**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра МО ЭВМ**

# ОТЧЕТ

**по лабораторной работе №1**

**по дисциплине «Операционные системы»**

**Тема: Исследование структур загрузочных модулей**

Студентка гр. 8382 _____ Ефимова М.А.

Преподаватель _____ Ефремов М.А,

Санкт-Петербург

2020

**Цель работы.**

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

**Ход выполнения.**

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

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

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

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


рис. 1 – результат выполнения .COM модуля

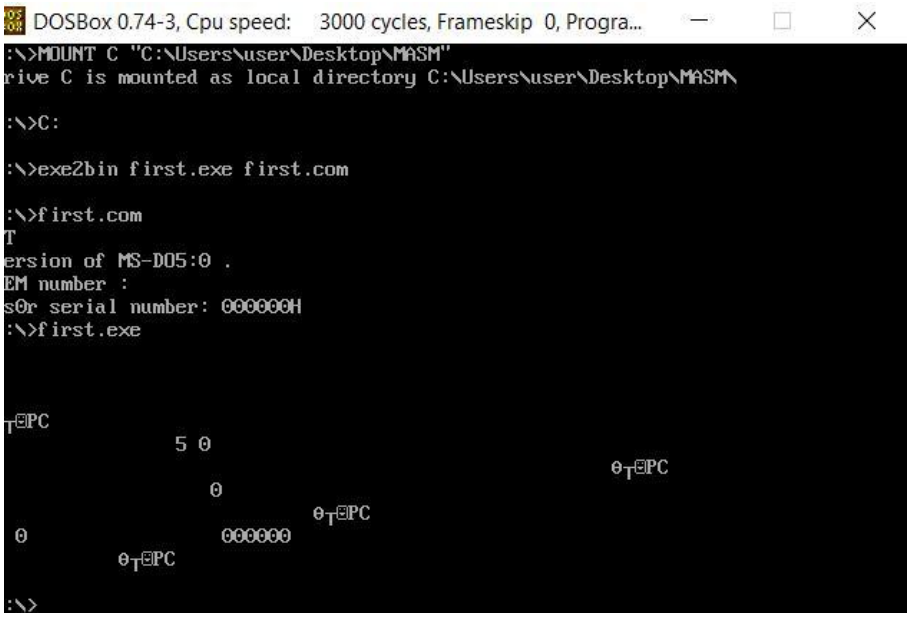



рис. 2 – результат выполнения «плохого» .EXE модуля

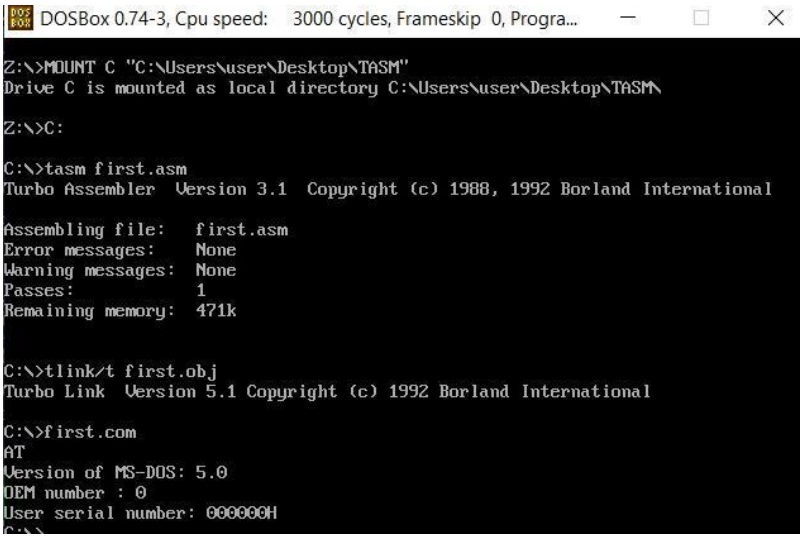



рис. 3 -результат выполнения «хорошего» .EXE модуля

**Отличия исходных текстов COM и EXE программ**

1. Сколько сегментов должна содержать COM-программа?

   У файла типа .COM есть один сегмент команд.

2. Сколько сегментов должна содержать EXE-программа?

