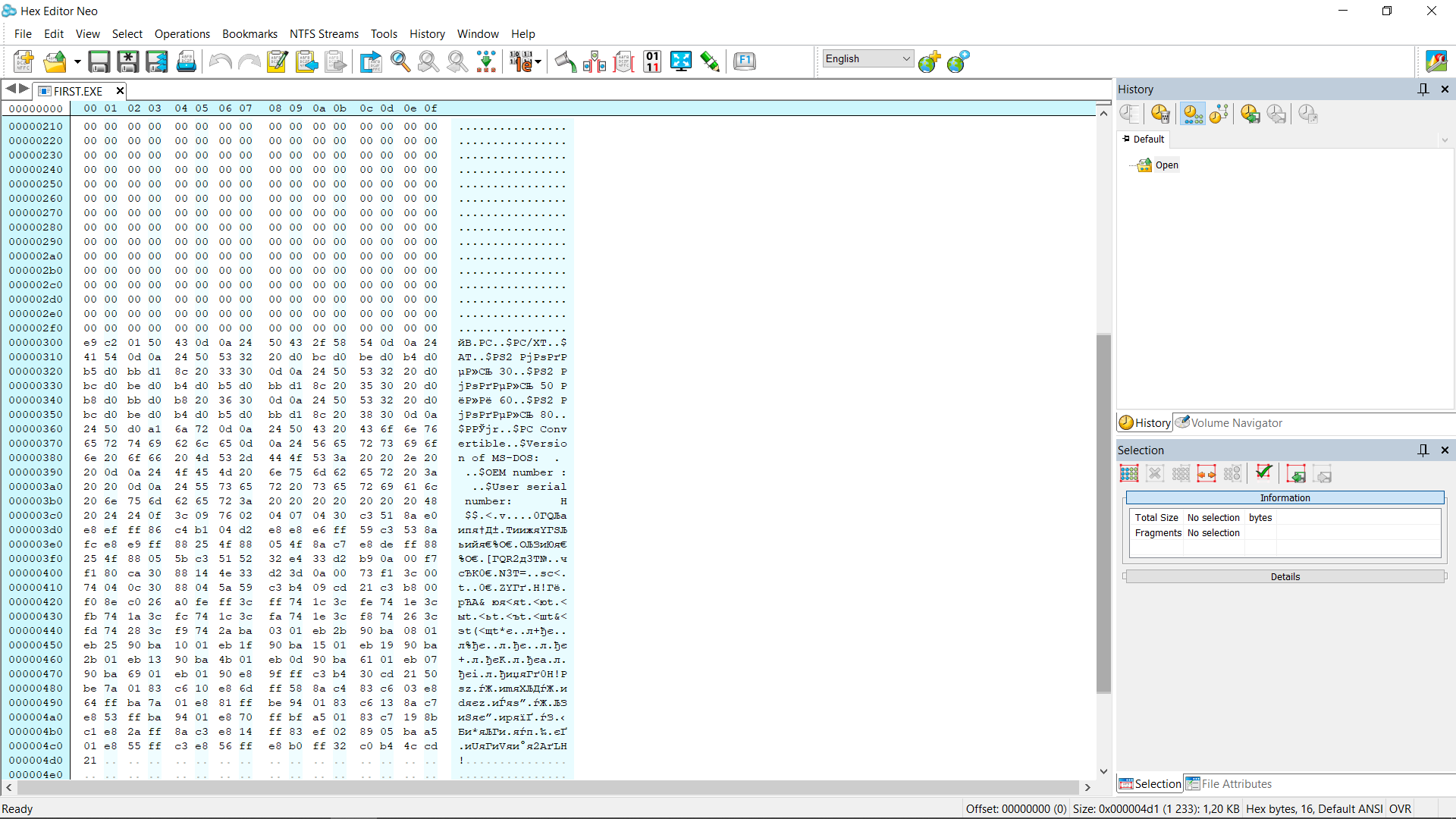
4. Все ли форматы команд можно использовать в COM-программе?

Файл типа .COM не является перемещаемым. У такого файла отсутствует информация, необходимая для перемещения. Вместо этого у программы, составляющей файл типа .COM, должен быть перемещаем сегмент команд.

