

**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**

Студент гр. 8382

Мирончик П.Д.

Преподаватель

\_\_\_\_\_

\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2019

## 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 работы.

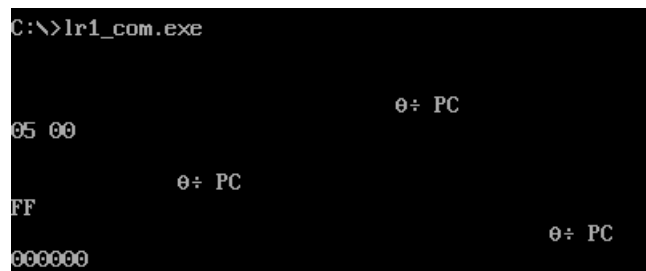
Был написан “хороший” .com модуль, который определяет тип PC и версию системы. Программа считывает содержимое предпоследнего байта ROM BIOS, сопоставляет его со значением из таблицы и выводит соответствующий тип PC. Затем используется функция 30h прерывания 21h для определения версии системы (формируется сорока формата xx.yy), серийного номера OEM и серийного номера пользователя (последние выводятся посимвольно в HEX формате).

Из этого текста были скомпилированы “хороший” .com модуль и “плохой” .exe модуль.



```
C:\>lr1_com
PC Convertible
05.00
FF
000000
```

Рисунок 1 – результат исполнения хорошего .com модуля



```
C:\>lr1_com.exe
PC Convertible
05.00
FF
000000
```

Рисунок 2 – результат исполнения плохого .exe модуля

На основе хорошего .com текста был написан текст хорошего .exe модуля.



```
C:\>lr1_exe.exe
PC Convertible
05.00
FF
000000
```

### **Отличия исходных текстов .com и .exe программ**

1. Сколько сегментов должна содержать .com программа?

Только один сегмент.

2. Сколько сегментов должна содержать .exe программа?

EXE программа может содержать несколько сегментов, однако обязательным является только сегмент кода.

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

org – задает смещение адресации внутри кода. Первые 100 байт COM программы занимают управляющие структуры и при расчете адресов необходимо учитывать это смещение.

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

Нельзя использовать упрощенные директивы сегментации и команды, использующие сегменты (такие как mov reg, seg:data), т.к. отсутствует таблица настройки адресов.

### **Отличия форматов файлов COM и EXE модулей.**

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

Файл COM содержит только один сегмент, в отличие от EXE программ. Код располагается с адреса 0.

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

Плохой EXE содержит один сегмент. Код располагается с адреса 300h, а в адресе 0 располагается DOS заголовок – сигнатура MZ, обозначающую, что данный файл является EXE файлом, а также таблица настройки адресов.

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

В хорошем EXE код и данные поделены на сегменты, в плохом — это все единый сегмент. Также отличается содержимое заголовков: в заголовке хорошего EXE файла содержатся сигнатура файла, начальные значения сегментов SS, SP, CS, IP, размер файла, количество сегментов и т.д.

### Загрузка модулей с помощью отладчика TD.

```
File Edit View Run Breakpoints Data Options Window Help
[ ]-CPU 80486
cs:0100 jmp 01F9
cs:0103 50 push ax
cs:0104 43 inc bx
cs:0105 0D0A24 or ax,240A
cs:0108 50 push ax
cs:0109 43 inc bx
cs:010A 2F das
cs:010B 58 pop ax
cs:010C 54 push sp
cs:010D 0D0A24 or ax,240A
cs:0110 50 push ax
cs:0111 43 inc bx
cs:0112 2F das
ax 0000 c=0
bx 0000 z=0
cx 0000 s=0
dx 0000 o=0
si 0000 p=0
di 0000 a=0
bp 0000 i=1
sp FFFE d=0
ds 48DD
es 48DD
ss 48DD
cs 48DD
ip 0100
ds:0000 CD 20 FF 9F 00 EA FF FF = f 9
ds:0008 AD DE E4 01 C9 15 AE 01
ds:0010 C9 15 80 02 24 10 92 01
ds:0018 01 01 01 00 02 FF FF FF
ss:0000 20CD
ss:FFFE 0000
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu
```

