

**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

\_\_\_\_\_

Колногоров Д.Г.

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.

### **Выполнение работы.**

Был написан текст исходного **.COM** модуля (представлен в приложении А), который определяет тип PC и версию системы. Ассемблерная программа читает 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, серийного номера OEM и серийного номера пользователя. Далее был получен «хороший» **.COM** модуль, а также из его исходного кода был построен «плохой» **.EXE** модуль. Был построен «хороший» **.EXE** модуль, который выполняет те же функции.



```
C:\>LR1_COM.COM
IBM PC type: AT
MSDOS version: 05.00
OEM serial number: FF
User serial number: 000000
```

Рисунок 1 — результат исполнения COM модуля

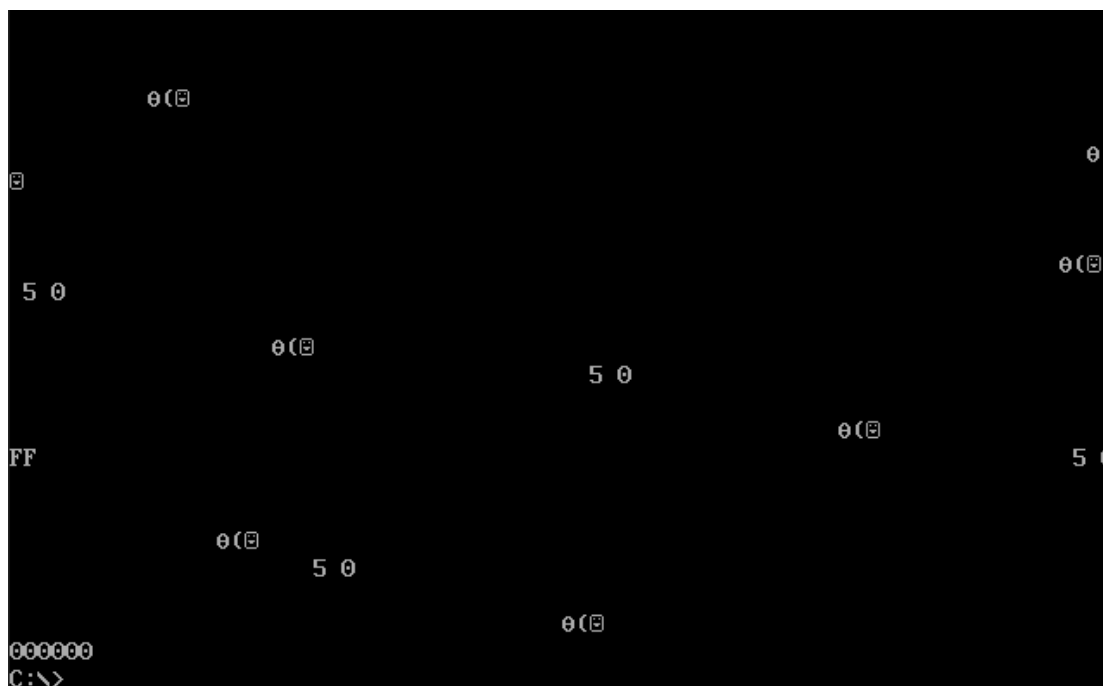


Рисунок 2 — результат исполнения «плохого» EXE модуля

```

C:\>LR1_EXE.EXE
IBM PC type: AT
MSDOS version: 05.00
OEM serial number: FF
User serial number: 000000

```

Рисунок 3 — результат исполнения «хорошего» EXE модуля

### **Отличия исходных текстов COM и EXE программ.**

1) Сколько сегментов должна содержать COM-программа?

COM-программа должна содержать один сегмент.

2) EXE-программа?

EXE-программа может содержать несколько сегментов.

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

COM-программа должна содержать директиву `ORG 100H`, которая задаёт смещение для адресации в 100H байт, так как адреса с 0 до 100H занимает PSP. Также необходима директива `ASSUME` для инициализации регистров.

```
LR1_COM.ASM(82): error A2062: Missing or unreachable CS
LR1_COM.ASM(89): error A2062: Missing or unreachable CS
LR1_COM.ASM(100): error A2062: Missing or unreachable CS
LR1_COM.ASM(106): error A2062: Missing or unreachable CS
LR1_COM.ASM(116): error A2062: Missing or unreachable CS
LR1_COM.ASM(120): error A2062: Missing or unreachable CS
LR1_COM.ASM(126): error A2062: Missing or unreachable CS
LR1_COM.ASM(130): error A2062: Missing or unreachable CS
LR1_COM.ASM(134): error A2062: Missing or unreachable CS
LR1_COM.ASM(138): error A2062: Missing or unreachable CS
LR1_COM.ASM(142): error A2062: Missing or unreachable CS
LR1_COM.ASM(146): error A2062: Missing or unreachable CS
LR1_COM.ASM(150): error A2062: Missing or unreachable CS
LR1_COM.ASM(153): error A2062: Missing or unreachable CS
LR1_COM.ASM(156): error A2062: Missing or unreachable CS
LR1_COM.ASM(158): error A2062: Missing or unreachable CS
LR1_COM.ASM(178): error A2062: Missing or unreachable CS
LR1_COM.ASM(189): error A2062: Missing or unreachable CS

49958 + 453207 Bytes symbol space free

0 Warning Errors
24 Severe Errors
```

