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. 9381 Шахин Н.С

Преподаватель Ефремов М. А.

Санкт-Петербург

2021

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.

1. На основе шаблона, приведенного в методических указаниях, был написан текст исходного .COM модуля com.asm, который определяет тип PC и версию системы. Далее при помощи транслятора MASM.exe и компоновщика LINK.exe был скомпилирован плохой .EXE модуль. При помощи EXE2BIN.EXE был построен хороший .COM модуль.

2. Был написан текст исходного .Exe модуля exe.asm. Далее при помощи транслятора MASM.exe и компоновщика LINK.exe был скомпилирован хороший .EXE модуль.

3. Было выполнено сравнение исходных текстов com.asm и exe.asm.

4. При помощи программы FAR файлы загрузочных модулей com.com, com.exe и exe.exe были открыты в шестнадцатеричном виде.

5. Был исследован загрузочный модуль .com при помощи отладчика.

6. был исследован загрузочный модуль .exe при помощи отладчика.

Функции программ.

Названия функций	Описание
TETR_TO_HEX	Перевод десятичной цифры в код символа.
BYTE_TO_HEX	Перевод байта в 16-ной с/с в символьный код
WRD_TO_HEX	Перевод слова в 16-ной с/с в символьный код
BYTE_TO_DEC	Перевод байта в 16-ной с/с в символьный код в 10-ной с/с
PRINT_STRING	Вывод строки.

Ответы на контрольные вопросы.

Отличия исходных текстов COM и EXE программ

1. Сколько сегментов должна содержать COM-программа?

Один сегмент, в котором находятся код и данные.

2. EXE программа?

Программы в формате EXE могут иметь любое количество сегментов. EXE-программа предполагает отдельные сегменты для кода, данных и стека.

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

Директива ORG 100h, которая задает смещение для всех адресов программы на 256 байт для префикса программного сегмента (PSP).

Директива ASSUME. Указывает ассемблеру, с каким сегментом или группой сегментов связан тот или иной сегментный регистр. Она не изменяет значений сегментных регистров, а только позволяет ассемблеру проверять допустимость ссылок и самостоятельно вставлять при необходимости префиксы переопределения сегментов. Без этой директивы программа не скомпилируется, так как ассемблер не будет понимать, относительно чего вычислять смещения меток.

Директива END. Этой директивой завершается любая программа на ассемблере.

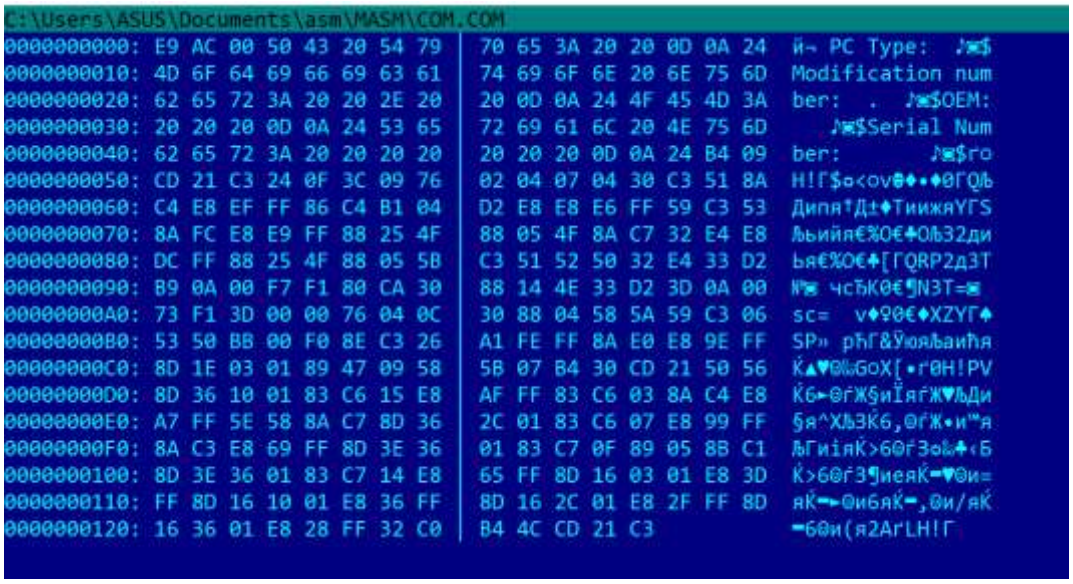
4. Все ли форматы команд можно использовать в COM программе?