Рисунок 4 – загруженный модуль COM

```
File Edit View Run Breakpoints Data Options Window Help
[ ]-CPU 80486
cs:006E mov ax,48FA
cs:0071 8ED8 mov ds,ax
cs:0073 B800F0 mov ax,F000
cs:0076 8EC0 mov es,ax
cs:0078 26A0FEFF mov al,es:[FFFE]
cs:007C E88CFF call 000B
cs:007F BA0000 mov dx,0000
cs:0082 B9FF00 mov cx,00FF
cs:0085 3BCB cmp cx,ax
cs:0087 7446 je 00CF
cs:0089 BA0500 mov dx,0005
cs:008C B9FE00 mov cx,00FE
cs:008F 3BCB cmp cx,ax
ax 0000 c=0
bx 0000 z=0
cx 0000 s=0
dx 0000 o=0
si 0000 p=0
di 0000 a=0
bp 0000 i=1
sp 00C8 d=0
ds 48DD
es 48DD
ss 48ED
cs 4900
ip 006E
ds:0000 CD 20 FF 9F 00 EA FF FF = f 9
ds:0008 AD DE E4 01 C9 15 AE 01
ds:0010 C9 15 80 02 24 10 92 01
ds:0018 01 01 01 00 02 FF FF FF
ss:00CA 0000
ss:00CB 0000
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu
```

### **Загрузка COM модуля в основную память.**

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

Код располагается с адреса 100h.

2. Что располагается с адреса 0?

С 0 адреса располагается PSP.

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

Сегментные регистры имеют одинаковое значение 48DDh и указывают на PSP.

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

Стек занимает всю доступную память, кроме PSP и кода. Регистр SP при этом в начале выполнения программы содержит последний адресуемый байт – FFFh.

### **Загрузка хорошего EXE модуля в память.**

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

Регистры DS и ES имеют значения 48DDh, а сегмент стека SS – 48Edh.

2. На что указывают регистры DS и ES?

На начало PSP (48DDh).

3. Как определяется стек?

С использованием директивы `SEGMENT` выделяется память под сегмент стека и при загрузке программы регистр `SP` инициализируется соответствующим значением.

#### 4. Как определяется точка входа?

Точку входа можно явно задать, указав с помощью директивы `END` точку входа в программу.

### Дополнительные вопросы.

1. Что произойдет при удалении строки `ASSUME` из `COM` программы при компиляции?

```
**Error** lr1_com.asm(88) Near jump or call to different CS
**Error** lr1_com.asm(90) Near jump or call to different CS
**Error** lr1_com.asm(103) Near jump or call to different CS
**Error** lr1_com.asm(123) Near jump or call to different CS
**Error** lr1_com.asm(128) Near jump or call to different CS
**Error** lr1_com.asm(133) Near jump or call to different CS
**Error** lr1_com.asm(138) Near jump or call to different CS
**Error** lr1_com.asm(143) Near jump or call to different CS
**Error** lr1_com.asm(148) Near jump or call to different CS
**Error** lr1_com.asm(153) Near jump or call to different CS
**Error** lr1_com.asm(158) Near jump or call to different CS
**Error** lr1_com.asm(163) Near jump or call to different CS
**Error** lr1_com.asm(175) Near jump or call to different CS
**Error** lr1_com.asm(180) Near jump or call to different CS
**Error** lr1_com.asm(191) Near jump or call to different CS
**Error** lr1_com.asm(202) Near jump or call to different CS
**Error** lr1_com.asm(204) Near jump or call to different CS
**Error** lr1_com.asm(206) Near jump or call to different CS
Error messages: 24
Warning messages: None
Passes: 1
Remaining memory: 472k
```

Видны множественные ошибки *Near jump or call to different CS* на строках, где используются разновидности *jmp* и вызов ближних (*NEAR*) функций. Из описания ошибки “Это сообщение об ошибке выдается при попытке осуществить ближний вызов или переход по адресу, определенному в области, где регистр *CS* указывает на другой сегмент” видно, что проблема в несоответствии сегмента кода (содержимого регистра) с адресом текущего сегмента.