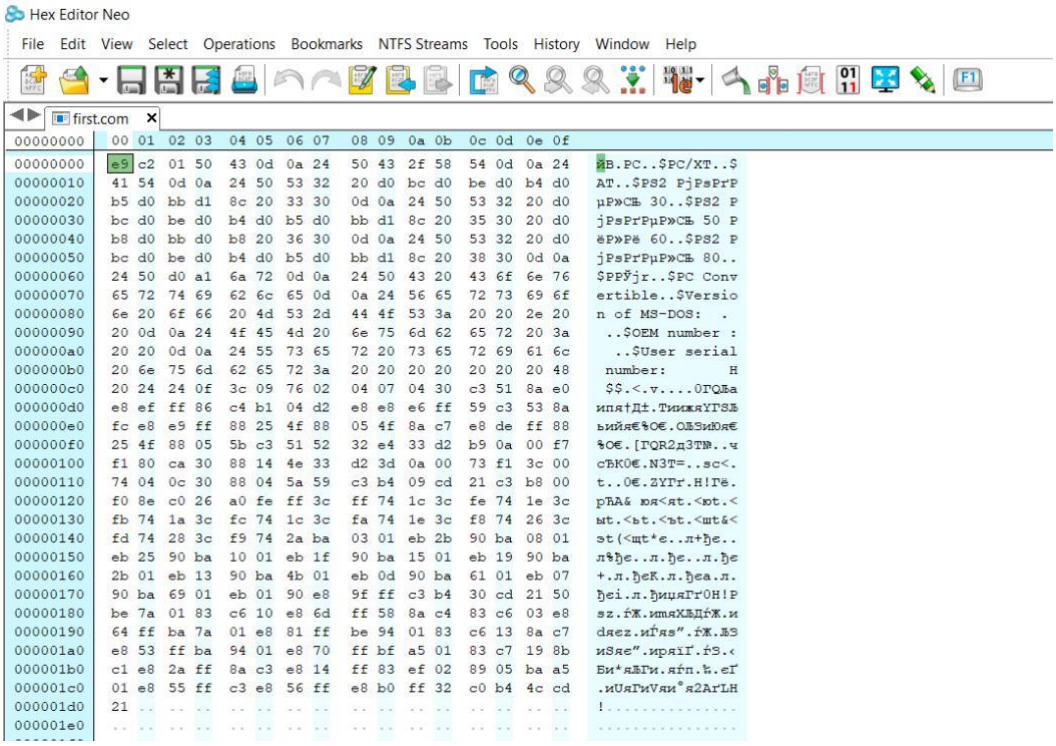
   У файла типа .EXE может содержать не менее одного сегмента.

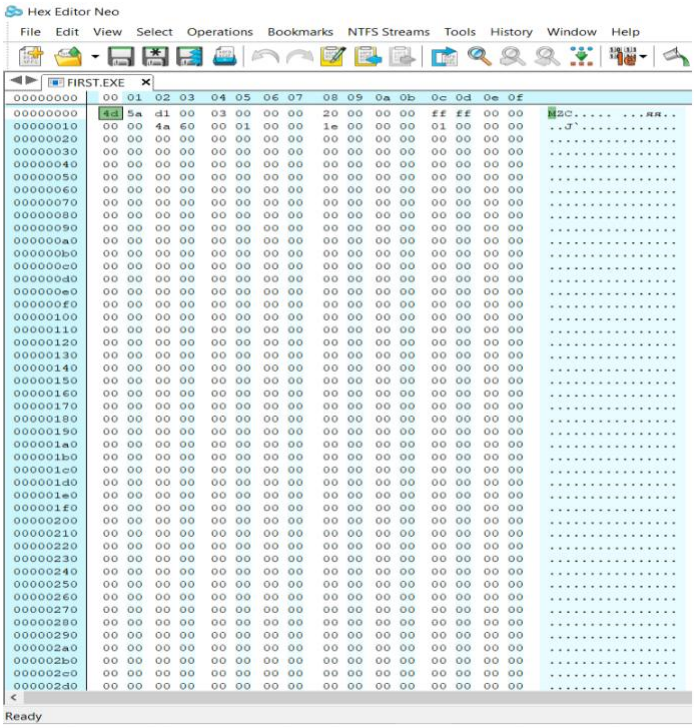
3. Какие директивы должны обязательно быть в тексте COM-программы?

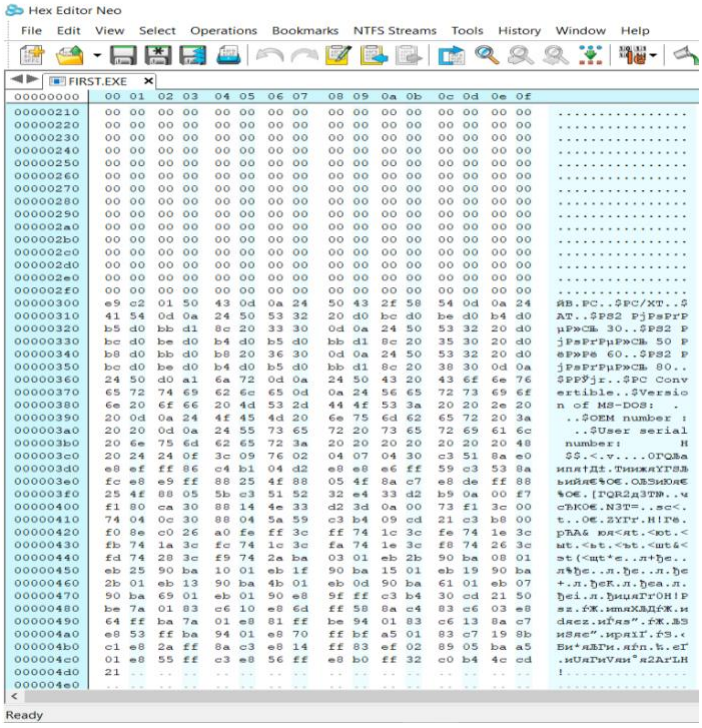
DOS передает управление в сегмент памяти, отведенный для команд, в точку со смещением 100H.Должна присутствовать директива ORG 100h и директива ASSUME для того, чтобы сегмент кода и сегмент данных указывали на общий сегмент.