Нет, не все. Нельзя использовать команды, связанные с адресом сегмента, потому что адрес сегмента до загрузки неизвестен, так как в COM-программах в DOS не содержится таблицы настройки, которая содержит описание адресов, зависящих от размещения загрузочного модуля в ОП, потому что подобные адреса в нем запрещены.

Отличия форматов файлов COM и EXE модулей

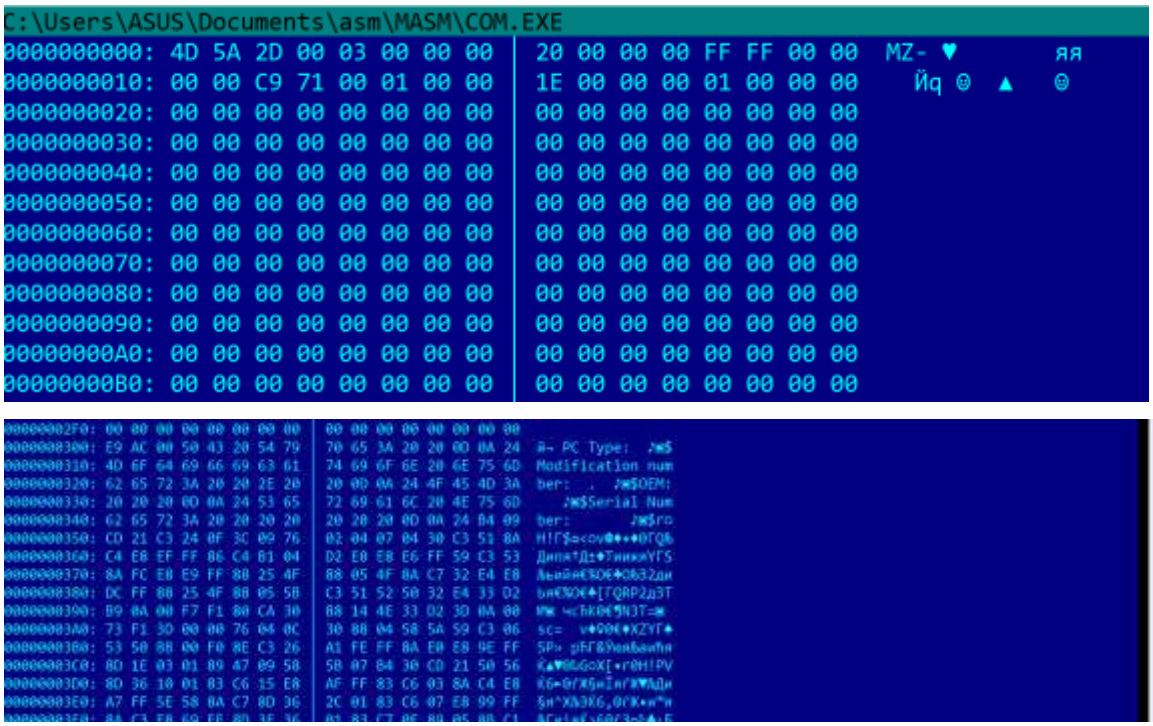
1. Какова структура файла COM? С какого адреса располагается код?

В COM-файле код, данные и стек располагаются в одном сегменте. Код (как и данные) начинается с адреса 0h.