### Вывод.

В ходе лабораторной работы были изучены особенности написания исходного кода для COM и EXE файлов, структура файлов загрузочных модулей, способы просмотра шестнадцатеричного вида файлов, а также способ их загрузки в основную память.

## ПРИЛОЖЕНИЕ А.

### КОД ИСХОДНОГО СОМ МОДУЛЯ

```
MAIN      SEGMENT
          ASSUME CS:MAIN, DS:MAIN, SS:NOTHING
          ORG 100H

START:    JMP BEGIN
; DATA
FF_name db 'PC',13,10,'$'
FE_name db 'PC/XT',13,10,'$'
FB_name db 'PC/XT',13,10,'$'
FC_name db 'AT',13,10,'$'
FA_name db 'PS2 model 30',13,10,'$'
F8_name db 'PS2 model 80',13,10,'$'
FD_name db 'PCjr',13,10,'$'
F9_name db 'PC Convertible',13,10,'$'
dos_version db '00.00',13,10,'$'
serial_number db 13,10,'Serial number OEM: $'
user_serial_number db 13,10,'User serial number: $'
ENDL db 13,10,'$'

; PROCEDURES
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
; input:      AL=F8h (число)
; output:     AL={f}, AH={8} (в фигурных скобках символы)
;
; переводит AL в два символа в 16-й сс в AX
; в AL находится старшая, в AH младшая цифры
          push    cx
          mov     ah,al
          call    TETR_TO_HEX
          xchg    al,ah
          mov     cl,4
          shr     al,cl
          call    TETR_TO_HEX
          pop     cx
          ret
BYTE_TO_HEX ENDP
;-----
WRD_TO_HEX PROC      near
; input:      AX=FH7Ah (число)
;
;             DI={адрес} (указатель на последний символ в памяти, куда
будет записан результат)
; output:     начиная с [DI-3] лежат символы числа в 16-й сс
;
;             AX не сохраняет начальное значение
```



```

;
; перевод AX в 16-ю сс
    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
; input:        AL=0Fh (число)
;               SI={адрес} (адрес поля младшей цифры)
;
; перевод AL в 10-ю сс
    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,10
    je      end_l
    or      al,30h
    mov     [si],al
end_l:
    pop     ax
    pop     dx
    pop     cx
    ret
BYTE_TO_DEC     ENDP

WRITE_AL_HEX    PROC NEAR
    push    ax
    push    dx
    call    BYTE_TO_HEX

    mov     dl, al
    mov     al, ah
    mov     ah, 02h
    int     21h

```

```

        mov dl, al
        int 21h
        pop dx
        pop ax
        ret
WRITE_AL_HEX ENDP

;-----
; CODE
BEGIN:
        mov ax, 0F000h
        mov es, ax
        mov al, es: [0FFFEh]
        call BYTE_TO_HEX

        mov dx, offset FF_name
        mov cx, 0FFh
        cmp cx, ax
        je WRITE_TYPE

        mov dx, offset FE_name
        mov cx, 0FEh
        cmp cx, ax
        je WRITE_TYPE

        mov dx, offset FB_name
        mov cx, 0FBh
        cmp cx, ax
        je WRITE_TYPE

        mov dx, offset FC_name
        mov cx, 0FCh
        cmp cx, ax
        je WRITE_TYPE

        mov dx, offset FA_name
        mov cx, 0FAh
        cmp cx, ax
        je WRITE_TYPE

        mov dx, offset F8_name
        mov cx, 0F8h
        cmp cx, ax
        je WRITE_TYPE

        mov dx, offset FD_name
        mov cx, 0FDh
        cmp cx, ax
        je WRITE_TYPE

        mov dx, offset F9_name
        mov cx, 0F9h
        cmp cx, ax
        je WRITE_TYPE

WRITE_TYPE:
        mov ah, 09h
        int 21h