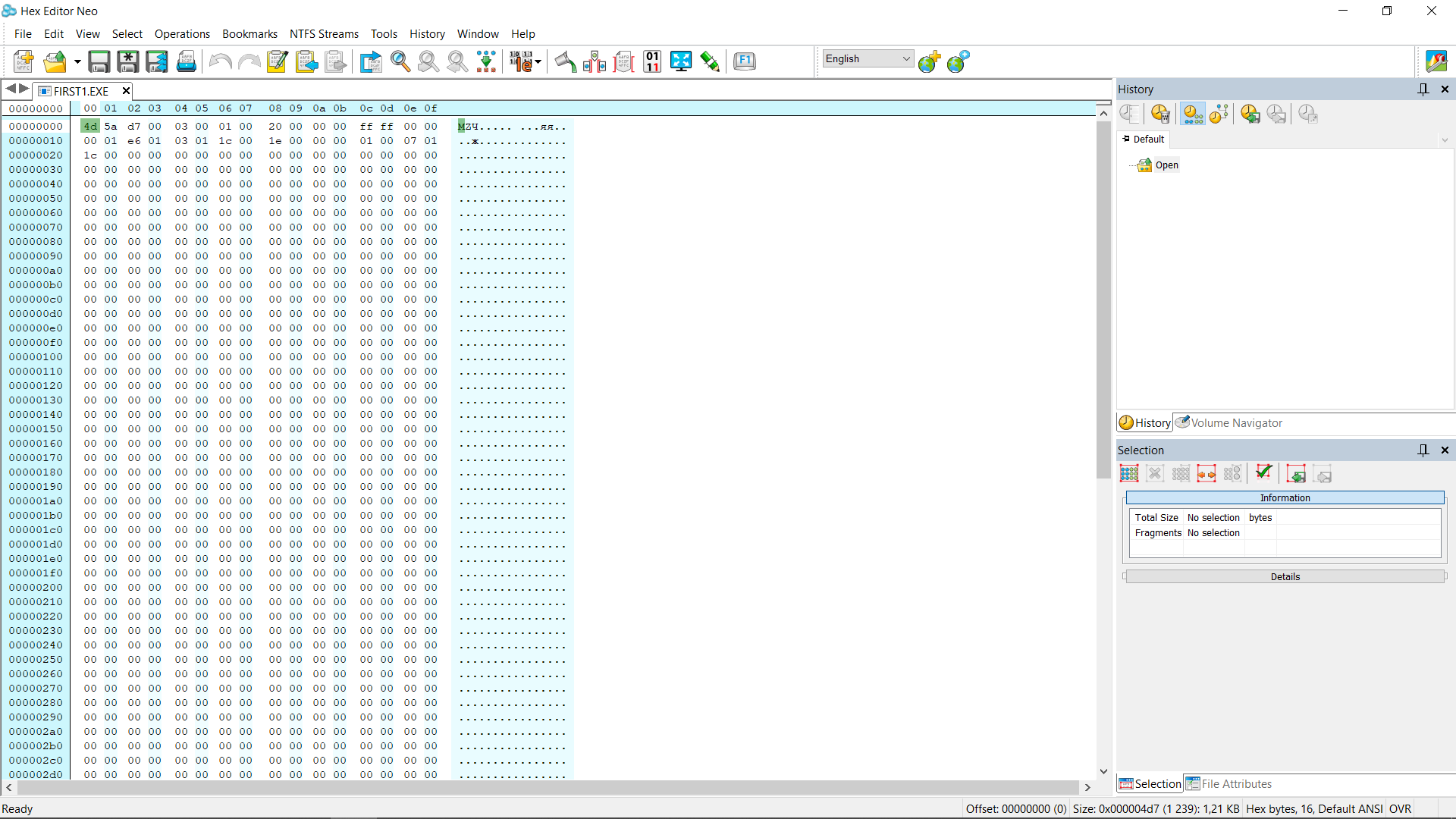