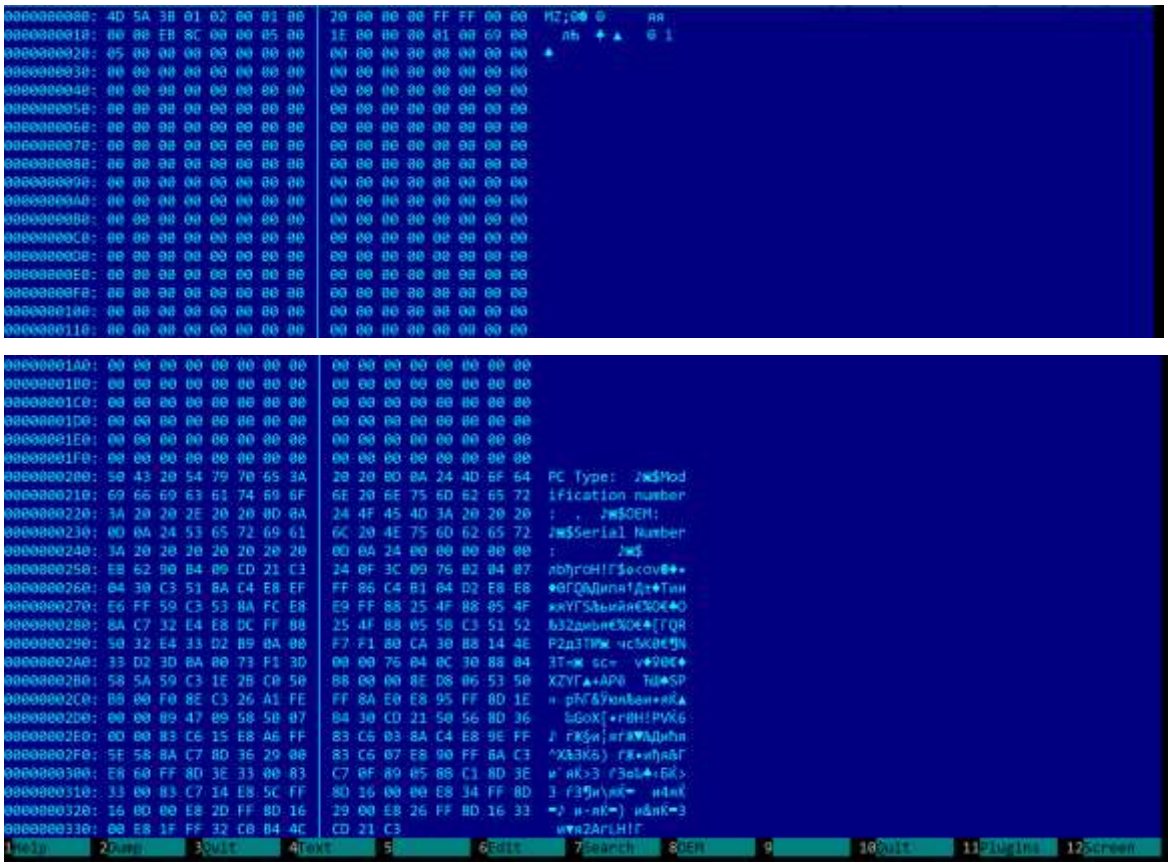
2. Какова структура файла «плохого» EXE? С какого адреса располагается код? Что располагается с 0 адреса?

В плохом EXE-файле код, данные и стек располагаются в одном сегменте. Код (как и данные) начинается с адреса 300h. С адреса 0h располагается управляющая информация для загрузчика, которая содержит заголовок и таблицу настройки адресов.



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

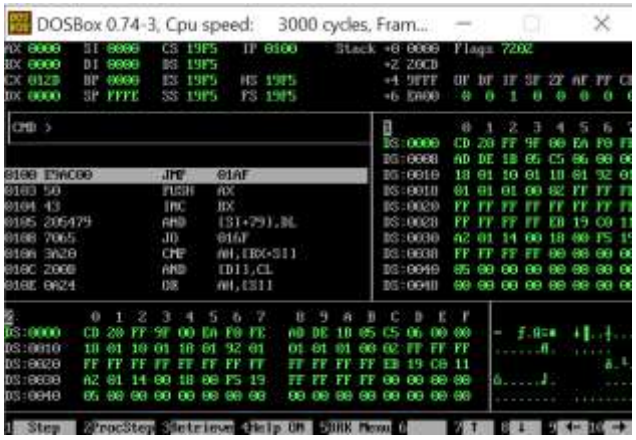
В отличие от плохого, хороший EXE-файл не содержит директивы ORG 100h (которая выделяет память под PSP), поэтому код начинается с адреса 200h. В хорошем EXE-файле код, данные и стек находятся в различных сегментах, а в плохом – в одном и том же сегменте.



