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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Операционные системы»

Тема: «Исследование структур заголовочных модулей»

Студент гр. 7381 _____ Вологдин М.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 загрузки в основную память.

Основные теоретические положения.

Тип IBM PC узнается путем считывания предпоследнего байта с ROM BIOS. Соответствие байта типу IBM PC представлено в табл. 1.

Таблица 1 – соответствие байта и типа IBM PC

Тип IBM PC	Значения байта
PC	FF
PC/XT	FE,FB
AT	FC
PS2 модель 30	FA
PS2 модель 50 или 60	FC
PS2 модель 80	F8
PCjr	FD
PC Convertible	F9

Для определения версии MS DOS следует воспользоваться функцией 30h прерывания 21h.

Входные параметры:

- mov ah, 30h
- int 21h

Выходные параметры:

- AL – номер основной версии
- AH – номер модификации
- BH – серийный номер OEM
- BL:CX – 24-битовый серийный номер пользователя

Выполнение работы.

Был написан текст исходного .COM модуля, который определяет тип PC и версию системы. Ассемблерная программа считывает предпоследний байт ROM BIOS и после сравнения его с данными в таблице выводит на экран либо идентифицированный тип PC, либо этот самый байт в шестнадцатеричном представлении.

Написан текст исходного .EXE модуля с тем же функционалом.

1. Результат выполнения «плохого» .EXE модуля:

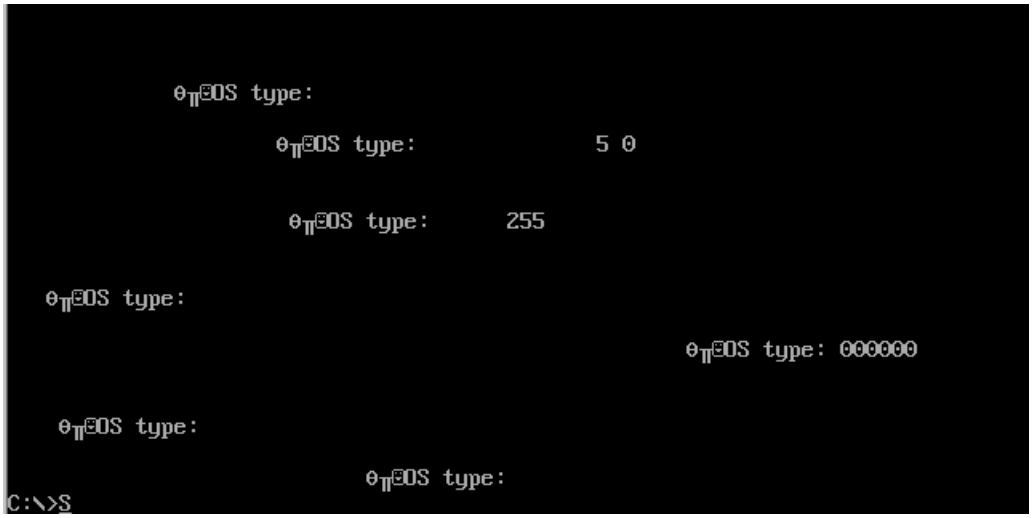


Рисунок 1 – «Плохой» .exe модуль

2. Результат выполнения «хорошего» .COM модуля:

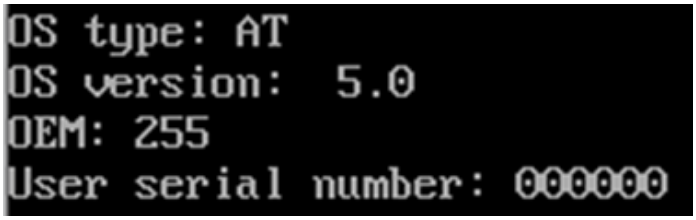


Рисунок 2 – «Хороший» .com модуль

3. Результат выполнения «хорошего» .EXE модуля:

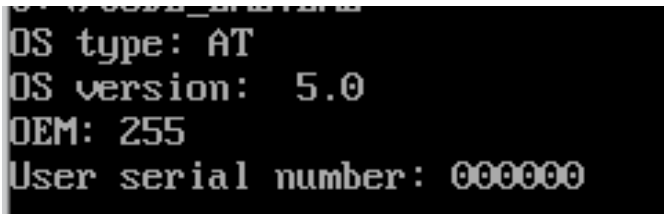


Рисунок 3 – «Хороший» .exe модуль

Выводы.

В процессе выполнения данной лабораторной работы были исследованы различия в структурах исходных текстов модулей типов .COM и .EXE, структур файлов загрузочных модулей и способов их загрузки в основную память.

Ответы на контрольные вопросы

- Отличия исходных текстов COM и EXE программ

1. Сколько сегментов должна содержать COM-программа? EXE-программа?

Ответ: Com-программа состоит из одного сегмента, в котором располагаются данные, код и стек. Exe-программа может содержать более одного сегмента.