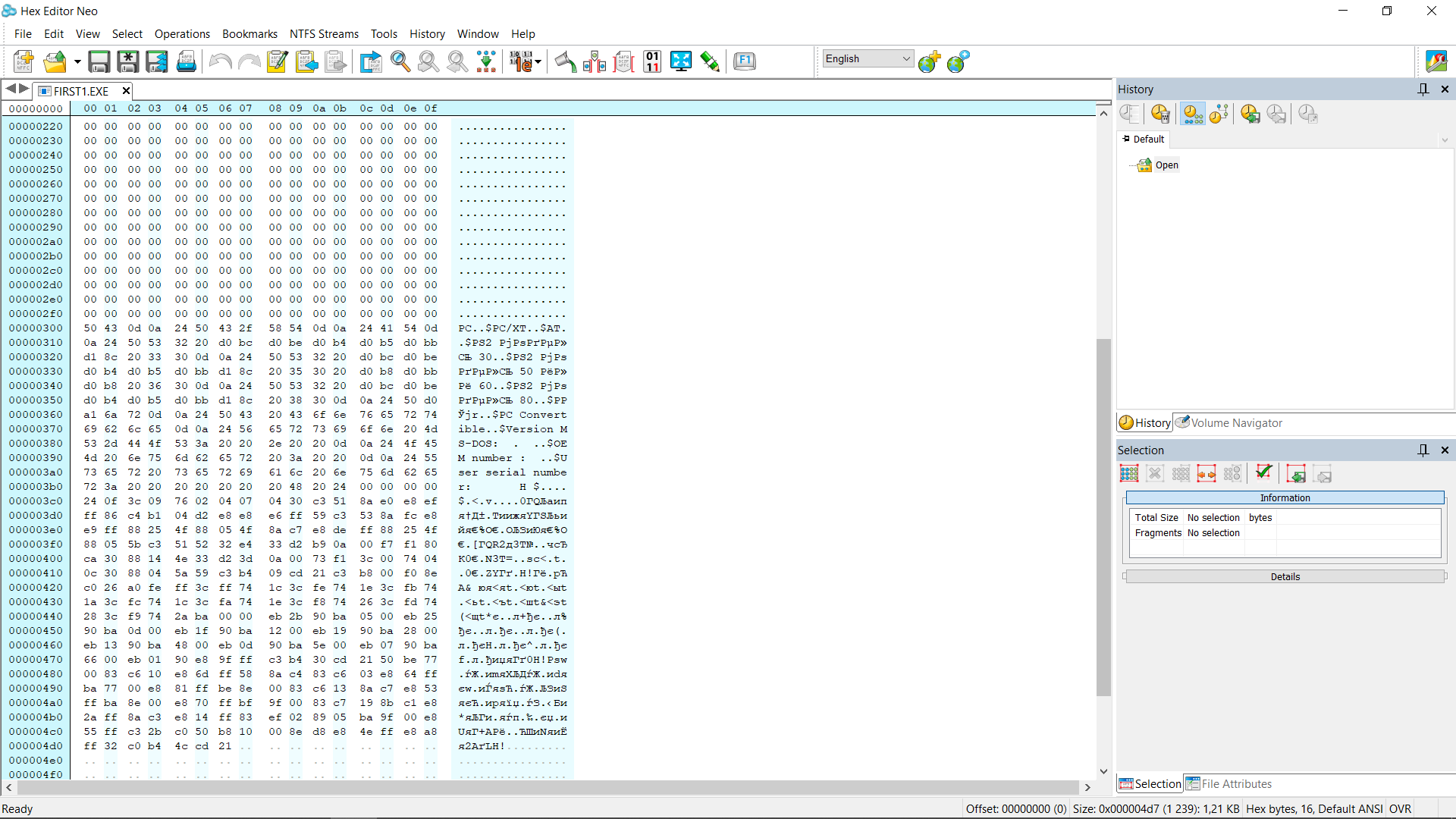
4