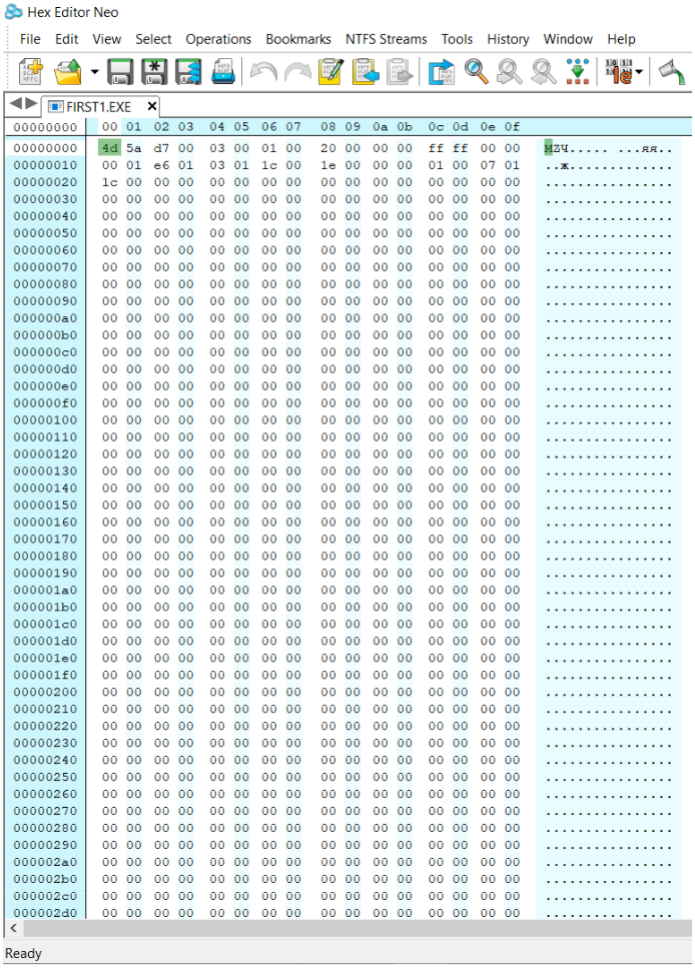
4. Все ли форматы команд можно использовать в COM-программе?
Нет, не все форматы команд можно использовать в COM-программе. В COM – файле есть только один сегмент, а команды, в которых присутствует адрес сегмента, использовать нельзя так как этот адрес становится известным только после загрузки сегмента в память.

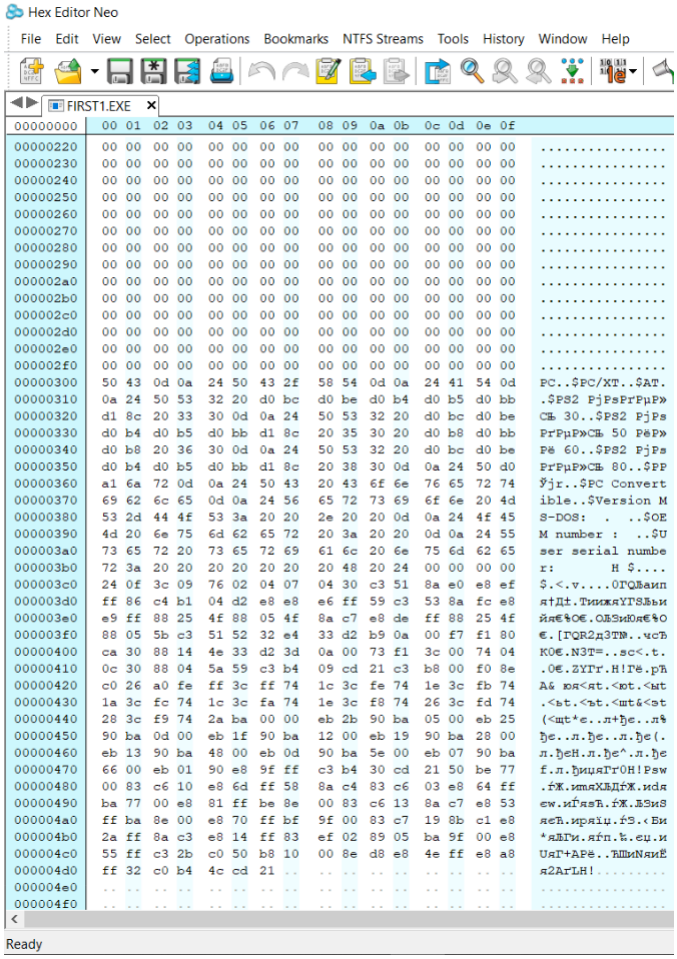
2. Файл загрузочного модуля .COM, «плохого» .EXE и «хорошего» в шестнадцатеричном виде представлены ниже.