Загрузка COM модуля в основную память

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

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



2. Что располагается с 0 адреса?

С адреса 0 располагается префикс программного сегмента (PSP).

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

Регистры DS, ES, CS, SS указывают на начало блока PSP.

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

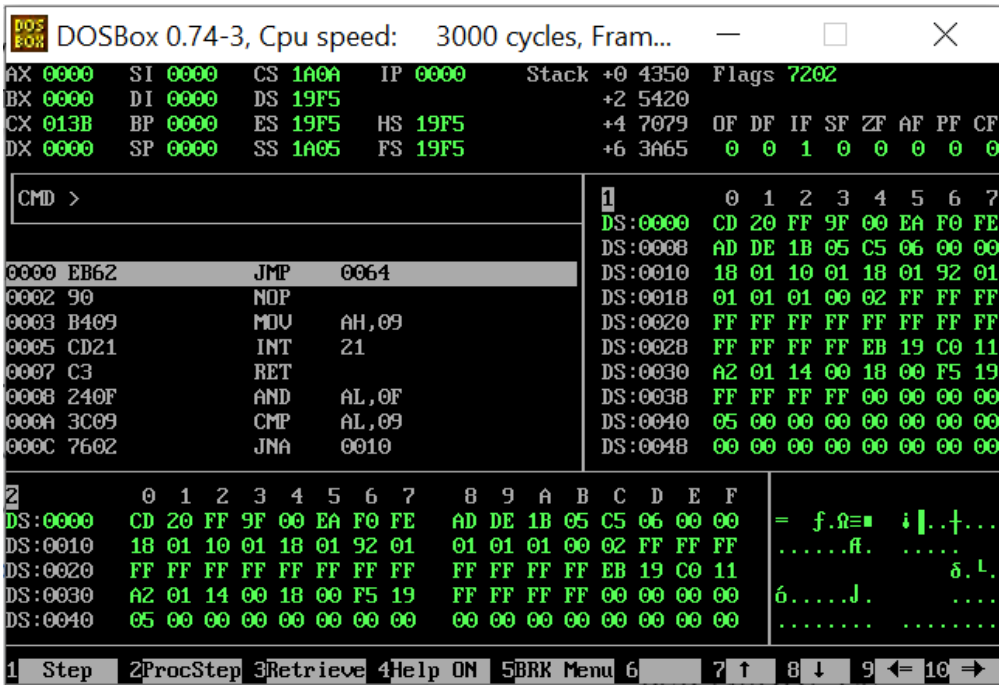
Стек создается автоматически. Регистр SS указывает на начало блока PSP (0h), а SP указывает на конец модуля (FFFFh). То есть стек расположен между адресами SS:0000h и SS:FFFFh и заполняется с конца модуля в строну уменьшения адресов.

Загрузка «хорошего» EXE модуля в память

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

Сначала создается PSP. Затем определяется длина тела загрузочного модуля, определяется начальный сегмент. Загрузочный модуль считывается в начальный сегмент, таблица настройки считывается в рабочую память, к полю каждого сегмента прибавляется сегментный адрес начального сегмента, определяются значения сегментных регистров. DS и ES указывают на начало

PSP (19f5), CS – на начало сегмента команд (1A0A), а SS – на начало сегмента стека (1A05).



2. На что указывают регистры DS и ES?

Изначально регистры DS и ES указывают на начало сегмента PSP.

3. Как определяется стек?

Регистр SS указывает на начало сегмента стека, а SP – на конец сегмента стека.

4. Как определяется точка входа?

С помощью директивы END, операндом которой является адрес, с которого начинается выполнение программы.

Вывод.