1. Файл загрузочного модуля .COM, «плохого» .EXE и «хорошего» в шестнадцатеричном виде представлены ниже.





5



6

**Отличия форматов файлов СOM и EXE модулей**

1. Какова структура файла COM? С какого адреса располагается

код?

Содержит один сегмент. Не превышает 64 Кб. Код располагается с адреса 0h, в первых 100h байт размещается PSP.

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

«Плохой» EXE файл, так же, как и COM файла, имеет один сегмент. Код располагается с адреса 300h. С адреса 0 располагается таблица настроек.

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

В «хорошем» EXE файле есть сегменты данных, кода и стека, в отличии от «плохого», в котором только один сегмент. Так же в «хорошем» EXE файле адресация кода начнется с 200h (размер PSP) + размер памяти, выделенной под стек. В «плохом» файле адресация всегда начинается с адреса 300h.

**Загрузка COM модуля в основную память**

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

- Смещение в сегменте команд равно 100h, сегментные регистры указывают на PSP.

1. Что располагается с адреса 0?

- С адреса 0 до адреса 100h располагается PSP.

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

- Сегментные регистры имеют значения 48DD и указывают на PSP.

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

- Значение регистра SP устанавливается (автоматически) так, чтобы он указывал на последнюю доступную в сегменте ячейку памяти (SP указывает на FFFE). Таким образом программа занимает начало, а стек - конец сегмента.

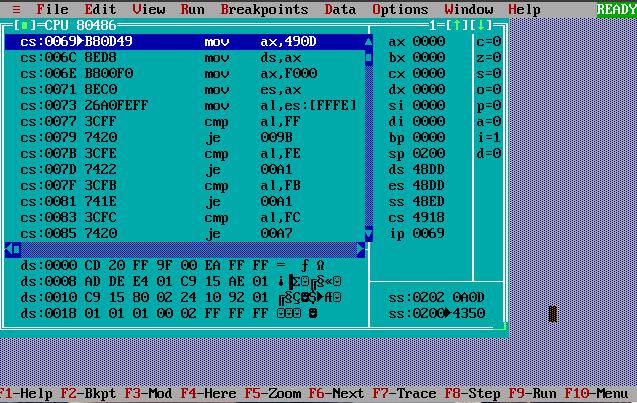


рис 9 – файл «хорошего» .EXE в отладчике

10

**Загрузка «хорошего» EXE модуля в основную память**

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

- Значения DS, ES (48DD) устанавливаются на начало PSP, SS (48ED) -- на начало сегмента стека, CS (4918) – на начало сегмента команд.

1. На что указывают регистры DS и ES?

* Регистры DS и ES указывают на начало PSP. 3. Как определяется стек?
* Стек определяется при помощи директивы .STACK или при помощи диррективы ASSUME, которая установит сегментый регистр SS на начало сегмента стека.

4. Как определяется точка входа?

* Точка входа определяется при помощи директивы END (за ней идет название функции или метки, с которой нужно начать выполнение программы).

11

**Выводы**

* ходе выполенения работы были изучены COM и EXE файлы и их различия. Так же были получены две программы typecom.com и typeexe.exe.

12

**ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД TYPECOM.ASM**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TESTPC | SEGMENT | | |  |
|  |  | ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING | | |
|  |  | ORG 100H ; резервирование места для PSP | |  |
| START: | JMP BEGIN | | |  |
| PC db | 'PC',0DH,0AH,'$' | | |  |
| XT db | 'PC/XT',0DH,0AH,'$' | | |  |
| tAT db | 'AT',0DH,0AH,'$' | | |  |
| PS2\_30 | db | 'PS2 model 30',0DH,0AH,'$' | |  |
| PS2\_80 | db | 'PS2 model 80',0DH,0AH,'$' | |  |
| PCJR db | 'PCjr',0DH,0AH,'$' | | |  |
| PC\_CONVERTIBLE db 'PC Convertible',0DH,0AH,'$' | | | |  |
| OTHER\_TYPE db | | | 'Other type:',0DH,0AH,'$' |  |
| END\_LINE db ' | | | ', 0DH,0AH,'$' |  |
| VERSION db | | 'Version: ',0DH,0AH,'$' | |  |
| VERS db ' $' | | |  |  |
| MODIFICATION db | | | '.$' |  |
| VERSION2 db | | 'Version <2.0',0DH,0AH,'$' | |  |
| OEM db | 'OEM number:',0DH,0AH,'$' | | |  |
| SERIAL\_NUMBER db 'User serial number:', 0AH, ' | | | | ', 0DH,0AH,'$' |