Hex Editor Neo
File Edit View Select Operations Bookmarks NTFS Streams Tools History Window Help
FIRST1.EXE
00000000 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
00000000 4d 5a d7 00 03 00 01 00 20 00 00 00 ff ff 00 00 MZ×..... ...яя..
00000010 00 01 e6 01 03 01 1c 00 1e 00 00 00 01 00 07 01 ..ж.............
00000020 1c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000000f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000200 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000210 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000220 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000230 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000240 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000250 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000260 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000270 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000280 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000290 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000002a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000002b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000002c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000002d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
Ready


Hex Editor Neo
File Edit View Select Operations Bookmarks NTFS Streams Tools History Window Help
FIRST1.EXE
00000000 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
00000220 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000230 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000240 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000250 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000260 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000270 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000280 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000290 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000002a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000002b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000002c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000002d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000002e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000002f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000300 50 43 0d 0a 24 50 43 2f 58 54 0d 0a 24 41 54 0d PC..$PC/XT..$AT.
00000310 0a 24 50 53 32 20 d0 bc d0 be d0 b4 d0 b5 d0 bb .$PS2 РјРѕРґРµ»
00000320 d1 8c 20 33 30 0d 0a 24 50 53 32 20 d0 bc d0 be СЊ 30..$PS2 РјРѕ
00000330 d0 b4 d0 b5 d0 bb d1 8c 20 35 30 20 d0 b8 d0 bb РґРµСЊ 50 РёР»
00000340 d0 b8 20 36 30 0d 0a 24 50 53 32 20 d0 bc d0 be Рё 60..$PS2 РјРѕ
00000350 d0 b4 d0 b5 d0 bb d1 8c 20 38 30 0d 0a 24 50 d0 РґРµСЊ 80..$PР
00000360 a1 6a 72 0d 0a 24 50 43 20 43 6f 6e 76 65 72 74 Ўjr..$PC Convert
00000370 69 62 6c 65 0d 0a 24 56 65 72 73 69 6f 6e 20 4d ible..$Version M
00000380 53 2d 44 4f 53 3a 20 20 2e 20 20 0d 0a 24 4f 45 S-DOS: . ..$OE
00000390 4d 20 6e 75 6d 62 65 72 20 3a 20 20 0d 0a 24 55 M number : ..$U
000003a0 73 65 72 20 73 65 72 69 61 6c 20 6e 75 6d 62 65 ser serial numbe
000003b0 72 3a 20 20 20 20 20 20 20 48 20 24 00 00 00 00 r: H $....
000003c0 24 0f 3c 09 76 02 04 07 04 30 c3 51 8a e0 e8 ef $.<.v....0ГQЉаип
000003d0 ff 86 c4 b1 04 d2 e8 e8 e6 ff 59 c3 53 8a fc e8 яtДt.ТиижяYГSЉьи
000003e0 e9 ff 88 25 4f 88 05 4f 8a c7 e8 de ff 88 25 4f йяЂ%OЂ.OЉЗиЮяЂ%O
000003f0 88 05 5b c3 51 52 32 e4 33 d2 b9 0a 00 f7 f1 80 Ђ.[ГQR2д3ТЙ..чсЪ
00000400 ca 30 88 14 4e 33 d2 3d 0a 00 73 f1 3c 00 74 04 К0Ђ.N3Т=..sс<.t.
00000410 0c 30 88 04 5a 59 c3 b4 09 cd 21 c3 b8 00 f0 8e .0Ђ.ZYГт.Н!ГЁ.рЋ
00000420 c0 26 a0 fe ff 3c ff 74 1c 3c fe 74 1e 3c fb 74 А& юя<яt.<юt.<ыt
00000430 1a 3c fc 74 1c 3c fa 74 1e 3c f8 74 26 3c fd 74 .<ьt.<ъt.<шt&<эt
00000440 28 3c f9 74 2a ba 00 00 eb 2b 90 ba 05 00 eb 25 (<щt*є..л+ђе..л%
00000450 90 ba 0d 00 eb 1f 90 ba 12 00 eb 19 90 ba 28 00 ђе..л.ђе..л.ђе(.
00000460 eb 13 90 ba 48 00 eb 0d 90 ba 5e 00 eb 07 90 ba л.ђеН.л.ђе^.л.ђе
00000470 66 00 eb 01 90 e8 9f ff c3 b4 30 cd 21 50 be 77 f.л.ђицяГт0Н!Psw
00000480 00 83 c6 10 e8 6d ff 58 8a c4 83 c6 03 e8 64 ff .ѓЖ.имяXЉДѓЖ.идя
00000490 ba 77 00 e8 81 ff be 8e 00 83 c6 13 8a c7 e8 53 ew.иЃяЋ.ѓЖ.ЊЗиS
000004a0 ff ba 8e 00 e8 70 ff bf 9f 00 83 c7 19 8b c1 e8 яєЋ.иряїц.ѓЗ.‹Би
000004b0 2a ff 8a c3 e8 14 ff 83 ef 02 89 05 ba 9f 00 e8 *яЉГи.яѓп.‰.єц.и
000004c0 55 ff c3 2b c0 50 b8 10 00 8e d8 e8 4e ff e8 a8 UяГ+АРё..ЋШиNяиЁ
000004d0 ff 32 c0 b4 4c cd 21 .. .. .. .. .. .. .. .. я2АrLH!........
000004e0 .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..
000004f0 .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..
Ready