```

```

; Вывод версии системы
mov ah, 30h
int 21h

mov si, offset dos_version
add si, 1
call BYTE_TO_DEC

mov si, offset dos_version
add si, 4
mov al, ah
call BYTE_TO_DEC

mov dx, offset dos_version
mov ah, 09h
int 21h

; Серийный номер OEM
mov ah, 30h
int 21h

mov al, bh
call WRITE_AL_HEX

mov dx, offset ENDL
mov ah, 09h
int 21h

; Серийный номер пользователя
mov ah, 30h
int 21h

mov al, bl
call WRITE_AL_HEX
mov al, ch
call WRITE_AL_HEX
mov al, cl
call WRITE_AL_HEX

; Выход в DOS
xor     al, al
mov     ah, 4Ch
int     21h
MAIN    ENDS
        END START

```

## ПРИЛОЖЕНИЕ Б.

## КОД ИСХОДНОГО EXE МОДУЛЯ

```

STACK SEGMENT STACK
      DW 100 DUP(?)
STACK ENDS

```

DATA SEGMENT

```
FF_name      db    'PC',13,10,'$'
FE_name      db    'PC/XT',13,10,'$'
FB_name      db    'PC/XT',13,10,'$'
FC_name      db    'AT',13,10,'$'
FA_name      db    'PS2 model 30',13,10,'$'
F8_name      db    'PS2 model 80',13,10,'$'
FD_name      db    'PCjr',13,10,'$'
F9_name      db    'PC Convertible',13,10,'$'
dos_version  db    '00.00',13,10,'$'
ENDL         db    13,10,'$'
```

DATA ENDS

CODE	SEGMENT
------	---------

```
ASSUME CS:CODE, DS:DATA, SS:STACK
```

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

```

```
; input: AL=F8h (число)
```

```
; output:      AL={ f },  AH={ 8 }  (в фигурных скобках символы)
```

*i*

; переводит AL в два символа в 16-й сс в AX

; в AL находится старшая, в AH младшая цифры

push cx

```
mov     ah, al
```

```
call      TETR TO HEX
```

xchg       $\overline{al}, \overline{ah}$

```
mov     cl,4
```

shr      al,cl

```
call      TETR TO HEX
```

pop            cx

```
ret
```

```
BYTE TO HEX      ENDP
```

;

```
WRD TO HEX      PROC      near
```

```
; input:      AX=FH7Ah  (число)
```

; DI={адрес} (указатель на последний символ в памяти, куда будет записан результат)

```
; output:      начиная с [DI-3] лежат символы числа в 16-й сс
```

АХ не сохраняет начальное значение

*i*

```

; перевод AX в 16-ю сс
    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
; input:      AL=0Fh (число)
;            SI={адрес} (адрес поля младшей цифры)
;
; перевод AL в 10-ю сс
    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,10
    je      end_l
    or      al,30h
    mov     [si],al
end_l:
    pop     ax
    pop     dx
    pop     cx
    ret
BYTE_TO_DEC    ENDP

WRITE_AL_HEX    PROC NEAR
    push    ax
    push    dx
    call    BYTE_TO_HEX

    mov     dl, al
    mov     al, ah
    mov     ah, 02h
    int     21h

    mov     dl, al

```

```

        int 21h
        pop dx
        pop ax
        ret
WRITE_AL_HEX ENDP

;-----
; CODE
BEGIN    PROC    NEAR
        mov ax, DATA
        mov ds, ax

        mov ax, 0F000h
        mov es, ax
        mov al, es:[0FFFEh]
        call BYTE_TO_HEX

        mov dx, offset FF_name
        mov cx, 0FFh
        cmp cx, ax
        je WRITE_TYPE

        mov dx, offset FE_name
        mov cx, 0FEh
        cmp cx, ax
        je WRITE_TYPE

        mov dx, offset FB_name
        mov cx, 0FBh
        cmp cx, ax
        je WRITE_TYPE