BEGIN:

mov ax,0F000H

mov es,ax

mov al,es:[0FFFEH]

cmp al, 0FFH

je itIsPC

cmp al, 0FEH

je itIsPC\_XT

cmp al, 0FBH

je itIsPC\_XT

cmp al, 0FCH

je itIsAT

cmp al, 0FAH

je itIsPS2\_30

cmp al, 0F8H

je itIsPS2\_80

cmp al, 0FDH

je itIsPCjr

cmp al, 0F9H

13

je itIsPCconvertible

cmp al, 0F9H

jne itIsOther

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

* Для вывода типа itIsPC:

mov dx, offset PC jmp writeType

itIsPC\_XT:

mov dx, offset XT

jmp writeType

itIsAT:

mov dx, offset tAT

jmp writeType

itIsPS2\_30:

mov dx, offset PS2\_30

jmp writeType

itIsPS2\_80:

mov dx, offset PS2\_80

jmp writeType

itIsPCjr:

mov dx, offset PCJR

jmp writeType

itIsPCconvertible:

mov dx, offset PC\_CONVERTIBLE

jmp writeType

itIsOther:

mov dx, offset OTHER\_TYPE

mov ah, 09h

int 21h

call BYTE\_TO\_HEX

call PRINT\_NUM

mov al, ah

call PRINT\_NUM

call PRINT\_END\_LINE

jmp OS\_VERSION

writeType:

mov ah, 09h

14

int 21h

jmp OS\_VERSION

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

;-------------------------------------------- ; Для вывода версии системы

OS\_VERSION:

mov ah, 30h

int 21h

printVer:

push ax

cmp al, 0

je ver2

mov dx, offset VERSION

push ax

mov ah, 09h

int 21h

pop ax

mov si, offset VERS

call BYTE\_TO\_DEC

add si, 1

mov dx, offset VERS

mov ah, 09h

int 21h

pop ax

jmp numMod

ver2:

mov dx, offset VERSION2

mov ah, 09h

int 21h

pop ax

numMod:

mov dx, offset MODIFICATION

push ax

mov ah, 09h

int 21h

pop ax

mov si, offset END\_LINE

mov al, ah

call BYTE\_TO\_DEC

add si, 1

15

mov dx, offset END\_LINE

mov ah, 09h

int 21h

numOEM:

mov dx, offset OEM

push ax

mov ah, 09h

int 21h

pop ax

mov si, offset END\_LINE

mov al, bh

call BYTE\_TO\_DEC

add si, 1

mov dx, offset END\_LINE

mov ah, 09h

int 21h

serialNumb:

mov di, offset SERIAL\_NUMBER

add di, 25

mov ax, cx

call WRD\_TO\_HEX

mov al, bl

call BYTE\_TO\_HEX

sub di, 2

mov [di], ax

mov dx, offset SERIAL\_NUMBER

mov ah, 09h

int 21h

xor al, al

mov ah, 4Ch

int 21h

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

PRINT\_END\_LINE PROC near

push ax

mov dx, offset END\_LINE

mov ah, 09h

int 21h

pop ax

PRINT\_END\_LINE ENDP

16

PRINT\_NUM PROC near

* вывод al push ax mov dx, ax mov ah, 02h int 21h pop ax ret

PRINT\_NUM 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 шест. 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

17

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

ret

BYTE\_TO\_DEC ENDP

TESTPC ENDS

END START

18

**ПРИЛОЖЕНИЕ B**

**ИСХОДНЫЙ КОД TYPEEXE.ASM**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| AStack | SEGMENT | | STACK | |  |  |
|  |  | DW 100h DUP(?) | | | |  |
| AStack | ENDS |  |  |  |  |  |
| DATA | SEGMENT |  |  |  |  |  |
|  | PC db | 'PC',0DH,0AH,'$' | | | |  |
|  | XT db | 'PC/XT',0DH,0AH,'$' | | | |  |
|  | tAT db | 'AT',0DH,0AH,'$' | | | |  |
|  | PS2\_30 | db | 'PS2 model 30',0DH,0AH,'$' | | |  |
|  | PS2\_80 | db | 'PS2 model 80',0DH,0AH,'$' | | |  |
|  | PCJR db | | 'PCjr',0DH,0AH,'$' | | |  |
|  | PC\_CONVERTIBLE db | | | | 'PC Convertible',0DH,0AH,'$' |  |
|  | OTHER\_TYPE db | | | 'Other type:',0DH,0AH,'$' | |  |
|  | END\_LINE db ' | | |  | ', 0DH,0AH,'$' |  |
|  | VERSION db 'Version: ',0DH,0AH,'$' | | | | |  |
|  | VERS db ' $' | | |  |  |  |
|  | MODIFICATION | | | db | '.$' |  |
|  | VERSION2 db | | | 'Version <2.0',0DH,0AH,'$' | |  |
|  | OEM db | 'OEM number:',0DH,0AH,'$' | | | |  |
|  | SERIAL\_NUMBER db 'User serial number:', 0AH, ' | | | | | ', |