Рисунок 4 — результат компиляции при отсутствии директивы ASSUME

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

В COM-программах нельзя использовать команды, связанные с адресацией сегментов, так как в COM-программах отсутствует таблица настройки, необходимая для корректировки адресов сегментов в программе при её загрузке (как в EXE-программах).

Содержимое всех созданных загрузочных модулей представлены в приложениях В, Г и Д.

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

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

Файл COM содержит один сегмент. Код располагается с адреса 0, но директива ORG 100H указывает компилятору о необходимости смещения адресации на 100H байтов.

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

«Плохой» EXE также содержит один сегмент. Код располагается с адреса 300H. С адреса 0 располагается DOS-заголовок (первые два байта содержат сигнатуру MZ обозначающую, что данный файл является EXE файлом).

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

В «хорошем» EXE код и данные поделены на сегменты, а в «плохом» - это единый сегмент. Код также начинается с адреса 300H. С адреса 0 располагается DOS-заголовок (200h байт), а затем сегмент стека (100h байт).

Модули COM и EXE были загружены с помощью отладчика TD.

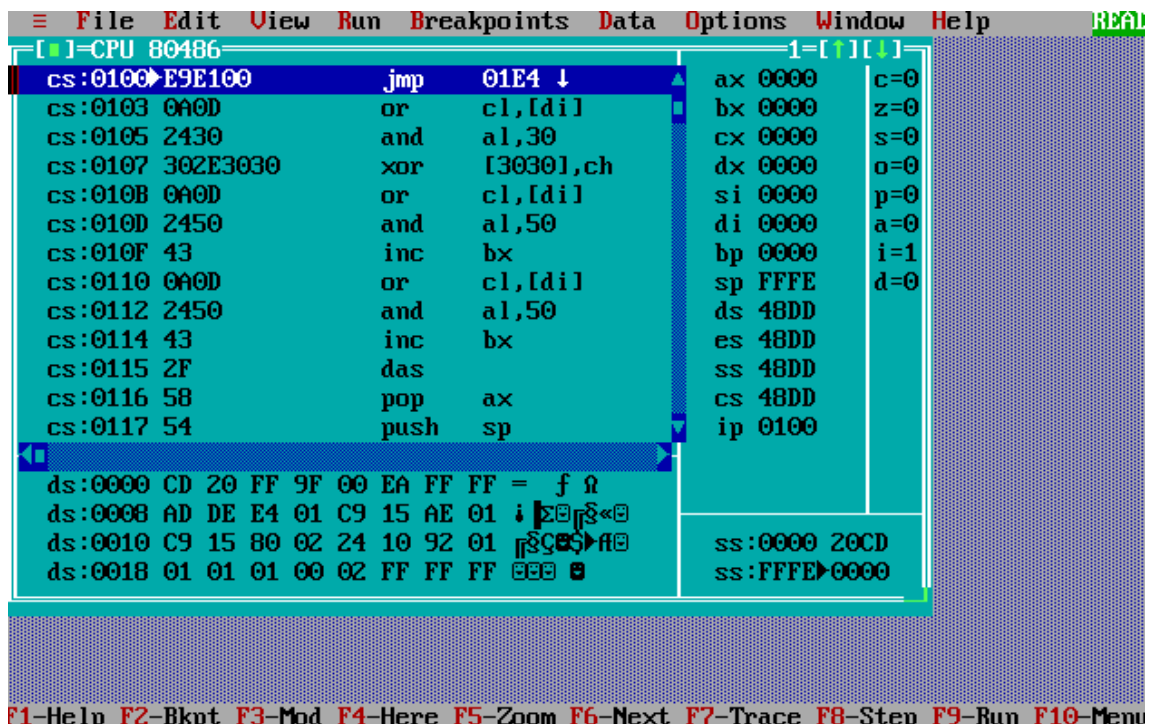


Рисунок 5 — загруженный модуль COM

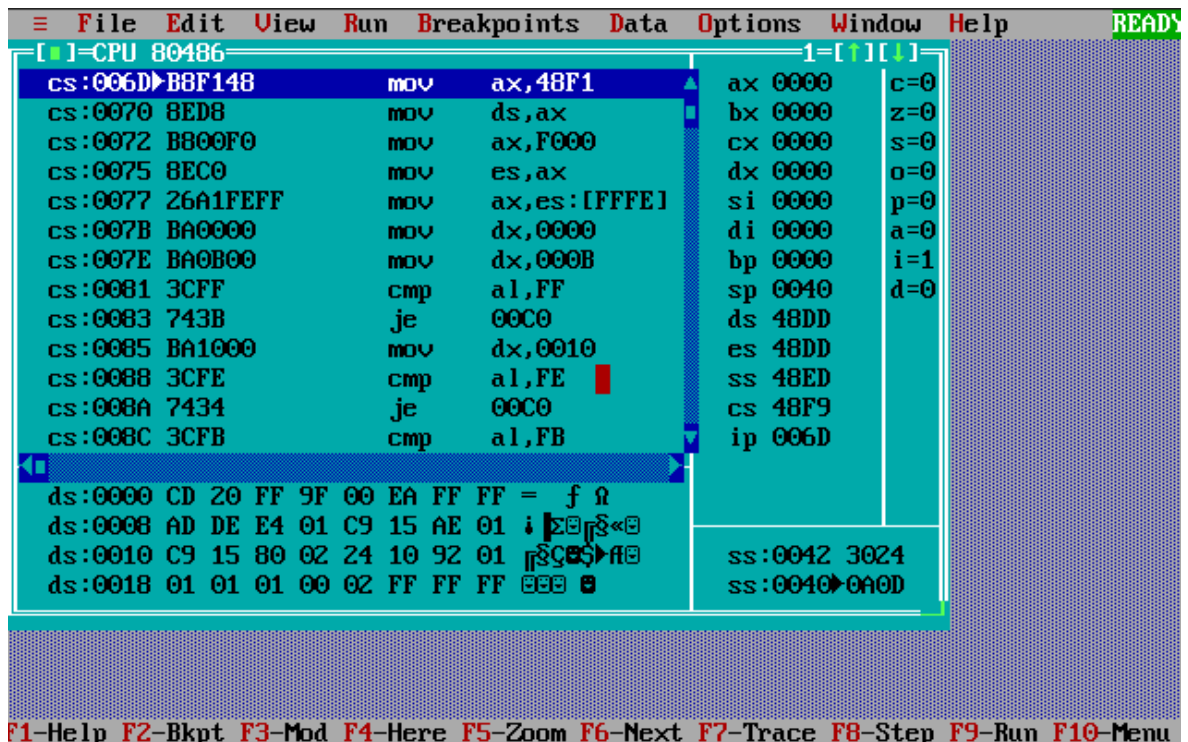


Рисунок 6 — загруженный «хороший» модуль EXE

### 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 адреса 0 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 за вычетом PSP и кода. В регистре SP хранится последний адресуемый байт из 64кб, то есть FFFh.

### Загрузка «хорошего» EXE модуля в основную память.

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

Регистры DS и ES указывают на начало PSP (48DDh), а сегмент стека SS и сегмент кода CS — на соответствующие сегменты.

2) На что указывают регистры DS и ES?

Регистры DS и ES указывают на начало PSP (48DDh).

3) Как определяется стек?

С использованием директивы SEGMENT выделяется память под сегмент стека (в данной работе имеет название AStack) и при загрузке программы регистр SP инициализируется соответствующим значением.

4) Как определяется точка входа?

Точку входа можно явно задать с помощью указания директиве END метки, которую необходимо считать точкой входа.

### **Вывод.**

В ходе выполнения работы были исследованы различия в структурах исходных текстов модулей типов .COM и .EXE, структурах файлов загрузочных модулей и способы их загрузки в основную память.