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

Ответ: В тексте COM-программы обязательно должна присутствовать директива ORG 100h (256), так как в первых 256 байтах программы располагается PSP.

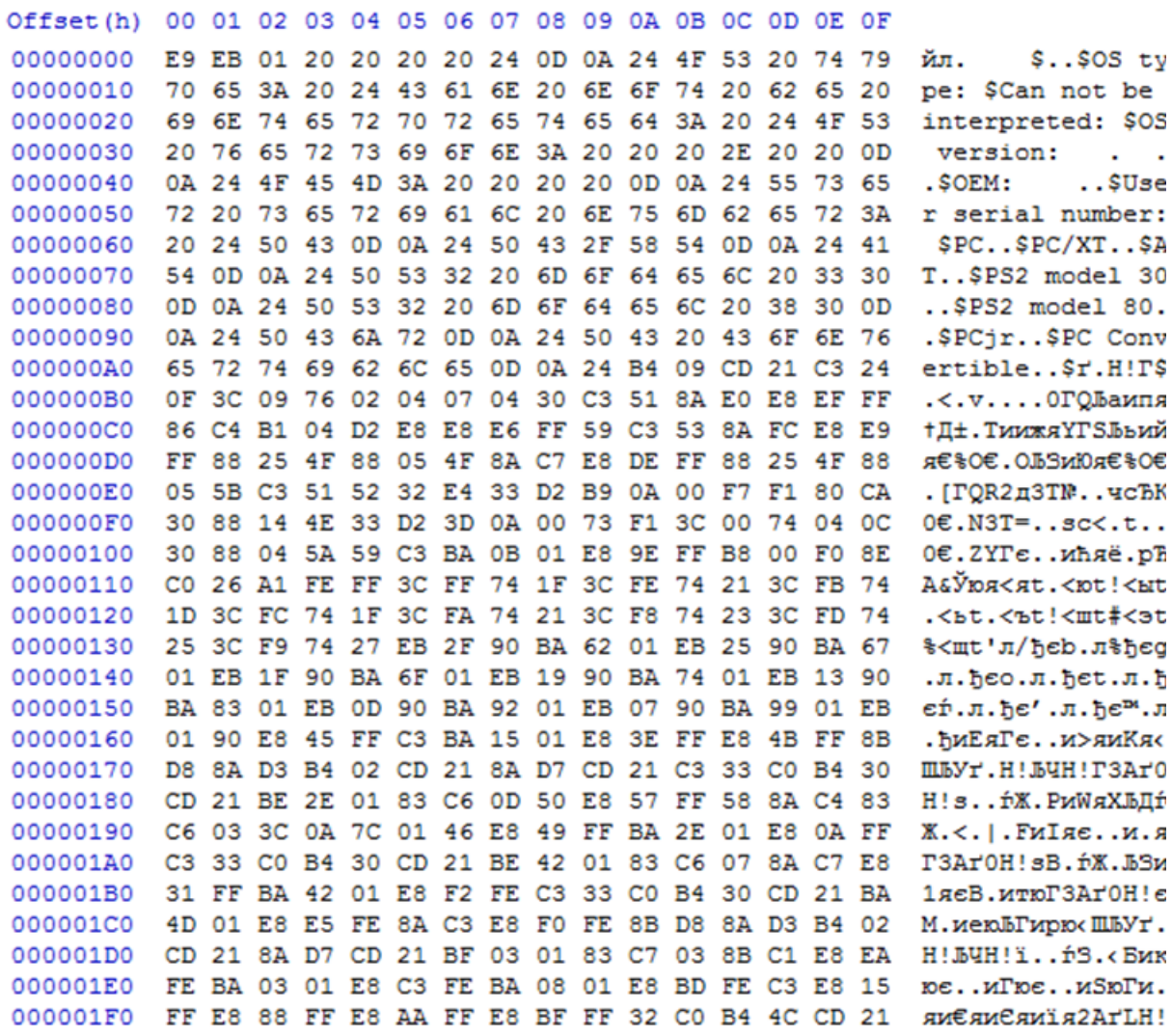
3. Все ли форматы команд можно использовать в COM-программе?

Ответ: В COM-программе нельзя использовать команды вида mov с rvalue в виде адресов сегментов и команды, содержащие дальнюю адресацию. Это связано с тем, что в COM-программе отсутствует таблица настроек (Relocation Table), с помощью которой в момент запуска программы загрузчик определяет и подставляет адреса сегментов.

