**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.



Рисунок 1 – результат исполнения хорошего .com модуля

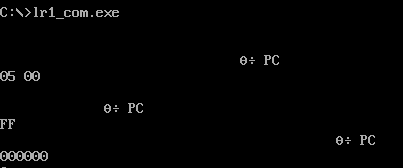


Рисунок 2 – результат исполнения плохого .exe модуля

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

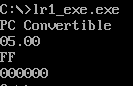


Рисунок 3 – вывод хорошего .exe модуля

**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 команды, связанные с адресацией сегментов, т.к. в COM программе есть только один сегмент.

**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 – сигнатуру MZ, обозначающую, что данный файл является EXE файлом.

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.

**Загрузка модулей с помощью отладчика TD.**

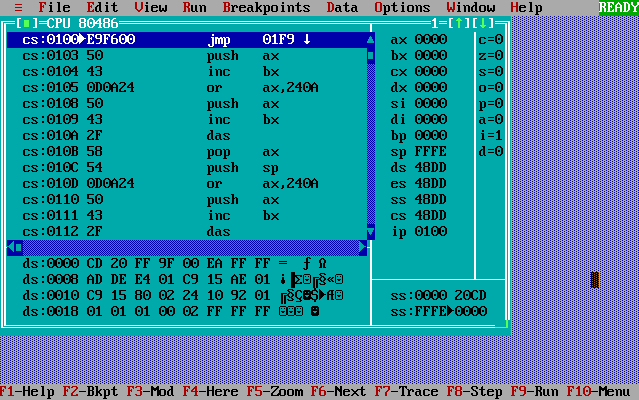


Рисунок 4 – загруженный модуль COM

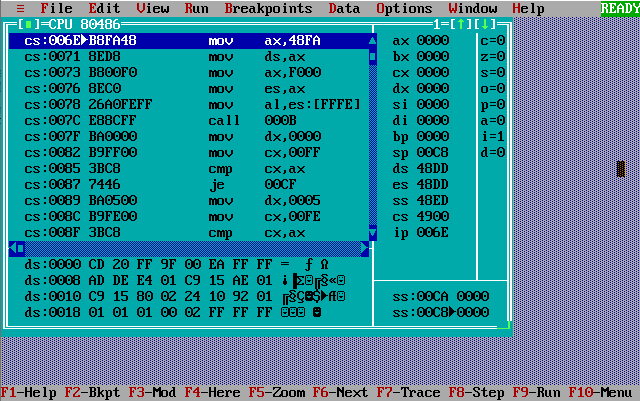


Рисунок 5 – хороший загруженный модуль EXE

**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 – FFFEh.

**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.

**Вывод.**

В ходе лабораторной работы были изучены особенности написание 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 COM 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

ASTACK SEGMENT STACK

    DW 100 DUP(?)

ASTACK 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:ASTACK

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   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 com 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

0000000000: 4D 5A A1 00 03 00 00 00 │ 20 00 00 00 FF FF 00 00

0000000010: 00 00 00 00 00 01 00 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 00 00

0000000040: 00 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

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: 00 00 00 00 00 00 00 00 │ 00 00 00 00 00 00 00 00

00000002E0: 00 00 00 00 00 00 00 00 │ 00 00 00 00 00 00 00 00

00000002F0: 00 00 00 00 00 00 00 00 │ 00 00 00 00 00 00 00 00

0000000300: E9 F6 00 50 43 0D 0A 24 │ 50 43 2F 58 54 0D 0A 24

0000000310: 50 43 2F 58 54 0D 0A 24 │ 41 54 0D 0A 24 50 53 32

0000000320: 20 6D 6F 64 65 6C 20 33 │ 30 0D 0A 24 50 53 32 20

0000000330: 6D 6F 64 65 6C 20 38 30 │ 0D 0A 24 50 43 6A 72 0D

0000000340: 0A 24 50 43 20 43 6F 6E │ 76 65 72 74 69 62 6C 65

0000000350: 0D 0A 24 30 30 2E 30 30 │ 0D 0A 24 0D 0A 53 65 72

0000000360: 69 61 6C 20 6E 75 6D 62 │ 65 72 20 4F 45 4D 3A 20

0000000370: 24 0D 0A 55 73 65 72 20 │ 73 65 72 69 61 6C 20 6E

0000000380: 75 6D 62 65 72 3A 20 24 │ 0D 0A 24 24 0F 3C 09 76

0000000390: 02 04 07 04 30 C3 51 8A │ E0 E8 EF FF 86 C4 B1 04

00000003A0: D2 E8 E8 E6 FF 59 C3 53 │ 8A FC E8 E9 FF 88 25 4F

00000003B0: 88 05 4F 8A C7 E8 DE FF │ 88 25 4F 88 05 5B C3 51

00000003C0: 52 50 32 E4 33 D2 B9 0A │ 00 F7 F1 80 CA 30 88 14

00000003D0: 4E 33 D2 3D 0A 00 73 F1 │ 3D 0A 00 74 04 0C 30 88

00000003E0: 04 58 5A 59 C3 50 52 E8 │ AC FF 8A D0 8A C4 B4 02

00000003F0: CD 21 8A D0 CD 21 5A 58 │ C3 B8 00 F0 8E C0 26 A0

0000000400: FE FF E8 91 FF BA 03 01 │ B9 FF 00 3B C8 74 46 BA

0000000410: 08 01 B9 FE 00 3B C8 74 │ 3C BA 10 01 B9 FB 00 3B

0000000420: C8 74 32 BA 18 01 B9 FC │ 00 3B C8 74 28 BA 1D 01

0000000430: B9 FA 00 3B C8 74 1E BA │ 2C 01 B9 F8 00 3B C8 74

0000000440: 14 BA 3B 01 B9 FD 00 3B │ C8 74 0A BA 42 01 B9 F9

0000000450: 00 3B C8 74 00 B4 09 CD │ 21 B4 30 CD 21 BE 53 01

0000000460: 83 C6 01 E8 59 FF BE 53 │ 01 83 C6 04 8A C4 E8 4E

0000000470: FF BA 53 01 B4 09 CD 21 │ B4 30 CD 21 8A C7 E8 64

0000000480: FF BA 88 01 B4 09 CD 21 │ B4 30 CD 21 8A C3 E8 54

0000000490: FF 8A C5 E8 4F FF 8A C1 │ E8 4A FF 32 C0 B4 4C CD

00000004A0: 21 │

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