        mov dx, offset FC_name
        mov cx, 0FCh
        cmp cx, ax
        je WRITE_TYPE

        mov dx, offset FA_name
        mov cx, 0FAh
        cmp cx, ax
        je WRITE_TYPE

        mov dx, offset F8_name
        mov cx, 0F8h
        cmp cx, ax
        je WRITE_TYPE

        mov dx, offset FD_name
        mov cx, 0FDh
        cmp cx, ax
        je WRITE_TYPE

        mov dx, offset F9_name
        mov cx, 0F9h
        cmp cx, ax
        je WRITE_TYPE

WRITE_TYPE:

```

```

    mov ah,09h
    int 21h

; Вывод версии системы
    mov ah, 30h
    int 21h

    mov si, offset dos_version
    add si,1
    call BYTE_TO_DEC

    mov si, offset dos_version
    add si, 4
    mov al, ah
    call BYTE_TO_DEC

    mov dx, offset dos_version
    mov ah, 09h
    int 21h

; Серийный номер OEM
    mov ah, 30h
    int 21h

    mov al, bh
    call WRITE_AL_HEX

    mov dx, offset ENDL
    mov ah, 09h
    int 21h

; Серийный номер пользователя
    mov ah, 30h
    int 21h

    mov al, bl
    call WRITE_AL_HEX
    mov al, ch
    call WRITE_AL_HEX
    mov al, cl
    call WRITE_AL_HEX