Были изучены различия в структурах исходных текстов модулей типов .COM и .EXE, структур файлов загрузочных модулей и способов их загрузки в основную память.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл com.asm:

```
TESTPC SEGMENT
            ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
            ORG 100H
START:      JMP   BEGIN

;ДАННЫЕ
PC_Type                 db    'PC Type:  ', 0dh, 0ah,'$'
Mod_numb          db    'Modification number:  .  ', 0dh, 0ah,'$'
OEM                     db    'OEM:   ', 0dh, 0ah, '$'
S_numb          db      'Serial Number:       ', 0dh, 0ah, '$'

;ПРОЦЕДУРЫ
;-------------------------------------------------------------------------
----
;печать строки
PRINT_STRING PROC near
            mov   ah, 09h
            int         21h
            ret
PRINT_STRING ENDP

;-------------------------------------------------------------------------
----
;перевод десятичной цифры в код символа
TETR_TO_HEX       PROC  near
            and         al, 0fh ;логическое умножение всех пар битов
            cmp         al, 09
            jbe         NEXT ;Переход если ниже или равно
            add         al, 07
NEXT: add         al, 30h
            ret
TETR_TO_HEX       ENDP

;-------------------------------------------------------------------------
----
;перевод байта 16 с.с в символьный код
;байт в AL переводится в два символа шестнадцетиричного числа в AX
BYTE_TO_HEX       PROC near
            push  cx
            mov         al, ah
            call  TETR_TO_HEX
            xchg  al, ah
            mov         cl, 4
            shr         al, cl ;логический сдвиг вправо
            call  TETR_TO_HEX ;в AL старшая цифра
            pop         cx                ;в AH младшая
            ret
BYTE_TO_HEX       ENDP

;-------------------------------------------------------------------------
----
;перевод в 16 с/с 16-ти разрядного числа
;в AX - число, DI - адрес последнего символа
WRD_TO_HEX        PROC  near
            push  bx
            mov         bh, ah
            call  BYTE_TO_HEX
```

```
            mov         [di], ah
            dec         di
            mov         [di], al
            dec         di
            mov         al, bh
            xor         ah, ah
            call  BYTE_TO_HEX
            mov         [di], ah
            dec         di
            mov         [di], al
            pop         bx
            ret
WRD_TO_HEX        ENDP

;-------------------------------------------------------------------------
----
;перевод байта 16 с.с в символьный код 10 с.с
;si - адрес поля младшей цифры
BYTE_TO_DEC       PROC  near
            push  cx
            push  dx
            push  ax
            xor         ah, ah
            xor         dx, dx
            mov         cx, 10
loop_bd:div       cx
            or          dl, 30h
            mov   [si], dl
            dec   si
            xor         dx, dx
            cmp         ax, 10
            jae         loop_bd
            cmp         ax, 00h
            jbe         end_l
            or          al, 30h
            mov         [si], al
end_l:      pop         ax
            pop         dx
            pop         cx
            ret
BYTE_TO_DEC       ENDP

;-------------------------------------------------------------------------
----
BEGIN:
            ;PC_Type
            push  es
            push  bx
            push  ax
            mov   bx, 0F000h
            mov   es, bx
            mov   ax, es:[0FFFEh]
            mov   ah, al
            call  BYTE_TO_HEX
            lea         bx, PC_Type
            mov   [bx + 9], ax; смещение на количество символов
            pop         ax
            pop   bx
            pop   es

            ;Mod_numb
            mov   ah, 30h
            int         21h
```

```
            push  ax
            push  si
            lea         si, Mod_numb
            add         si, 21
            call  BYTE_TO_DEC
            add         si, 3
            mov   al, ah
            call        BYTE_TO_DEC
            pop   si
            pop   ax

            ;OEM
            mov   al, bh
            lea         si, OEM
            add         si, 7
            call  BYTE_TO_DEC

            ;S_numb
            mov   al, bl
            call  BYTE_TO_HEX
            lea         di, S_numb
            add         di, 15
            mov   [di], ax
            mov   ax, cx
            lea         di, S_numb
            add         di, 20
            call  WRD_TO_HEX

            lea         dx, PC_Type
            call  PRINT_STRING
            lea         dx, Mod_numb
            call  PRINT_STRING
            lea         dx, OEM
            call  PRINT_STRING
            lea         dx, S_numb
            call  PRINT_STRING

            xor         al, al
            mov   ah, 4ch
            int         21h
            ret
TESTPC      ENDS
            END   START
```