0DH,0AH,'$'

DATA ENDS

CODE SEGMENT

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

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

PRINT\_END\_LINE PROC near

push ax

mov dx, offset END\_LINE

mov ah, 09h

int 21h

pop ax

PRINT\_END\_LINE ENDP

PRINT\_NUM PROC near

* вывод al push ax mov dx, ax mov ah, 02h int 21h

19

pop ax

ret

PRINT\_NUM 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 шест. 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 в 10с/с, SI - адрес поля младшей цифры

20

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

ret

BYTE\_TO\_DEC ENDP

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

Main PROC FAR

mov ax, DATA

mov ds, ax

mov ax,0F000H

mov es,ax

mov al,es:[0FFFEH]

cmp al, 0FFH

je itIsPC

cmp al, 0FEH

je itIsPC\_XT

cmp al, 0FBH

je itIsPC\_XT

cmp al, 0FCH

je itIsAT

cmp al, 0FAH

je itIsPS2\_30

cmp al, 0F8H

je itIsPS2\_80

cmp al, 0FDH

je itIsPCjr

cmp al, 0F9H

21

je itIsPCconvertible

cmp al, 0F9H

jne itIsOther

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

* Для вывода типа itIsPC:

mov dx, offset PC jmp writeType

itIsPC\_XT:

mov dx, offset XT

jmp writeType

itIsAT:

mov dx, offset tAT

jmp writeType

itIsPS2\_30:

mov dx, offset PS2\_30

jmp writeType

itIsPS2\_80:

mov dx, offset PS2\_80

jmp writeType

itIsPCjr:

mov dx, offset PCJR

jmp writeType

itIsPCconvertible:

mov dx, offset PC\_CONVERTIBLE

jmp writeType

itIsOther:

mov dx, offset OTHER\_TYPE

mov ah, 09h

int 21h

call BYTE\_TO\_HEX

call PRINT\_NUM

mov al, ah

call PRINT\_NUM

call PRINT\_END\_LINE

jmp OS\_VERSION

writeType:

mov ah, 09h

22

int 21h

jmp OS\_VERSION

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

;-------------------------------------------- ; Для вывода версии системы

OS\_VERSION:

mov ah, 30h

int 21h

printVer:

push ax

cmp al, 0

je ver2

mov dx, offset VERSION

push ax

mov ah, 09h

int 21h

pop ax

mov si, offset VERS

call BYTE\_TO\_DEC

add si, 1

mov dx, offset VERS

mov ah, 09h

int 21h

pop ax

jmp numMod

ver2:

mov dx, offset VERSION2

mov ah, 09h

int 21h

pop ax

numMod:

mov dx, offset MODIFICATION

push ax

mov ah, 09h

int 21h

pop ax

mov si, offset END\_LINE

mov al, ah

call BYTE\_TO\_DEC

add si, 1

23

mov dx, offset END\_LINE

mov ah, 09h

int 21h

numOEM:

mov dx, offset OEM

push ax

mov ah, 09h

int 21h

pop ax

mov si, offset END\_LINE

mov al, bh

call BYTE\_TO\_DEC

add si, 1

mov dx, offset END\_LINE

mov ah, 09h

int 21h

serialNumb:

mov di, offset SERIAL\_NUMBER

add di, 25

mov ax, cx

call WRD\_TO\_HEX

mov al, bl

call BYTE\_TO\_HEX

sub di, 2

mov [di], ax

mov dx, offset SERIAL\_NUMBER

mov ah, 09h

int 21h

xor al, al

mov ah, 4Ch

int 21h

Main ENDP

CODE ENDS

END Main

24