## ПРИЛОЖЕНИЕ А

### КОД ИСХОДНОГО СОМ МОДУЛЯ

```
TESTPC      SEGMENT
              ; ASSUME  CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
              ORG      100H

START:      JMP      BEGIN

; DATA
NEW_LINE      DB 10,13,'$'
STR_PC_TYPE    DB 'IBM PC TYPE: $'
STR_DOS_VERSION DB 'MSDOS VERSION: $'
STR_OEM_NUMBER DB 'OEM SERIAL NUMBER: $'
STR_USER_NUMBER DB 'USER SERIAL NUMBER: $'
DOS_VERSION    DB '00.00',          10,13,'$'
TYPE1          DB 'PC',              10,13,'$'
TYPE2          DB 'PC/XT',           10,13,'$'
TYPE3          DB 'AT',              10,13,'$'
TYPE4          DB 'PS2 модель 30',    10,13,'$'
TYPE5          DB 'PS2 модель 50 или 60', 10,13,'$'
TYPE6          DB 'PS2 80',          10,13,'$'
TYPE7          DB 'PCJR',            10,13,'$'
TYPE8          DB 'PC CONVERTIBLE',  10,13,'$'

PRINT_BYTE     PROC NEAR
; PRINTS AL AS TWO HEX DIGITS
              PUSH BX
              PUSH DX

              CALL BYTE_TO_HEX
              MOV BH, AH

              MOV DL, AL
              MOV AH, 02H
              INT 21H

              MOV DL, BH
              MOV AH, 02H
              INT 21H
```



```

        POP DX
        POP BX
        RET
PRINT_BYTE    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
; AL --> TWO HEX SYMBOLS IN AX
        PUSH    CX
        MOV     AH,AL
        CALL    TETR_TO_HEX
        XCHG    AL,AH
        MOV     CL,4
        SHR     AL,CL
        CALL    TETR_TO_HEX ; AL - HIGH DIGIT
        POP     CX          ; AH - LOW DIGIT
        RET
BYTE_TO_HEX   ENDP
;-----
WRD_TO_HEX    PROC NEAR
; AX --> HEX, DI - ADDRESS OF LAST SYMBOL
        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
; AL --> DEC, SI - ADDRESS OF LOWER DIGIT
        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
;-----
; CODE
BEGIN:
        MOV     AX, 0F000H
        MOV     ES, AX

        MOV     DX, OFFSET STR_PC_TYPE
        MOV     AH, 09H
        INT     21H

        MOV     AX, ES:[0FFFEH]
        MOV     DX, 0

TEST_TYPE1:
        MOV     DX, OFFSET TYPE1
        CMP     AL, 0FFH
        JE      WRITE_TYPE

```

```

TEST_TYPE2:
    MOV DX, OFFSET TYPE2
    CMP AL, 0FEH
    JE WRITE_TYPE
    CMP AL, 0FBH
    JE WRITE_TYPE
TEST_TYPE3:
    MOV DX, OFFSET TYPE3
    CMP AL, 0FCH
    JE WRITE_TYPE
TEST_TYPE4:
    MOV DX, OFFSET TYPE4
    CMP AL, 0FAH
    JE WRITE_TYPE
TEST_TYPE5:
    MOV DX, OFFSET TYPE5
    CMP AL, 0FCH
    JE WRITE_TYPE
TEST_TYPE6:
    MOV DX, OFFSET TYPE6
    CMP AL, 0F8H
    JE WRITE_TYPE
TEST_TYPE7:
    MOV DX, OFFSET TYPE7
    CMP AL, 0FDH
    JE WRITE_TYPE
TEST_TYPE8:
    MOV DX, OFFSET TYPE8
    CMP AL, 0F9H
    JE WRITE_TYPE
DONESNT_MATCH:
    CALL PRINT_BYTE
    MOV DX, OFFSET NEW_LINE
WRITE_TYPE:
    MOV AH, 09H
    INT 21H
TYPE_END:

PRINT_VERSION:
    MOV DX, OFFSET STR_DOS_VERSION
    MOV AH, 09H

```

```

INT 21H

    MOV AH, 30H
    INT 21H

    MOV SI, OFFSET DOS_VERSION
    ADD SI, 1
    CALL BYTE_TO_DEC

    ADD SI, 3
    MOV AL, AH
    CALL BYTE_TO_DEC

    MOV DX, OFFSET DOS_VERSION
    MOV AH, 09H
    INT 21H

PRINT_OEM:
    MOV DX, OFFSET STR_OEM_NUMBER
    MOV AH, 09H
    INT 21H

    MOV AH, 30H
    INT 21H

    MOV AL, BH
    CALL PRINT_BYTE

PRINT_SERIAL:
    MOV DX, OFFSET NEW_LINE
    MOV AH, 09H
    INT 21H

    MOV DX, OFFSET STR_USER_NUMBER
    MOV AH, 09H
    INT 21H

    MOV AH, 30H
    INT 21H

```

```

        MOV AL, BL
        CALL PRINT_BYTE
        MOV AL, CH
        CALL PRINT_BYTE
        MOV AL, CL
        CALL PRINT_BYTE

; RETURN TO DOS
        XOR     AL,AL
        MOV     AH,4Ch
        INT     21H
TESTPC  ENDS
        END     START      ; MODULE END START - ENTRY POINT