- Отличия форматов файлов COM и EXE модулей

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

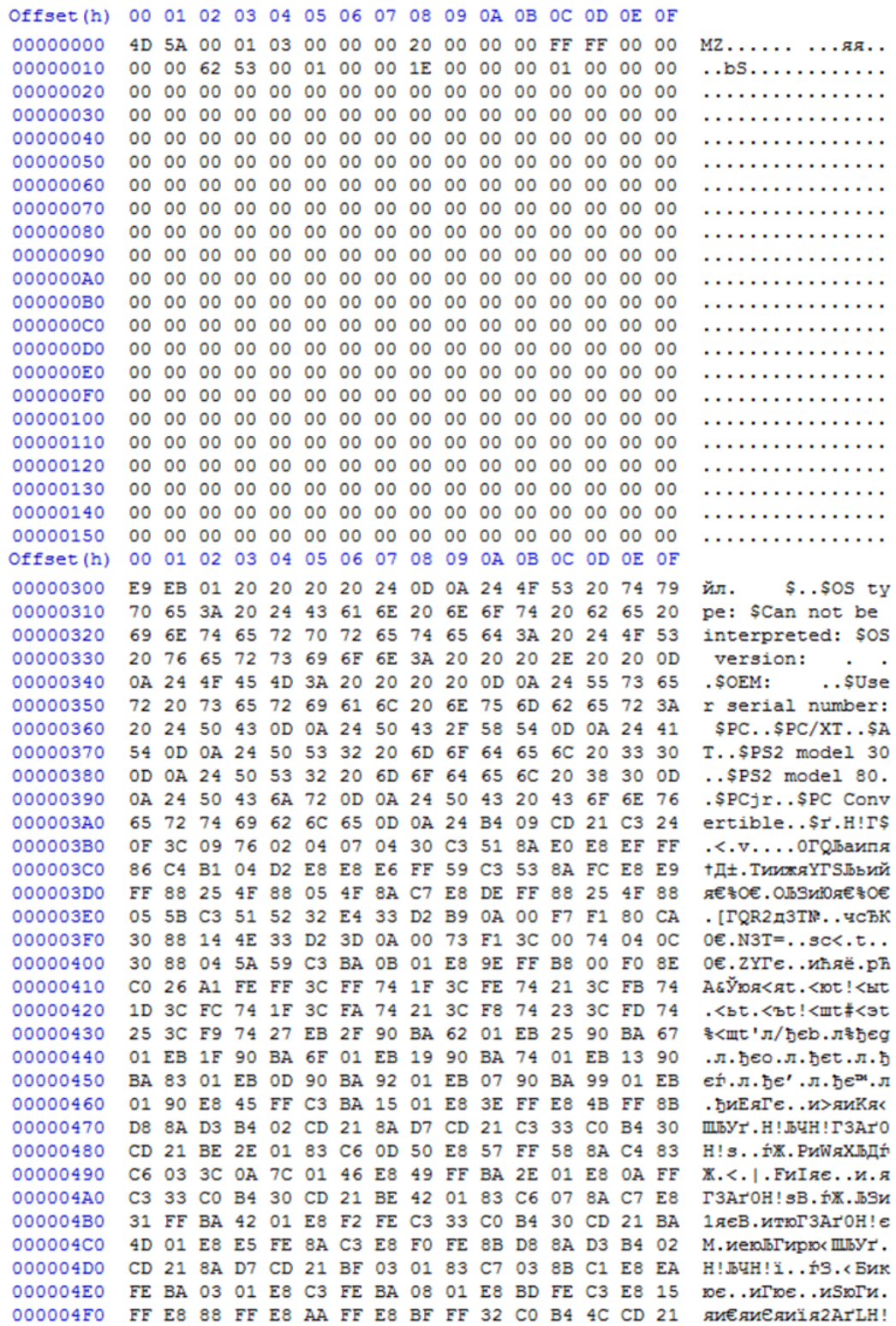
Ответ: COM-файл содержит только данные и команды. В файле загрузочного модуля команды начинаются с нулевого адреса. (рис. 4)



```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000   E9 EB 01 20 20 20 20 24 0D 0A 24 4F 53 20 74 79  йл.    $..$OS ty
00000010   70 65 3A 20 24 43 61 6E 20 6E 6F 74 20 62 65 20  pe: $Can not be
00000020   69 6E 74 65 72 70 72 65 74 65 64 3A 20 24 4F 53  interpreted: $OS
00000030   20 76 65 72 73 69 6F 6E 3A 20 20 20 2E 20 20 0D   version:   .  .
00000040   0A 24 4F 45 4D 3A 20 20 20 20 0D 0A 24 55 73 65  .$OEM:    ..$Use
00000050   72 20 73 65 72 69 61 6C 20 6E 75 6D 62 65 72 3A  r serial number:
00000060   20 24 50 43 0D 0A 24 50 43 2F 58 54 0D 0A 24 41   $PC..$PC/XT..$A
00000070   54 0D 0A 24 50 53 32 20 6D 6F 64 65 6C 20 33 30  T..$PS2 model 30
00000080   0D 0A 24 50 53 32 20 6D 6F 64 65 6C 20 38 30 0D  ..$PS2 model 80.
00000090   0A 24 50 43 6A 72 0D 0A 24 50 43 20 43 6F 6E 76  .$PCjr..$PC Conv
000000A0   65 72 74 69 62 6C 65 0D 0A 24 B4 09 CD 21 C3 24  ertible..$ґ.H!Г$
000000B0   0F 3C 09 76 02 04 07 04 30 C3 51 8A E0 E8 EF FF  .<.v....0ГQЉаипя
000000C0   86 C4 B1 04 D2 E8 E8 E6 FF 59 C3 53 8A FC E8 E9  †Д±.ТиижяYГSЉьий
000000D0   FF 88 25 4F 88 05 4F 8A C7 E8 DE FF 88 25 4F 88  я€%O€.OЉЗиЮя€%O€