; Выход в DOS
    xor     al,al
    mov     ah,4Ch
    int     21h

BEGIN   ENDP
CODE    ENDS
        END BEGIN

```

## ПРИЛОЖЕНИЕ В.

### МОДУЛЬ СОМ В ШЕСТНАДЦАТЕРИЧНОМ ВИДЕ

0000000000:	E9 F6 00 50 43 0D 0A 24	50 43 2F 58 54 0D 0A 24
0000000010:	50 43 2F 58 54 0D 0A 24	41 54 0D 0A 24 50 53 32
0000000020:	20 6D 6F 64 65 6C 20 33	30 0D 0A 24 50 53 32 20
0000000030:	6D 6F 64 65 6C 20 38 30	0D 0A 24 50 43 6A 72 0D
0000000040:	0A 24 50 43 20 43 6F 6E	76 65 72 74 69 62 6C 65
0000000050:	0D 0A 24 30 30 2E 30 30	0D 0A 24 0D 0A 53 65 72
0000000060:	69 61 6C 20 6E 75 6D 62	65 72 20 4F 45 4D 3A 20
0000000070:	24 0D 0A 55 73 65 72 20	73 65 72 69 61 6C 20 6E
0000000080:	75 6D 62 65 72 3A 20 24	0D 0A 24 24 0F 3C 09 76
0000000090:	02 04 07 04 30 C3 51 8A	E0 E8 EF FF 86 C4 B1 04
00000000A0:	D2 E8 E8 E6 FF 59 C3 53	8A FC E8 E9 FF 88 25 4F
00000000B0:	88 05 4F 8A C7 E8 DE FF	88 25 4F 88 05 5B C3 51
00000000C0:	52 50 32 E4 33 D2 B9 0A	00 F7 F1 80 CA 30 88 14
00000000D0:	4E 33 D2 3D 0A 00 73 F1	3D 0A 00 74 04 0C 30 88
00000000E0:	04 58 5A 59 C3 50 52 E8	AC FF 8A D0 8A C4 B4 02
00000000F0:	CD 21 8A D0 CD 21 5A 58	C3 B8 00 F0 8E C0 26 A0
0000000100:	FE FF E8 91 FF BA 03 01	B9 FF 00 3B C8 74 46 BA
0000000110:	08 01 B9 FE 00 3B C8 74	3C BA 10 01 B9 FB 00 3B
0000000120:	C8 74 32 BA 18 01 B9 FC	00 3B C8 74 28 BA 1D 01
0000000130:	B9 FA 00 3B C8 74 1E BA	2C 01 B9 F8 00 3B C8 74
0000000140:	14 BA 3B 01 B9 FD 00 3B	C8 74 0A BA 42 01 B9 F9
0000000150:	00 3B C8 74 00 B4 09 CD	21 B4 30 CD 21 BE 53 01
0000000160:	83 C6 01 E8 59 FF BE 53	01 83 C6 04 8A C4 E8 4E
0000000170:	FF BA 53 01 B4 09 CD 21	B4 30 CD 21 8A C7 E8 64
0000000180:	FF BA 88 01 B4 09 CD 21	B4 30 CD 21 8A C3 E8 54
0000000190:	FF 8A C5 E8 4F FF 8A C1	E8 4A FF 32 C0 B4 4C CD
00000001A0:	21	



## ПРИЛОЖЕНИЕ Г.

## ПЛОХОЙ МОДУЛЬ EXE В ШЕСТНАДЦАТЕРИЧНОМ ВИДЕ

[illegible]

00000000240:	00	00	00	00	00	00	00	00		00	00	00	00	00	00	00	00
00000000250:	00	00	00	00	00	00	00	00		00	00	00	00	00	00	00	00
00000000260:	00	00	00	00	00	00	00	00		00	00	00	00	00	00	00	00
00000000270:	00	00	00	00	00	00	00	00		00	00	00	00	00	00	00	00
00000000280:	00	00	00	00	00	00	00	00		00	00	00	00	00	00	00	00
00000000290:	00	00	00	00	00	00	00	00		00	00	00	00	00	00	00	00
000000002A0:	00	00	00	00	00	00	00	00		00	00	00	00	00	00	00	00
000000002B0:	00	00	00	00	00	00	00	00		00	00	00	00	00	00	00	00
000000002C0:	00	00	00	00	00	00	00	00		00	00	00	00	00	00	00	00
000000002D0:	00	00	00	00	00	00	00	00		00	00	00	00	00	00	00	00
000000002E0:	00	00	00	00	00	00	00	00		00	00	00	00	00	00	00	00
000000002F0:	00	00	00	00	00	00	00	00		00	00	00	00	00	00	00	00
00000000300:	E9	F6	00	50	43	0D	0A	24		50	43	2F	58	54	0D	0A	24
00000000310:	50	43	2F	58	54	0D	0A	24		41	54	0D	0A	24	50	53	32
00000000320:	20	6D	6F	64	65	6C	20	33		30	0D	0A	24	50	53	32	20
00000000330:	6D	6F	64	65	6C	20	38	30		0D	0A	24	50	43	6A	72	0D
00000000340:	0A	24	50	43	20	43	6F	6E		76	65	72	74	69	62	6C	65
00000000350:	0D	0A	24	30	30	2E	30	30		0D	0A	24	0D	0A	53	65	72
00000000360:	69	61	6C	20	6E	75	6D	62		65	72	20	4F	45	4D	3A	20
00000000370:	24	0D	0A	55	73	65	72	20		73	65	72	69	61	6C	20	6E
00000000380:	75	6D	62	65	72	3A	20	24		0D	0A	24	24	0F	3C	09	76
00000000390:	02	04	07	04	30	C3	51	8A		E0	E8	EF	FF	86	C4	B1	04
000000003A0:	D2	E8	E8	E6	FF	59	C3	53		8A	FC	E8	E9	FF	88	25	4F
000000003B0:	88	05	4F	8A	C7	E8	DE	FF		88	25	4F	88	05	5B	C3	51
000000003C0:	52	50	32	E4	33	D2	B9	0A		00	F7	F1	80	CA	30	88	14
000000003D0:	4E	33	D2	3D	0A	00	73	F1		3D	0A	00	74	04	0C	30	88
000000003E0:	04	58	5A	59	C3	50	52	E8		AC	FF	8A	D0	8A	C4	B4	02
000000003F0:	CD	21	8A	D0	CD	21	5A	58		C3	B8	00	F0	8E	C0	26	A0
00000000400:	FE	FF	E8	91	FF	BA	03	01		B9	FF	00	3B	C8	74	46	BA
00000000410:	08	01	B9	FE	00	3B	C8	74		3C	BA	10	01	B9	FB	00	3B
00000000420:	C8	74	32	BA	18	01	B9	FC									

## ПРИЛОЖЕНИЕ Д.

### ХОРОШИЙ МОДУЛЬ EXE В ШЕСТНАДЦАТЕРИЧНОМ ВИДЕ

0000000000:	4D	5A	4B	00	03	00	01	00	20	00	00	00	FF	FF	00	00
0000000010:	C8	00	00	00	6E	00	13	00	3E	00	00	00	01	00	FB	50