```

## ПРИЛОЖЕНИЕ Б

### КОД ИСХОДНОГО EXE МОДУЛЯ

```
ASTACK SEGMENT STACK
    DW 128 DUP(?)
ASTACK ENDS
```

DATA SEGMENT

```
NEW_LINE          DB 0DH,0AH,'$'
STR_PC_TYPE       DB 'IBM PC TYPE: $'
STR_DOS_VERSION   DB 'MSDOS VERSION: $'
STR_OEM_NUMBER    DB 'OEM SERIAL NUMBER: $'
STR_USER_NUMBER   DB 'USER SERIAL NUMBER: $'
DOS_VERSION       DB '00.00',          0DH,0AH,'$'
TYPE1             DB 'PC',             0DH,0AH,'$'
TYPE2             DB 'PC/XT',          0DH,0AH,'$'
TYPE3             DB 'AT',             0DH,0AH,'$'
TYPE4             DB 'PS2 модель 30',  0DH,0AH,'$'
TYPE5             DB 'PS2 модель 50 или 60', 0DH,0AH,'$'
TYPE6             DB 'PS2 80',         0DH,0AH,'$'
TYPE7             DB 'PCJR',           0DH,0AH,'$'
TYPE8             DB 'PC CONVERTIBLE', 0DH,0AH,'$'
```

DATA ENDS

CODE SEGMENT

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

```
PRINT_BYTE        PROC NEAR
; PRINTS AL AS TWO HEX DIGITS
    PUSH BX
    PUSH DX

    CALL BYTE_TO_HEX
    MOV BH, AH

    MOV DL, AL
    MOV AH, 02H
```

```

        INT 21H

        MOV DL, BH
        MOV AH, 02H
        INT 21H

        POP DX
        POP BX
        RET

PRINT_BYTE    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
; AL --> TWO HEX SYMBOLS IN AX
        PUSH    CX
        MOV     AH,AL
        CALL    TETR_TO_HEX
        XCHG    AL,AH
        MOV     CL,4
        SHR     AL,CL
        CALL    TETR_TO_HEX ; AL - HIGH DIGIT
        POP     CX          ; AH - LOW DIGIT
        RET

BYTE_TO_HEX    ENDP
;-----
WRD_TO_HEX    PROC NEAR
; AX --> HEX, DI - ADDRESS OF LAST SYMBOL
        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
; AL --> DEC, SI - ADDRESS OF LOWER DIGIT
        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
;-----
; CODE
BEGIN     PROC FAR
        MOV     AX, DATA
        MOV     DS, AX

        MOV     DX, OFFSET STR_PC_TYPE
        MOV     AH, 09H
        INT     21H

```



```

        MOV AX, 0F000H
        MOV ES, AX
        MOV AX, ES:[0FFFEH]
        MOV DX, 0
TEST_TYPE1:
        MOV DX, OFFSET TYPE1
        CMP AL, 0FFH
        JE WRITE_TYPE
TEST_TYPE2:
        MOV DX, OFFSET TYPE2
        CMP AL, 0FEH
        JE WRITE_TYPE
        CMP AL, 0FBH
        JE WRITE_TYPE
TEST_TYPE3:
        MOV DX, OFFSET TYPE3
        CMP AL, 0FCH
        JE WRITE_TYPE
TEST_TYPE4:
        MOV DX, OFFSET TYPE4
        CMP AL, 0FAH
        JE WRITE_TYPE
TEST_TYPE5:
        MOV DX, OFFSET TYPE5
        CMP AL, 0FCH
        JE WRITE_TYPE
TEST_TYPE6:
        MOV DX, OFFSET TYPE6
        CMP AL, 0F8H
        JE WRITE_TYPE
TEST_TYPE7:
        MOV DX, OFFSET TYPE7
        CMP AL, 0FDH
        JE WRITE_TYPE
TEST_TYPE8:
        MOV DX, OFFSET TYPE8
        CMP AL, 0F9H
        JE WRITE_TYPE
DOESNT_MATCH:
        CALL PRINT_BYTE
        MOV DX, OFFSET NEW_LINE

```

WRITE\_TYPE:

MOV AH, 09H

INT 21H

TYPE\_END:

PRINT\_VERSION:

MOV DX, OFFSET STR\_DOS\_VERSION

MOV AH, 09H

INT 21H

MOV AH, 30H

INT 21H

MOV SI, OFFSET DOS\_VERSION

ADD SI, 1

CALL BYTE\_TO\_DEC

ADD SI, 3

MOV AL, AH

CALL BYTE\_TO\_DEC

MOV DX, OFFSET DOS\_VERSION

MOV AH, 09H

INT 21H

PRINT\_OEM:

MOV DX, OFFSET STR\_OEM\_NUMBER

MOV AH, 09H

INT 21H

MOV AH, 30H

INT 21H

MOV AL, BH

CALL PRINT\_BYTE

PRINT\_SERIAL:

MOV DX, OFFSET NEW\_LINE

MOV AH, 09H

INT 21H

```

        MOV DX, OFFSET STR_USER_NUMBER
MOV AH, 09H
INT 21H

        MOV AH, 30H
INT 21H

        MOV AL, BL
        CALL PRINT_BYTE
        MOV AL, CH
        CALL PRINT_BYTE
        MOV AL, CL
        CALL PRINT_BYTE

; RETURN TO DOS
        XOR     AL,AL
        MOV     AH,4CH
        INT     21H
BEGIN     ENDP
CODE     ENDS
END       BEGIN      ; MODULE END START - ENTRY POINT

```

## ПРИЛОЖЕНИЕ В

### МОДУЛЬ СОМ В ШЕСТНАДЦАТЕРИЧНОМ ВИДЕ

```

00000000: e9e1 000a 0d24 3030 2e30 300a 0d24 5043 .....$00.00..$PC
00000010: 0a0d 2450 432f 5854 0a0d 2441 540a 0d24 ..$PC/XT..$AT..$
00000020: 5053 3220 d0bc d0be d0b4 d0b5 d0bb d18c PS2 .....
00000030: 2033 300a 0d24 5053 3220 d0bc d0be d0b4 30..$PS2 .....
00000040: d0b5 d0bb d18c 2035 3020 d0b8 d0bb d0b8 ..... 50 .....
00000050: 2036 300a 0d24 5053 3220 3830 0a0d 2450 60..$PS2 80..$P
00000060: 436a 720a 0d24 5043 2043 6f6e 7665 7274 Cjr..$PC Convert
00000070: 6962 6c65 0a0d 2453 52e8 1c00 8afc 8ad0 ible..$SR.....
00000080: b402 cd21 8ad7 b402 cd21 5a5b c324 0f3c ...!.....!Z[.$.<
00000090: 0976 0204 0704 30c3 518a e0e8 efff 86c4 .v....0.Q.....
000000a0: b104 d2e8 e8e6 ff59 c353 8afc e8e9 ff88 .....Y.S.....
000000b0: 254f 8805 4f8a c7e8 deff 8825 4f88 055b %0..0.....%0..[
000000c0: c351 5232 e433 d2b9 0a00 f7f1 80ca 3088 .QR2.3.....0.
000000d0: 144e 33d2 3d0a 0073 f13c 0074 040c 3088 .N3.=..s.<.t..0.
000000e0: 045a 59c3 b800 f08e c026 a1fe ffba 0000 .ZY.....&.....
000000f0: ba0e 013c ff74 3bba 1301 3cfe 7434 3cfb ...<.t;...<.t4<.
00000100: 7430 ba1b 013c fc74 29ba 2001 3cfa 7422 t0...<.t). .<.t"
00000110: ba36 013c fc74 1bba 5601 3cf8 7414 ba5f .6.<.t..V.<.t.._
00000120: 013c fd74 0dba 6601 3cf9 7406 e848 ffba .<.t..f.<.t..H..
00000130: 0301 b409 cd21 b430 cd21 be06 0183 c601 .....!.0.!.....
00000140: e87e ff83 c603 8ac4 e876 ffba 0601 b409 .~.....v.....
00000150: cd21 b430 cd21 8ac7 e81c ffb4 30cd 21ba .!.0.!.....0.!.
00000160: 0301 b409 cd21 8ac3 e80c ff8a c5e8 07ff .....!.....
00000170: 8ac1 e802 ff32 c0b4 4ccd 210a .....2..L.!.