000000E0   05 5B C3 51 52 32 E4 33 D2 B9 0A 00 F7 F1 80 CA  .[ГQR2д3Т№..чсЋК
000000F0   30 88 14 4E 33 D2 3D 0A 00 73 F1 3C 00 74 04 0C  0€.N3Т=..sc<.t..
00000100   30 88 04 5A 59 C3 BA 0B 01 E8 9E FF B8 00 F0 8E  0€.ZYГє..ићяё.рЋ
00000110   C0 26 A1 FE FF 3C FF 74 1F 3C FE 74 21 3C FB 74  А&Ўюя<яt.<юt!<ыt
00000120   1D 3C FC 74 1F 3C FA 74 21 3C F8 74 23 3C FD 74  .<ьt.<ъt!<шt#<эt
00000130   25 3C F9 74 27 EB 2F 90 BA 62 01 EB 25 90 BA 67  %<щt'л/ђєb.л%ђєg
00000140   01 EB 1F 90 BA 6F 01 EB 19 90 BA 74 01 EB 13 90  .л.ђєo.л.ђєt.л.ђ
00000150   BA 83 01 EB 0D 90 BA 92 01 EB 07 90 BA 99 01 EB  єѓ.л.ђє'.л.ђє™.л
00000160   01 90 E8 45 FF C3 BA 15 01 E8 3E FF E8 4B FF 8B  .ђиЕяГє..и>яиКя<
00000170   D8 8A D3 B4 02 CD 21 8A D7 CD 21 C3 33 C0 B4 30  ШЊУґ.H!ЊЧН!Г3Аґ0
00000180   CD 21 BE 2E 01 83 C6 0D 50 E8 57 FF 58 8A C4 83  H!s..ѓЖ.РиWяXЊДѓ
00000190   C6 03 3C 0A 7C 01 46 E8 49 FF BA 2E 01 E8 0A FF  Ж.<.|.FиIяє..и.я
000001A0   C3 33 C0 B4 30 CD 21 BE 42 01 83 C6 07 8A C7 E8  Г3Аґ0H!sB.ѓЖ.ЉЗи
000001B0   31 FF BA 42 01 E8 F2 FE C3 33 C0 B4 30 CD 21 BA  1яєB.итюГ3Аґ0H!є
000001C0   4D 01 E8 E5 FE 8A C3 E8 F0 FE 8B D8 8A D3 B4 02  M.иеюЊГирю‹ШЊУґ.
000001D0   CD 21 8A D7 CD 21 BF 03 01 83 C7 03 8B C1 E8 EA  H!ЊЧН!ї..ѓЗ.‹Бик
000001E0   FE BA 03 01 E8 C3 FE BA 08 01 E8 BD FE C3 E8 15  ює..иГює..иЅюГи.
000001F0   FF E8 88 FF E8 AA FF E8 BF FF 32 C0 B4 4C CD 21  яи€яиЄяиїя2АґLH!
```