0000000020:	6A	72	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000030:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	6F	00
0000000040:	13	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000050:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000060:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000070:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000080:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000090:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000000A0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000000B0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000000C0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000000D0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000000E0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000000F0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000100:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000110:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000120:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000130:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000140:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000150:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000160:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000170:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000180:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000190:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000001A0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000001B0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000001C0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000001D0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000001E0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000001F0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000200:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000210:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000220:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000230:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000240:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000250:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000260:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000270:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000280:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000290:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000002A0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000002B0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000002C0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000002D0:	50	43	0D	0A	24	50	43	2F	58	54	0D	0A	24	50	43	2F
00000002E0:	58	54	0D	0A	24	41	54	0D	0A	24	50	53	32	20	6D	6F
00000002F0:	64	65	6C	20	33	30	0D	0A	24	50	53	32	20	6D	6F	64
0000000300:	65	6C	20	38	30	0D	0A	24	50	43	6A	72	0D	0A	24	50
0000000310:	43	20	43	6F	6E	76	65	72	74	69	62	6C	65	0D	0A	24
0000000320:	30	30	2E	30	30	0D	0A	24	0D	0A	24	00	00	00	00	00
0000000330:	24	0F	3C	09	76	02	04	07	04	30	C3	51	8A	E0	E8	EF

0000000340:	FF	86	C4	B1	04	D2	E8	E8	E6	FF	59	C3	53	8A	FC	E8
0000000350:	E9	FF	88	25	4F	88	05	4F	8A	C7	E8	DE	FF	88	25	4F
0000000360:	88	05	5B	C3	51	52	50	32	E4	33	D2	B9	0A	00	F7	F1
0000000370:	80	CA	30	88	14	4E	33	D2	3D	0A	00	73	F1	3D	0A	00
0000000380:	74	04	0C	30	88	04	58	5A	59	C3	50	52	E8	AC	FF	8A
0000000390:	D0	8A	C4	B4	02	CD	21	8A	D0	CD	21	5A	58	C3	B8	0D
00000003A0:	00	8E	D8	B8	00	F0	8E	C0	26	A0	FE	FF	E8	8C	FF	BA
00000003B0:	00	00	B9	FF	00	3B	C8	74	46	BA	05	00	B9	FE	00	3B
00000003C0:	C8	74	3C	BA	0D	00	B9	FB	00	3B	C8	74	32	BA	15	00
00000003D0:	B9	FC	00	3B	C8	74	28	BA	1A	00	B9	FA	00	3B	C8	74
00000003E0:	1E	BA	29	00	B9	F8	00	3B	C8	74	14	BA	38	00	B9	FD
00000003F0:	00	3B	C8	74	0A	BA	3F	00	B9	F9	00	3B	C8	74	00	B4
0000000400:	09	CD	21	B4	30	CD	21	BE	50	00	83	C6	01	E8	54	FF
0000000410:	BE	50	00	83	C6	04	8A	C4	E8	49	FF	BA	50	00	B4	09
0000000420:	CD	21	B4	30	CD	21	8A	C7	E8	5F	FF	BA	58	00	B4	09
0000000430:	CD	21	B4	30	CD	21	8A	C3	E8	4F	FF	8A	C5	E8	4A	FF
0000000440:	8A	C1	E8	45	FF	32	C0	B4	4C	CD	21					