**Отличия форматов файлов COM и EXE модулей**

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

Содержит один сегмент. Не превышает 64 Кб. Код располагается с адреса 0h.

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

«Плохой» EXE файл, так же, как и COM файла, имеет один сегмент. Код располагается с адреса 300h. С адреса 0 располагается информация для загрузчика, которая образует заголовок.

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

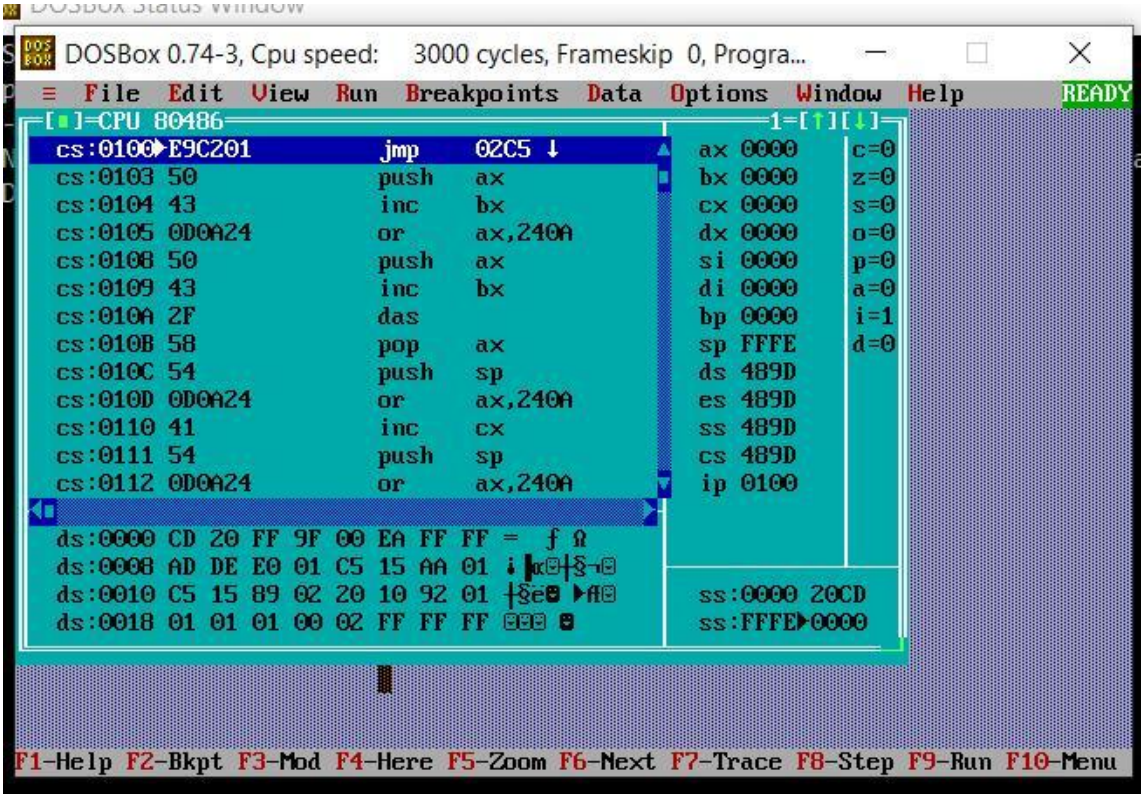
В «хорошем» EXE файле есть сегменты данных, кода и стека, в отличии от «плохого», в котором только один сегмент.

В «плохом» файле адресация всегда начинается с адреса 300h, а в «хорошем» EXE файле адресация кода начнется с 200h + размер памяти, выделенной под стек.

В COM файл резервируется 100h для PSP.При переводе в EXE должно быть 200h, поэтому получается 200h + 100h = 300h всегда в начале.

EXE-файл загружается, начиная с адреса PSP:0100h. В процессе загрузки считывается информация заголовка EXE в начале файла и выполняется перемещение адресов сегментов. Адресация начинается с адреса 200h + размер стека

Результат работы отладчика для COM файла представлен ниже




**Загрузка COM модуля в основную память**

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

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

2. Что располагается с адреса 0?

- С адреса 0 до адреса 100h располагается PSP.

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

- Сегментные регистры имеют значения 489D и указывают на PSP.