Рисунок 4 – .com файл

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



```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   4D 5A 00 01 03 00 00 00 20 00 00 00 FF FF 00 00  MZ..... ...яя..
00000010   00 00 62 53 00 01 00 00 1E 00 00 00 01 00 00 00  ..bS............
00000020   00 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  ................
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000300   E9 EB 01 20 20 20 20 24 0D 0A 24 4F 53 20 74 79  йл.    $..$OS ty
00000310   70 65 3A 20 24 43 61 6E 20 6E 6F 74 20 62 65 20  pe: $Can not be
00000320   69 6E 74 65 72 70 72 65 74 65 64 3A 20 24 4F 53  interpreted: $OS
00000330   20 76 65 72 73 69 6F 6E 3A 20 20 20 2E 20 20 0D   version:   .  .
00000340   0A 24 4F 45 4D 3A 20 20 20 20 0D 0A 24 55 73 65  .$OEM:    ..$Use
00000350   72 20 73 65 72 69 61 6C 20 6E 75 6D 62 65 72 3A  r serial number:
00000360   20 24 50 43 0D 0A 24 50 43 2F 58 54 0D 0A 24 41   $PC..$PC/XT..$A
00000370   54 0D 0A 24 50 53 32 20 6D 6F 64 65 6C 20 33 30  T..$PS2 model 30
00000380   0D 0A 24 50 53 32 20 6D 6F 64 65 6C 20 38 30 0D  ..$PS2 model 80.
00000390   0A 24 50 43 6A 72 0D 0A 24 50 43 20 43 6F 6E 76  .$PCjr..$PC Conv
000003A0   65 72 74 69 62 6C 65 0D 0A 24 B4 09 CD 21 C3 24  ertible..$ґ.H!Г$
000003B0   0F 3C 09 76 02 04 07 04 30 C3 51 8A E0 E8 EF FF  .<.v....0ГQЉаипя
000003C0   86 C4 B1 04 D2 E8 E8 E6 FF 59 C3 53 8A FC E8 E9  †Д±.ТиижяYГSЉьий
000003D0   FF 88 25 4F 88 05 4F 8A C7 E8 DE FF 88 25 4F 88  яЂ%O€.OЉЗиЮяЂ%O€
000003E0   05 5B C3 51 52 32 E4 33 D2 B9 0A 00 F7 F1 80 CA  .[ГQR2д3Т№..чсЂК
000003F0   30 88 14 4E 33 D2 3D 0A 00 73 F1 3C 00 74 04 0C  0€.N3Т=..sc<.t..
00000400   30 88 04 5A 59 C3 BA 0B 01 E8 9E FF B8 00 F0 8E  0€.ZYГє..ићяё.рЋ
00000410   C0 26 A1 FE FF 3C FF 74 1F 3C FE 74 21 3C FB 74  А&Ўюя<ят.<ют!<ыт
00000420   1D 3C FC 74 1F 3C FA 74 21 3C F8 74 23 3C FD 74  .<ьт.<ът!<шт#<эт
00000430   25 3C F9 74 27 EB 2F 90 BA 62 01 EB 25 90 BA 67  %<щт'л/ђєb.л%ђєg
00000440   01 EB 1F 90 BA 6F 01 EB 19 90 BA 74 01 EB 13 90  .л.ђєo.л.ђєt.л.ђ
00000450   BA 83 01 EB 0D 90 BA 92 01 EB 07 90 BA 99 01 EB  єѓ.л.ђє'.л.ђєв„.л
00000460   01 90 E8 45 FF C3 BA 15 01 E8 3E FF E8 4B FF 8B  .ђиЕяГє..и>яиКя<
00000470   D8 8A D3 B4 02 CD 21 8A D7 CD 21 C3 33 C0 B4 30  ШЉУґ.H!ЉЧH!Г3Аґ0
00000480   CD 21 BE 2E 01 83 C6 0D 50 E8 57 FF 58 8A C4 83  H!s..ѓЖ.РиWяXЉДѓ
00000490   C6 03 3C 0A 7C 01 46 E8 49 FF BA 2E 01 E8 0A FF  Ж.<.|.FиIяє..и.я
000004A0   C3 33 C0 B4 30 CD 21 BE 42 01 83 C6 07 8A C7 E8  Г3Аґ0H!sB.ѓЖ.ЉЗи
000004B0   31 FF BA 42 01 E8 F2 FE C3 33 C0 B4 30 CD 21 BA  1яєB.итюГ3Аґ0H!є
000004C0   4D 01 E8 E5 FE 8A C3 E8 F0 FE 8B D8 8A D3 B4 02  M.иеюЉГирю‹ШЉУґ.
000004D0   CD 21 8A D7 CD 21 BF 03 01 83 C7 03 8B C1 E8 EA  H!ЉЧH!ї..ѓЗ.‹Бик
000004E0   FE BA 03 01 E8 C3 FE BA 08 01 E8 BD FE C3 E8 15  ює..иГює..иЅюГи.
000004F0   FF E8 88 FF E8 AA FF E8 BF FF 32 C0 B4 4C CD 21  яиЂяиЄяиїя2АґLH!
```