```

## ПРИЛОЖЕНИЕ Г

### «ПЛОХОЙ» МОДУЛЬ EXE В ШЕСТНАДЦАТЕРИЧНОМ ВИДЕ

00000000:	4d5a	7b00	0300	0000	2000	0000	ffff	0000	MZ{..... .....
00000010:	0000	bfc4	0001	0000	1e00	0000	0100	0000	.....
00000020:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000030:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000040:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000050:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000060:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000070:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000080:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000090:	0000	0000	0000	0000	0000	0000	0000	0000	.....
000000a0:	0000	0000	0000	0000	0000	0000	0000	0000	.....
000000b0:	0000	0000	0000	0000	0000	0000	0000	0000	.....
000000c0:	0000	0000	0000	0000	0000	0000	0000	0000	.....
000000d0:	0000	0000	0000	0000	0000	0000	0000	0000	.....
000000e0:	0000	0000	0000	0000	0000	0000	0000	0000	.....
000000f0:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000100:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000110:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000120:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000130:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000140:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000150:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000160:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000170:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000180:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000190:	0000	0000	0000	0000	0000	0000	0000	0000	.....
000001a0:	0000	0000	0000	0000	0000	0000	0000	0000	.....
000001b0:	0000	0000	0000	0000	0000	0000	0000	0000	.....
000001c0:	0000	0000	0000	0000	0000	0000	0000	0000	.....
000001d0:	0000	0000	0000	0000	0000	0000	0000	0000	.....
000001e0:	0000	0000	0000	0000	0000	0000	0000	0000	.....
000001f0:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000200:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000210:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000220:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000230:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000240:	0000	0000	0000	0000	0000	0000	0000	0000	.....
00000250:	0000	0000	0000	0000	0000	0000	0000	0000	.....

```

00000260: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000270: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000280: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000290: 0000 0000 0000 0000 0000 0000 0000 0000 .....
000002a0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
000002b0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
000002c0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
000002d0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
000002e0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
000002f0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000300: e9e1 000a 0d24 3030 2e30 300a 0d24 5043 .....$00.00..$PC
00000310: 0a0d 2450 432f 5854 0a0d 2441 540a 0d24 ..$PC/XT..$AT..$
00000320: 5053 3220 d0bc d0be d0b4 d0b5 d0bb d18c PS2 .....
00000330: 2033 300a 0d24 5053 3220 d0bc d0be d0b4 30..$PS2 .....
00000340: d0b5 d0bb d18c 2035 3020 d0b8 d0bb d0b8 ..... 50 .....
00000350: 2036 300a 0d24 5053 3220 3830 0a0d 2450 60..$PS2 80..$P
00000360: 436a 720a 0d24 5043 2043 6f6e 7665 7274 Cjr..$PC Convert
00000370: 6962 6c65 0a0d 2453 52e8 1c00 8afc 8ad0 ible..$SR.....
00000380: b402 cd21 8ad7 b402 cd21 5a5b c324 0f3c ...!.....!Z[.$.<
00000390: 0976 0204 0704 30c3 518a e0e8 efff 86c4 .v....0.Q.....
000003a0: b104 d2e8 e8e6 ff59 c353 8afc e8e9 ff88 .....Y.S.....
000003b0: 254f 8805 4f8a c7e8 deff 8825 4f88 055b %0..0.....%0..[
000003c0: c351 5232 e433 d2b9 0a00 f7f1 80ca 3088 .QR2.3.....0.
000003d0: 144e 33d2 3d0a 0073 f13c 0074 040c 3088 .N3.=..s.<.t..0.
000003e0: 045a 59c3 b800 f08e c026 a1fe ffba 0000 .ZY.....&.....
000003f0: ba0e 013c ff74 3bba 1301 3cfe 7434 3cfb ...<.t;...<.t4<.
00000400: 7430 ba1b 013c fc74 29ba 2001 3cfa 7422 t0...<.t). .<.t"
00000410: ba36 013c fc74 1bba 5601 3cf8 7414 ba5f .6.<.t..V.<.t.._
00000420: 013c fd74 0dba 6601 3cf9 7406 e848 ffba .<.t..f.<.t..H..
00000430: 0301 b409 cd21 b430 cd21 be06 0183 c601 .....!.0.!.....
00000440: e87e ff83 c603 8ac4 e876 ffba 0601 b409 .~.....v.....
00000450: cd21 b430 cd21 8ac7 e81c ffb4 30cd 21ba .!.0.!.....0.!.
00000460: 0301 b409 cd21 8ac3 e80c ff8a c5e8 07ff .....!.....
00000470: 8ac1 e802 ff32 c0b4 4ccd 210a .....2..L.!.

```

## ПРИЛОЖЕНИЕ Д

### «ХОРОШИЙ» МОДУЛЬ EXE В ШЕСТНАДЦАТЕРИЧНОМ ВИДЕ

```

00000000: 4d5a e500 0300 0100 2000 0000 ffff 0000  MZ.....
00000010: 0001 fe90 6d00 1c00 1e00 0000 0100 6e00  ....m.....n.
00000020: 1c00 0000 0000 0000 0000 0000 0000 0000  .....
00000030: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000040: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000050: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000060: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000070: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000080: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000090: 0000 0000 0000 0000 0000 0000 0000 0000  .....
000000a0: 0000 0000 0000 0000 0000 0000 0000 0000  .....
000000b0: 0000 0000 0000 0000 0000 0000 0000 0000  .....
000000c0: 0000 0000 0000 0000 0000 0000 0000 0000  .....
000000d0: 0000 0000 0000 0000 0000 0000 0000 0000  .....
000000e0: 0000 0000 0000 0000 0000 0000 0000 0000  .....
000000f0: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000100: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000110: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000120: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000130: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000140: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000150: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000160: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000170: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000180: 0000 0000 0000 0000 0000 0000 0000 0000  .....
00000190: 0000 0000 0000 0000 0000 0000 0000 0000  .....
000001a0: 0000 0000 0000 0000 0000 0000 0000 0000  .....
000001b0: 0000 0000 0000 0000 0000 0000 0000 0000  .....
000001c0: 0000 0000 0000 0000 0000 0000 0000 0000  .....
000001d0: 0000 0000 0000 0000 0000 0000 0000 0000  .....
000001e0: 0000 0000 0000 0000 0000 0000 0000 0000  .....
000001f0: 0000 0000 0000 0000 0000 0000 0000 0000  .....
0000200: 0000 0000 0000 0000 0000 0000 0000 0000  .....
0000210: 0000 0000 0000 0000 0000 0000 0000 0000  .....
0000220: 0000 0000 0000 0000 0000 0000 0000 0000  .....
0000230: 0000 0000 0000 0000 0000 0000 0000 0000  .....
0000240: 0000 0000 0000 0000 0000 0000 0000 0000  .....
0000250: 0000 0000 0000 0000 0000 0000 0000 0000  .....