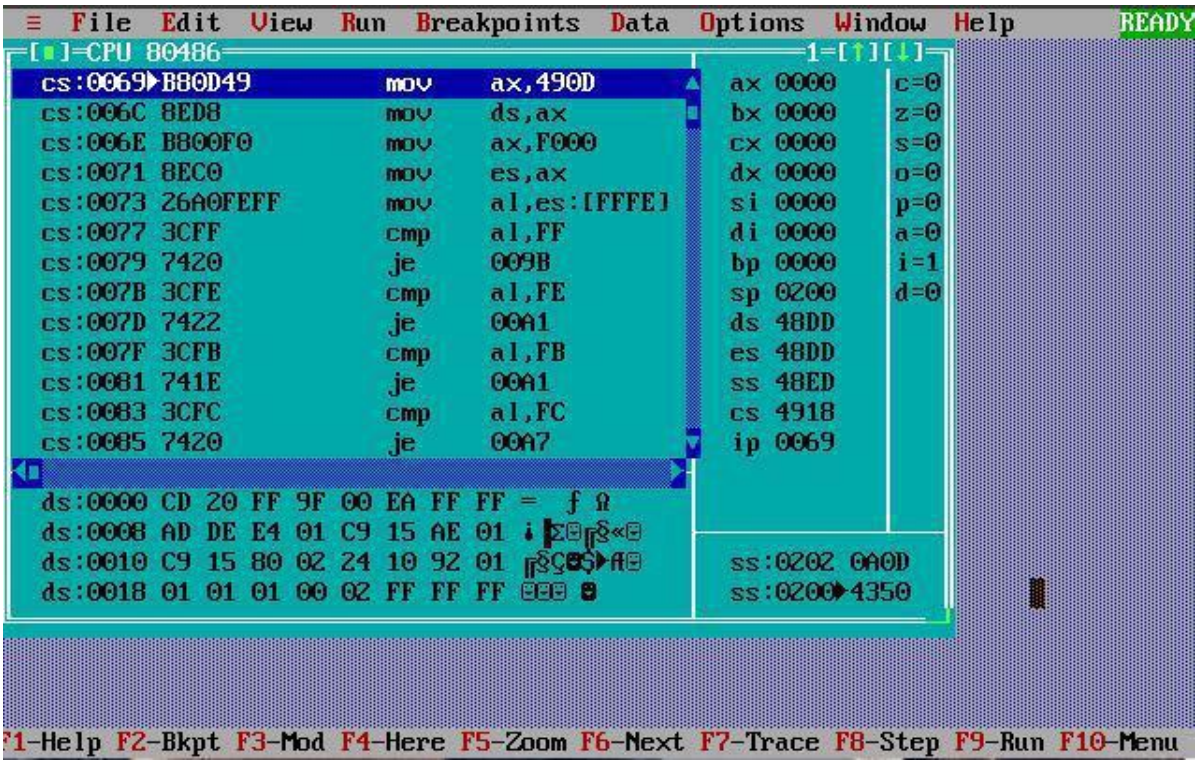



рис 9 – файл «хорошего» .EXE в отладчике

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

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

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

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

- Значения DS, ES (48DD) устанавливаются на начало PSP, SS (48ED) -

- на начало сегмента стека, CS (4918) – на начало сегмента команд.

2. На что указывают регистры DS и ES?

- Регистры DS и ES указывают на начало

- Стек определяется при помощи директивы .STACK или при помощи диррективы ASSUME, которая установит сегментый регистр SS на начало сегмента стека.

4. Как определяется точка входа?