Рисунок 4 – .exe файл

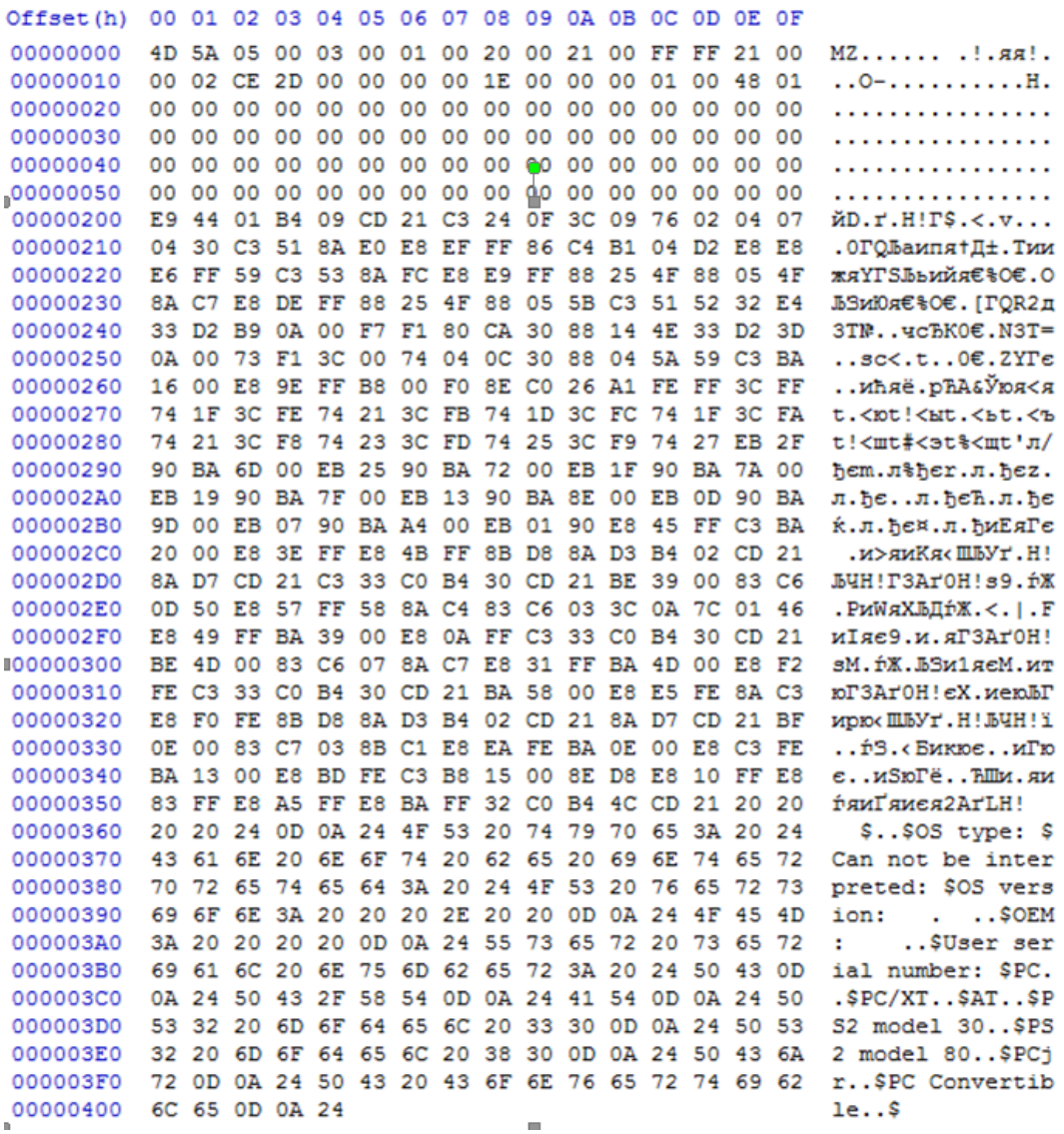
Ответ: В «плохом» EXE код и данные находятся в одном сегменте, как в COM-файле. При сравнении двух файлов можно заметить, что определённая

последовательность байт, начинающаяся с 0 адреса в COM-файле в EXE-файле начинается с адреса 300h. Это происходит по двум причинам. Во-первых, в exe-файле находится таблица настройки, которая занимает 200 байт. Во-вторых, директивой org 100h адресация команд смещается ещё на 100 байт. С нулевого адреса располагается заголовок. (рис. 5)

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

Ответ: HEX-представление «хорошего» EXE представлен на рис. 6.