```

```

00000260: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000270: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000280: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000290: 0000 0000 0000 0000 0000 0000 0000 0000 .....
000002a0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
000002b0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
000002c0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
000002d0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
000002e0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
000002f0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000300: 0d0a 2449 424d 2050 4320 7479 7065 3a20 ..$IBM PC type:
00000310: 244d 5344 4f53 2076 6572 7369 6f6e 3a20 $MSDOS version:
00000320: 244f 454d 2073 6572 6961 6c20 6e75 6d62 $OEM serial numb
00000330: 6572 3a20 2455 7365 7220 7365 7269 616c er: $User serial
00000340: 206e 756d 6265 723a 2024 3030 2e30 300d number: $00.00.
00000350: 0a24 5043 0d0a 2450 432f 5854 0d0a 2441 .$PC..$PC/XT..$A
00000360: 540d 0a24 5053 3220 d0bc d0be d0b4 d0b5 T..$PS2 .....
00000370: d0bb d18c 2033 300d 0a24 5053 3220 d0bc .... 30..$PS2 ..
00000380: d0be d0b4 d0b5 d0bb d18c 2035 3020 d0b8 ..... 50 ..
00000390: d0bb d0b8 2036 300d 0a24 5053 3220 3830 .... 60..$PS2 80
000003a0: 0d0a 2450 436a 720d 0a24 5043 2043 6f6e ..$PCjr..$PC Con
000003b0: 7665 7274 6962 6c65 0d0a 2400 0000 0000 vertible..$. ....
000003c0: 5352 e81c 008a fc8a d0b4 02cd 218a d7b4 SR.....!...
000003d0: 02cd 215a 5bc3 240f 3c09 7602 0407 0430 ..!Z[.$.<.v....0
000003e0: c351 8ae0 e8ef ff86 c4b1 04d2 e8e8 e6ff .Q.....
000003f0: 59c3 538a fce8 e9ff 8825 4f88 054f 8ac7 Y.S.....%0..0..
00000400: e8de ff88 254f 8805 5bc3 5152 32e4 33d2 ....%0..[.QR2.3.
00000410: b90a 00f7 f180 ca30 8814 4e33 d23d 0a00 .....0..N3.=..
00000420: 73f1 3c00 7404 0c30 8804 5a59 c3b8 1000 s.<.t..0..ZY....
00000430: 8ed8 ba03 00b4 09cd 21b8 00f0 8ec0 26a1 .....!.....&.
00000440: feff ba00 00ba 5200 3cff 743b ba57 003c .....R.<.t;.W.<
00000450: fe74 343c fb74 30ba 5f00 3cfc 7429 ba64 .t4<.t0._.<.t).d
00000460: 003c fa74 22ba 7a00 3cfc 741b ba9a 003c .<.t".z.<.t....<
00000470: f874 14ba a300 3cfd 740d baaa 003c f974 .t....<.t....<.t
00000480: 06e8 3cff ba00 00b4 09cd 21ba 1100 b409 ..<.....!.....
00000490: cd21 b430 cd21 be4a 0083 c601 e86b ff83 .!.0!.J.....k..
000004a0: c603 8ac4 e863 ffba 4a00 b409 cd21 ba21 .....c..J.....!!
000004b0: 00b4 09cd 21b4 30cd 218a c7e8 02ff ba00 ....!.0!. ....
000004c0: 00b4 09cd 21ba 3500 b409 cd21 b430 cd21 ....!.5....!.0.!
000004d0: 8ac3 e8eb fe8a c5e8 e6fe 8ac1 e8e1 fe32 .....2
000004e0: c0b4 4ccd 21 ..L.!

```