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

**Выводы**

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

# ПРИЛОЖЕНИЕ А
# ИСХОДНЫЙ КОД TYPECOM.ASM

```
TESTPC     SEGMENT
             ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
             ORG 100H ; резервирование места для PSP
START:     JMP BEGIN

PC db   'PC',0DH,0AH,'$'
XT db   'PC/XT',0DH,0AH,'$'
tAT db   'AT',0DH,0AH,'$'
PS2_30  db    'PS2 model 30',0DH,0AH,'$'
PS2_80  db    'PS2 model 80',0DH,0AH,'$'
PCJR db   'PCjr',0DH,0AH,'$'
PC_CONVERTIBLE db   'PC Convertible',0DH,0AH,'$'
OTHER_TYPE db        'Other type:',0DH,0AH,'$'
END_LINE db '             ', 0DH,0AH,'$'

VERSION db    'Version: ',0DH,0AH,'$'
VERS db ' $'
MODIFICATION db      '.$'
VERSION2 db    'Version <2.0',0DH,0AH,'$'
OEM db   'OEM number:',0DH,0AH,'$'
SERIAL_NUMBER db 'User serial number:', 0AH, '           ', 0DH,0AH,'$'


BEGIN:
             mov ax,0F000H
             mov es,ax
             mov al,es:[0FFFEH]

             cmp al, 0FFH
             je itIsPC
             cmp al, 0FEH
             je itIsPC_XT
             cmp al, 0FBH
             je itIsPC_XT
             cmp al, 0FCH
             je itIsAT
             cmp al, 0FAH
             je itIsPS2_30
             cmp al, 0F8H
             je itIsPS2_80
             cmp al, 0FDH
             je itIsPCjr
             cmp al, 0F9H
```

```
            je itIsPCconvertible

            cmp al, 0F9H
            jne itIsOther

;-------------------------------------
; Для вывода типа
itIsPC:
            mov dx, offset PC
            jmp writeType

itIsPC_XT:
            mov dx, offset XT
            jmp writeType

itIsAT:
            mov dx, offset tAT
            jmp writeType

itIsPS2_30:
            mov dx, offset PS2_30
            jmp writeType

itIsPS2_80:
            mov dx, offset PS2_80
            jmp writeType

itIsPCjr:
            mov dx, offset PCJR
            jmp writeType

itIsPCconvertible:
            mov dx, offset PC_CONVERTIBLE
            jmp writeType

itIsOther:
            mov dx, offset OTHER_TYPE
            mov ah, 09h
            int 21h
            call BYTE_TO_HEX
            call PRINT_NUM
            mov al, ah
            call PRINT_NUM
            call PRINT_END_LINE
            jmp OS_VERSION

writeType:
            mov ah, 09h
```

```
            int 21h
            jmp OS_VERSION
;-----------------------------------------


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

OS_VERSION:
            mov ah, 30h
            int 21h

printVer:
            push ax
            cmp al, 0
            je ver2

            mov dx, offset VERSION
            push ax
            mov ah, 09h
            int 21h
            pop ax

            mov si, offset VERS
            call BYTE_TO_DEC
            add si, 1
            mov dx, offset VERS
            mov ah, 09h
            int 21h
            pop ax
            jmp numMod

ver2:
            mov dx, offset VERSION2
            mov ah, 09h
            int 21h
            pop ax

numMod:
            mov dx, offset MODIFICATION
            push ax
            mov ah, 09h
            int 21h
            pop ax
            mov si, offset END_LINE
            mov al, ah
            call BYTE_TO_DEC
            add si, 1
```

```
            mov dx, offset END_LINE
            mov ah, 09h
            int 21h

numOEM:
            mov dx, offset OEM
            push ax
            mov ah, 09h
            int 21h
            pop ax
            mov si, offset END_LINE
            mov al, bh
            call BYTE_TO_DEC
            add si, 1
            mov dx, offset END_LINE
            mov ah, 09h
            int 21h

serialNumb:

            mov di, offset SERIAL_NUMBER
            add di, 25
            mov ax, cx
            call WRD_TO_HEX
            mov al, bl
            call BYTE_TO_HEX
            sub di, 2
            mov [di], ax
            mov dx, offset SERIAL_NUMBER
            mov ah, 09h
            int 21h

            xor al, al
            mov ah, 4Ch
            int 21h



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

PRINT_END_LINE PROC near
            push ax
            mov dx, offset END_LINE
            mov ah, 09h
            int 21h
            pop ax
PRINT_END_LINE ENDP
```

```
PRINT_NUM PROC near
            ; вывод al
            push ax mov
            dx, ax mov
            ah, 02h int
            21h pop ax
            ret

PRINT_NUM ENDP



TETR_TO_HEX PROC near
   and AL,0Fh
   cmp AL,09
   jbe next
   add AL,07
next:
   add AL,30h
   ret
TETR_TO_HEX ENDP
;-------------------------------
BYTE_TO_HEX PROC near
;байт в AL переводится в два символа шест. числа в AX
   push CX
   mov AH,AL
   call TETR_TO_HEX
   xchg AL,AH
   mov CL,4
   shr AL,CL
   call TETR_TO_HEX ;в AL старшая цифра
   pop CX ;в AH младшая
   ret
BYTE_TO_HEX ENDP
;-------------------------------
WRD_TO_HEX PROC near
;перевод в 16 с/с 16-ти разрядного числа
; в AX - число, DI - адрес последнего символа
   push BX
   mov BH,AH
   call BYTE_TO_HEX
   mov [DI],AH dec
   DI
   mov [DI],AL
   dec DI mov
   AL,BH
```

```
   call BYTE_TO_HEX
   mov [DI],AH
   dec DI
   mov [DI],AL
   pop BX
   ret
WRD_TO_HEX ENDP
;--------------------------------------------------
BYTE_TO_DEC PROC near
; перевод в 10с/с, SI - адрес поля младшей цифры
   push CX
   push DX
   xor AH,AH
   xor DX,DX
   mov CX,10
loop_bd:
   div CX
   or DL,30h
   mov [SI],DL
   dec SI
   xor DX,DX
   cmp AX,10
   jae loop_bd
   cmp AL,00h
   je end_l
   or AL,30h
   mov [SI],AL
end_l:
   pop DX
   pop CX
   ret
BYTE_TO_DEC ENDP



TESTPC     ENDS
             END START
```