В «хорошем» EXE-файле команды, стек и данные выделены в отдельные сегменты. Адресация команд начинается с 200h байта, т.к. первые 200h байт содержат таблицу настройки.



```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   4D 5A 05 00 03 00 01 00 20 00 21 00 FF FF 21 00  MZ..... .!.яя!.
00000010   00 02 CE 2D 00 00 00 00 1E 00 00 00 01 00 48 01  ..О-..........H.
00000020   00 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  ................
00000200   E9 44 01 B4 09 CD 21 C3 24 0F 3C 09 76 02 04 07  йD.ґ.H!Г$.<.v...
00000210   04 30 C3 51 8A E0 E8 EF FF 86 C4 B1 04 D2 E8 E8  .0ГQЉаипяtД±.Тии
00000220   E6 FF 59 C3 53 8A FC E8 E9 FF 88 25 4F 88 05 4F  жяYГSЉьийяЂ%OЂ.O
00000230   8A C7 E8 DE FF 88 25 4F 88 05 5B C3 51 52 32 E4  ЉЗиЮяЂ%OЂ.[ГQR2д
00000240   33 D2 B9 0A 00 F7 F1 80 CA 30 88 14 4E 33 D2 3D  3TN..чсЋK0Ђ.N3T=
00000250   0A 00 73 F1 3C 00 74 04 0C 30 88 04 5A 59 C3 BA  ..sc<.t..0Ђ.ZYГє
00000260   16 00 E8 9E FF B8 00 F0 8E C0 26 A1 FE FF 3C FF  ..ићяё.рЋА&Ўюя<я
00000270   74 1F 3C FE 74 21 3C FB 74 1D 3C FC 74 1F 3C FA  t.<юt!<ыt.<ьt.<ъ
00000280   74 21 3C F8 74 23 3C FD 74 25 3C F9 74 27 EB 2F  t!<шt#<эt%<щt'л/
00000290   90 BA 6D 00 EB 25 90 BA 72 00 EB 1F 90 BA 7A 00  ђєm.л%ђєr.л.ђєz.
000002A0   EB 19 90 BA 7F 00 EB 13 90 BA 8E 00 EB 0D 90 BA  л.ђє..л.ђєЋ.л.ђє
000002B0   9D 00 EB 07 90 BA A4 00 EB 01 90 E8 45 FF C3 BA  ќ.л.ђєЄ.л.ђиЕяГє
000002C0   20 00 E8 3E FF E8 4B FF 8B D8 8A D3 B4 02 CD 21   .и>яиКя‹ЩЬУґ.H!
000002D0   8A D7 CD 21 C3 33 C0 B4 30 CD 21 BE 39 00 83 C6  ЉЧН!Г3Аґ0H!s9.ѓЖ
000002E0   0D 50 E8 57 FF 58 8A C4 83 C6 03 3C 0A 7C 01 46  .РиWяXЉДѓЖ.<.|.F
000002F0   E8 49 FF BA 39 00 E8 0A FF C3 33 C0 B4 30 CD 21  иIяє9.и.яГ3Аґ0H!
00000300   BE 4D 00 83 C6 07 8A C7 E8 31 FF BA 4D 00 E8 F2  sM.ѓЖ.ЉЗи1яєM.ит
00000310   FE C3 33 C0 B4 30 CD 21 BA 58 00 E8 E5 FE 8A C3  юГ3Аґ0H!єX.иеюЊГ
00000320   E8 F0 FE 8B D8 8A D3 B4 02 CD 21 8A D7 CD 21 BF  ирю‹ЩЬУґ.H!ЉЧН!ї
00000330   0E 00 83 C7 03 8B C1 E8 EA FE BA 0E 00 E8 C3 FE  ..ѓЗ.‹Бикює..иГю
00000340   BA 13 00 E8 BD FE C3 B8 15 00 8E D8 E8 10 FF E8  є..иЅюГё..ЋШи.яи
00000350   83 FF E8 A5 FF E8 BA FF 32 C0 B4 4C CD 21 20 20  ѓяиЃяиєя2АґLH!
00000360   20 20 24 0D 0A 24 4F 53 20 74 79 70 65 3A 20 24    $..$OS type: $
00000370   43 61 6E 20 6E 6F 74 20 62 65 20 69 6E 74 65 72  Can not be inter
00000380   70 72 65 74 65 64 3A 20 24 4F 53 20 76 65 72 73  preted: $OS vers
00000390   69 6F 6E 3A 20 20 20 2E 20 20 0D 0A 24 4F 45 4D  ion:   .  ..$OEM
000003A0   3A 20 20 20 20 0D 0A 24 55 73 65 72 20 73 65 72  :    ..$User ser
000003B0   69 61 6C 20 6E 75 6D 62 65 72 3A 20 24 50 43 0D  ial number: $PC.
000003C0   0A 24 50 43 2F 58 54 0D 0A 24 41 54 0D 0A 24 50  .$PC/XT..$AT..$P
000003D0   53 32 20 6D 6F 64 65 6C 20 33 30 0D 0A 24 50 53  S2 model 30..$PS
000003E0   32 20 6D 6F 64 65 6C 20 38 30 0D 0A 24 50 43 6A  2 model 80..$PCj
000003F0   72 0D 0A 24 50 43 20 43 6F 6E 76 65 72 74 69 62  r..$PC Convertib
00000400   6C 65 0D 0A 24                                   le..$
```

Рисунок 5 – Хороший .exe файл

- Загрузка COM модуля в основную память

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

   Для ответа рассмотрим поэтапную загрузку модуля COM.

   1) Выделение сегмента памяти для модуля. 2) Установка всех сегментных регистров на начало выделенного сегмента памяти. 3) Построение в первых 100h байтах памяти PSP. 4)Загрузка содержимого COM-файла и присваивание регистру IP значения 100h. 5)Регистр SP устанавливается в конец сегмента.

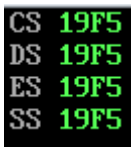
   Ответ: с адреса 100h.

2. Что располагается с адреса 0?

   Ответ: Адрес начала PSP.

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

   Ответ: Все сегментные регистры равны в данном случае 19F5 и указывают на начало PSP.



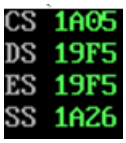
4. Как определяется стек? Какую область памяти он занимает? Какие адреса?

   Ответ: Адресом начала стека в COM-файле является адрес последнего байта выделенного сегмента (его конец), т.е. SP = FFFEh. Теоретически, регистр SP может принимать любые значения от 0000h до FFFFh.

- Загрузка «хорошего» EXE модуля в основную память

1. Как загружается «хороший» EXE? Какие значения имеют сегментные регистры? На что указывают регистры DS и ES?