Файл exe.asm:

```
AStack    SEGMENT  STACK
AStack    ENDS

DATA SEGMENT
PC_Type           db    'PC Type:  ', 0dh, 0ah,'$'
Mod_numb        db      'Modification number:  .  ', 0dh, 0ah,'$'
OEM                     db    'OEM:   ', 0dh, 0ah, '$'
S_numb              db  'Serial Number:       ', 0dh, 0ah, '$'
DATA ENDS

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

START:
      jmp BEGIN
;ПРОЦЕДУРЫ
```

```
;-------------------------------------------------------------------------
----
;печать строки
PRINT_STRING PROC near
            mov   ah, 09h
            int         21h
            ret
PRINT_STRING ENDP


;-------------------------------------------------------------------------
----
;перевод десятичной цифры в код символа
TETR_TO_HEX       PROC  near
            and         al, 0fh ;логическое умножение всех пар битов
            cmp         al, 09
            jbe         NEXT ;Переход если ниже или равно
            add         al, 07
NEXT: add         al, 30h
            ret
TETR_TO_HEX       ENDP

;-------------------------------------------------------------------------
----
;перевод байта 16 с.с в символьный код
;байт в AL переводится в 2 символа шестнадцетиричного числа в AX
BYTE_TO_HEX       PROC near
            push  cx
            mov         al, ah
            call  TETR_TO_HEX
            xchg  al, ah
            mov         cl, 4
            shr         al, cl ;логический сдвиг вправо
            call  TETR_TO_HEX
            pop         cx
            ret
BYTE_TO_HEX       ENDP


;-------------------------------------------------------------------------
----
;перевод в 16 с/с 16-ти разрядного числа
;в AX - число, DI - адрес последнего символа
WRD_TO_HEX        PROC  near
            push  bx
            mov         bh, ah
            call  BYTE_TO_HEX
            mov         [di], ah
            dec         di
            mov         [di], al
            dec         di
            mov         al, bh
            xor         ah, ah
            call  BYTE_TO_HEX
            mov         [di], ah
            dec         di
            mov         [di], al
            pop         bx
            ret
WRD_TO_HEX        ENDP
```

```
;--------------------------------------------------------------------------
----
;перевод байта 16 с.с в символьный код 10 с.с
;si - адрес поля младшей цифры
BYTE_TO_DEC       PROC  near
            push  cx
            push  dx
            push  ax
            xor         ah, ah
            xor         dx, dx
            mov         cx, 10
loop_bd:div       cx
            or          dl, 30h
            mov   [si], dl
            dec   si
            xor         dx, dx
            cmp         ax, 10
            jae         loop_bd
            cmp         ax, 00h
            jbe         end_l
            or          al, 30h
            mov         [si], al
end_l:      pop         ax
            pop         dx
            pop         cx
            ret
BYTE_TO_DEC       ENDP


;--------------------------------------------------------------------------
----
BEGIN:
            push  ds
    sub   ax, ax
    push  ax
    mov   ax, DATA
    mov   ds, ax

            ;PC_Type
            push  es
            push  bx
            push  ax
            mov   bx, 0F000h
            mov   es, bx
            mov   ax, es:[0FFFEh]
            mov   ah, al
            call  BYTE_TO_HEX
            lea         bx, PC_Type
            mov   [bx + 9], ax ;смещение на количество символов
            pop         ax
            pop   bx
            pop   es

            ;Mod_numb
            mov   ah, 30h
            int         21h
            push  ax
            push  si
            lea         si, Mod_numb
            add         si, 21
            call  BYTE_TO_DEC
            add         si, 3
            mov   al, ah
```

```
            call        BYTE_TO_DEC
            pop   si
            pop   ax

            ;OEM
            mov   al, bh
            lea         si, OEM
            add         si, 7
            call  BYTE_TO_DEC

            ;S_Numb
            mov   al, bl
            call  BYTE_TO_HEX
            lea         di, S_numb
            add         di, 15
            mov   [di], ax
            mov   ax, cx
            lea         di, S_numb
            add         di, 20
            call  WRD_TO_HEX

            lea         dx, PC_Type
            call  PRINT_STRING
            lea         dx, Mod_numb
            call  PRINT_STRING
            lea         dx, OEM
            call  PRINT_STRING
            lea         dx, S_numb
            call  PRINT_STRING

            ;выход в dos
            xor         al, al
            mov   ah, 4ch
            int         21h
            ret
CODE  ENDS
            END   START
```