# ПРИЛОЖЕНИЕ В

# ИСХОДНЫЙ КОД FIRST.ASM

```
TESTPC SEGMENT
   ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING,
   SS:NOTHING ORG 100H
START: JMP BEGIN
; Данные
TYPE_PC db  'PC',0DH,0AH,'$'
TYPE_PC_XT db 'PC/XT',0DH,0AH,'$'
TYPE_AT db  'AT',0DH,0AH,'$'
TYPE_PS2_M30 db 'PS2 модель 30',0DH,0AH,'$'
TYPE_PS2_M50_60 db 'PS2 модель 50 или 60',0DH,0AH,'$'
TYPE_PS2_M80 db 'PS2 модель 80',0DH,0AH,'$'
TYPE_PC_JR db 'PCjr',0DH,0AH,'$'
TYPE_PC_CONV db 'PC Convertible',0DH,0AH,'$'

VERSIONS db 'Version of MS-DOS:  .  ',0DH,0AH,'$'
SERIAL_NUMBER db  'OEM number :  ',0DH,0AH,'$'
USER_NUMBER db  'User serial number:       H $'

; Процедуры
;--------------------------------------------------
TETR_TO_HEX PROC near
   and AL,0Fh
   cmp AL,09
   jbe next
   add AL,07
next:
   add AL,30h
   ret
TETR_TO_HEX ENDP
;--------------------------------
BYTE_TO_HEX PROC near
;байт в AL переводится в два символа шест. числа в AX
   push CX
   mov AH,AL
   call TETR_TO_HEX
   xchg AL,AH
   mov CL,4
   shr AL,CL
   call TETR_TO_HEX ;в AL старшая цифра
   pop CX ;в AH младшая
   ret
BYTE_TO_HEX ENDP
;--------------------------------
WRD_TO_HEX PROC near
;перевод в 16 с/с 16-ти разрядного числа
; в AX - число, DI - адрес последнего символа
   push BX
   mov BH,AH
   call BYTE_TO_HEX
   mov [DI],AH dec
   DI
   mov [DI],AL
   dec DI mov
   AL,BH
   call BYTE_TO_HEX
   mov [DI],AH dec
   DI
   mov [DI],AL
   pop BX
   ret
```

```
WRD_TO_HEX ENDP
;-------------------------------------------------
BYTE_TO_DEC PROC near
; перевод в 10с/с, SI - адрес поля младшей
   цифры 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
;------------------------------
WRITESTRING PROC near
   mov AH,09h
   int 21h
   ret
WRITESTRING ENDP
;------------------------------
PC_TYPE PROC near
   mov ax, 0F000H ; получаем номер модели
       mov es, ax
       mov al, es:[0FFFEH]

       cmp al, 0FFH ; начинаем стравнивать
       je isPc
       cmp al, 0FEH
       je isPc_xt
       cmp al, 0FBH
       je isPc_xt
       cmp al, 0FCH
       je isPc_at
       cmp al, 0FAH
       je isPs2_m30
       cmp al, 0F8H
       je isPs2_m80
       cmp al, 0FDH
       je isPc_jr
       cmp al, 0F9H
       je isPc_cv
isPc:
             mov dx, offset TYPE_PC
             jmp writetype
isPc_xt:
             mov dx, offset TYPE_PC_XT
             jmp writetype
isPc_at:
             mov dx, offset TYPE_AT
             jmp writetype
isPs2_m30:
             mov dx, offset TYPE_PS2_M30
```

```
             jmp writetype
isPs2_m50_60:
             mov dx, offset TYPE_PS2_M50_60
             jmp writetype
isPs2_m80:
             mov dx, offset TYPE_PS2_M80
             jmp writetype
isPc_jr:
             mov dx, offset TYPE_PC_JR
             jmp writetype
isPc_cv:
             mov dx, offset TYPE_PC_CONV
             jmp writetype
writetype:
             call WRITESTRING
       ret
PC_TYPE ENDP
;--------------------------------
OS_VER PROC near
       mov ah, 30h
       int 21h
       push ax
       ;si - индекс источника данных, операции над строками, с ds
       mov si, offset VERSIONS
       add si, 19
       call BYTE_TO_DEC
   pop ax
   mov al, ah ; в al -> 30h
   add si, 3
       call BYTE_TO_DEC
       mov dx, offset VERSIONS
       call WRITESTRING

       mov si, offset SERIAL_NUMBER
       add si, 13
       mov al, bh
       call BYTE_TO_DEC
       mov dx, offset SERIAL_NUMBER
       call WRITESTRING

       ; di - индекс назначения, используется с es
       mov di, offset USER_NUMBER
       add di, 25 mov
       ax, cx call
       WRD_TO_HEX mov
       al, bl call
       BYTE_TO_HEX sub
       di, 2
       mov [di], ax
       mov dx, offset USER_NUMBER
       call WRITESTRING
       ret
OS_VER ENDP
;-------------------------------
; Код
BEGIN:
   call PC_TYPE
   call OS_VER

   xor AL,AL
   mov AH,4Ch
   int 21H
TESTPC ENDS
END START
```