Ответ: DS и ES имеют значение 19F5 и указывают на начало PSP, SS = 1A26 (начало сегмента стека), CS = 1A05 (начало сегмента команд).



2. Как определяется стек?

Ответ: Стек определяется с помощью директивы STACK; в момент запуска для стека выделяется сегмент с размером согласно этой директиве, в регистр SS заносится адрес этого сегмента.

3. Как определяется точка входа?

Ответ: Точка входа в программу определяется с помощью директивы END.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ ТЕКСТ .COM МОДУЛЯ

```
TESTPC SEGMENT
 ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
 ORG 100H
START: JMP BEGIN
; ДАННЫЕ
OSTYPE DB 'OS TYPE: $'
OSTYPENOTDEF DB 'NOT DEFINED: $'
OSVER DB 'OS VERSION:   .  ',0DH,0AH,'$'
STR_OEM DB 'OEM:    ',0DH,0AH,'$' ; ADDITIONAL 3 BYTES FOR DIGITS
SER_NUM DB 'USER SERIAL NUMBER: ','$'
STR_HEX DB '    $'
ENDL DB 0DH,0AH,'$'

PC DB 'PC',0DH,0AH,'$'
PCXT DB 'PC/XT',0DH,0AH,'$'
STR_AT DB 'AT',0DH,0AH,'$'
STR_PS2_30 DB 'PS2 MODEL 30',0DH,0AH,'$'
STR_PS2_80 DB 'PS2 MODEL 80',0DH,0AH,'$'
STR_PCJR DB 'PCJR',0DH,0AH,'$'
STR_PC_CNV DB 'PC CONVERTIBLE',0DH,0AH,'$'

PRINT PROC NEAR
     MOV AH,09H
     INT 21H
     RET
PRINT ENDP

CHECK_OS_TYPE PROC NEAR
     MOV DX, OFFSET OSTYPE
     CALL PRINT
     MOV AX,0F000H
     MOV ES,AX
```

```
MOV AX,ES:0FFFEH

CMP AL,0FFH
JE PC_
CMP AL,0FEH
JE PCXT_
CMP AL,0FBH
JE PCXT_
CMP AL,0FCH
JE LAT
CMP AL,0FAH
JE PS2_30
CMP AL,0F8H
JE PS2_80
CMP AL,0FDH
JE PCJR
CMP AL,0F9H
JE PC_CNV
JMP COT_ERR

PC_:
     MOV DX, OFFSET PC
     JMP COT_END
PCXT_:
     MOV DX, OFFSET PCXT
     JMP COT_END
LAT:
     MOV DX, OFFSET STR_AT
     JMP COT_END
PS2_30:
     MOV DX, OFFSET STR_PS2_30
     JMP COT_END
PS2_80:
     MOV DX, OFFSET STR_PS2_80
```

```
            JMP COT_END
      PCJR:
            MOV DX, OFFSET STR_PCJR
            JMP COT_END
      PC_CNV:
            MOV DX, OFFSET STR_PC_CNV
            JMP COT_END

      COT_END:
      CALL PRINT
      RET

      COT_ERR:
      MOV DX, OFFSET OSTYPENOTDEF
      CALL PRINT
      CALL BYTE_TO_HEX
      MOV BX,AX
      MOV DL,BL
      MOV AH,02H
      INT 21H
      MOV DL,BH
      INT 21H
      RET
CHECK_OS_TYPE ENDP

CHECK_OS_VERSION PROC NEAR
      XOR AX,AX
      MOV AH,30H
      INT 21H

      MOV SI,OFFSET OSVER
      ADD SI,13
      PUSH AX
      CALL BYTE_TO_DEC
```

```
POP AX
MOV AL,AH
ADD SI,3
CMP AL,10
JL COV_ONE_DIGIT_L
INC SI
COV_ONE_DIGIT_L:
CALL BYTE_TO_DEC

MOV DX,OFFSET OSVER
CALL PRINT

MOV SI,OFFSET STR_OEM
ADD SI,7
MOV AL,BH
CALL BYTE_TO_DEC

MOV DX,OFFSET STR_OEM
CALL PRINT

MOV DX,OFFSET SER_NUM
CALL PRINT

MOV  AL,BL
CALL BYTE_TO_HEX
MOV BX,AX
MOV DL,BL
MOV AH,02H
INT 21H
MOV DL,BH
INT 21H

MOV DI,OFFSET STR_HEX
```

```
      ADD DI,3
      MOV AX,CX
      CALL WRD_TO_HEX
      MOV DX,OFFSET STR_HEX
      CALL PRINT

      MOV DX,OFFSET ENDL
      CALL PRINT

      RET
CHECK_OS_VERSION 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
      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
```

```
        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
        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

BEGIN:
      CALL CHECK_OS_TYPE
      CALL CHECK_OS_VERSION
      XOR AL,AL
      MOV AH,4CH
      INT 21H
TESTPC ENDS
 END START
```