

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. 0382

Самулевич В.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

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

Задание.

Шаг 1. Напишите текст исходного **.COM** модуля, который определяет тип РС и версию системы. Это довольно простая задача и для тех, кто уже имеет опыт программирования на ассемблере, это будет небольшой разминкой. Для тех, кто раньше не сталкивался с программированием на ассемблере, это неплохая задача для первого опыта. За основу возьмите шаблон, приведенный в разделе «Основные сведения». Необходимые сведения о том, как извлечь требуемую информацию, представлены в следующем разделе. Ассемблерная программа должна читать содержимое предпоследнего байта ROM BIOS, по таблице, сравнивая коды, определять тип РС и выводить строку с названием модели. Если код не совпадает ни с одним значением, то двоичный код переводиться в символьную строку, содержащую запись шестнадцатеричного числа и выводиться на экран в виде соответствующего сообщения. Затем определяется версия системы. Ассемблерная программа должна по значениям регистров AL и AH формировать текстовую строку в формате xx.yy, где xx – номер основной версии, а yy - номер модификации в десятичной системе счисления, формировать строки с серийным номером OEM и серийным номером пользователя. Полученные строки выводятся на экран. Отладьте полученный исходный модуль. Результатом выполнения этого шага будет «хороший» **.COM** модуль, а также необходимо построить «плохой» **.EXE**, полученный из исходного текста для **.COM** модуля.

Шаг 2. Напишите текст исходного **.EXE** модуля, который выполняет те же функции, что и модуль в Шаге 1 и постройте и отладьте его. Таким образом, будет получен «хороший» **.EXE**.

Шаг 3. Сравните исходные тексты для .COM и .EXE модулей. Ответьте на контрольные вопросы «Отличия исходных текстов COM и EXE программ».

Шаг 4. Запустите FAR и откройте (F3/F4) файл загрузочного модуля .COM и файл «плохого» .EXE в шестнадцатеричном виде. Затем откройте (F3/F4) файл загрузочного модуля «хорошего» .EXE и сравните его с предыдущими файлами. Ответьте на контрольные вопросы «Отличия форматов файлов COM и EXE модулей».

Шаг 5. Откройте отладчик TD.EXE и загрузите .COM. Ответьте на контрольные вопросы «Загрузка COM модуля в основную память». Представьте в отчете план загрузки модуля .COM в основную память.

Шаг 6. Откройте отладчик TD.EXE и загрузите «хороший» .EXE. Ответьте на контрольные вопросы «Загрузка «хорошего» EXE модуля в основную память».

Шаг 7. Оформление отчета в соответствии с требованиями. В отчете необходимо привести скриншоты. Для файлов их вид в шестнадцатеричном виде, для загрузочных модулей – в отладчике.

Выполнение работы.

Для выполнения поставленной задачи были реализованы две процедуры: DOS_TYPE и PC_TYPE.

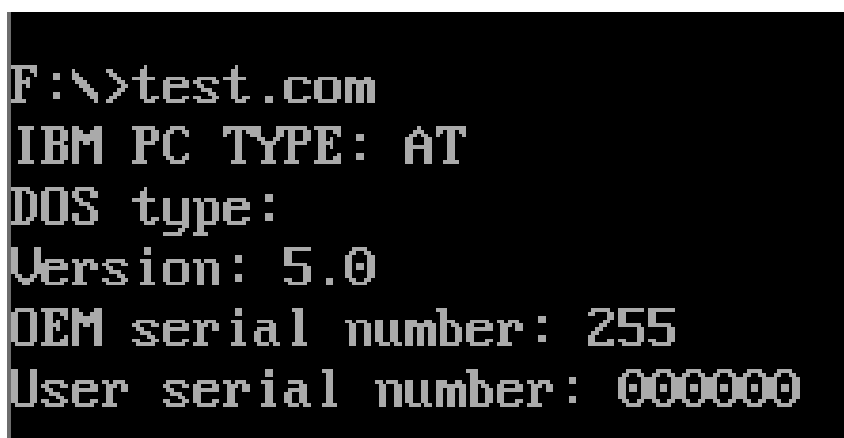
PC_TYPE определяет и выводит на экран тип PC. Процесс определения типа состоит из вызова прерывания int 15h с параметром AH = 0C0h. В результате его работы в ES:BX заносится адрес начала таблицы конфигурации, в которой, по смещению 2, располагается код IBM PC.

После получения кода, процедура, с помощью команд стр и je, находит строку, соответствующую его значению, после чего выводит ее на экран, с

помощью прерывания int 21h (AH = 09h) (Все возможные строки типов PC определены в самом начале сегмента кода).

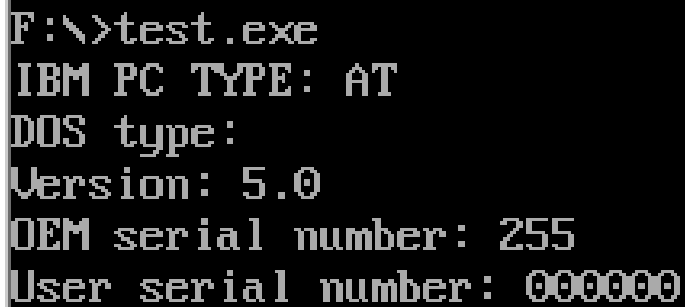
Процедура DOS_TYPE отвечает за определение и вывод основных параметров MS DOS. Их получение осуществляется с помощью прерывания int 21h (AH = 30h): в результате его работы в AL записывается номер основной версии, в AH – номер модификации, в BH – серийный номер OEM, а BL:CH – 24 битовый серийный номер пользователя. После получения требуемых значений, для них необходимо составить строковое представление. Для этих целей были созданы процедуры WRITE_BYTE, WRD_TO_HEX и WORD_TO_DEC (16ричное строковое представление для байта, 16ричное строковое представление для слова и десятичное строковое представление для слова соответственно). После преобразования параметров в строки, они выводятся на экран, вместе с поясняющим их значения текстом.

На основе написанных процедур было создано файла: один для .COM (test.asm), а другой – для .EXE (test_exe.asm). После этого версия для .COM была скомпилирована и как .COM, и как .EXE. В итоге получили хороший .COM и плохой .EXE файлы. Результаты их работы представлены на рис. 1 и рис. 2.



```
F:\>test.com
IBM PC TYPE: AT
DOS type:
Version: 5.0
OEM serial number: 255
User serial number: 000000
```

Рис. 1: Результат работы .COM файла.

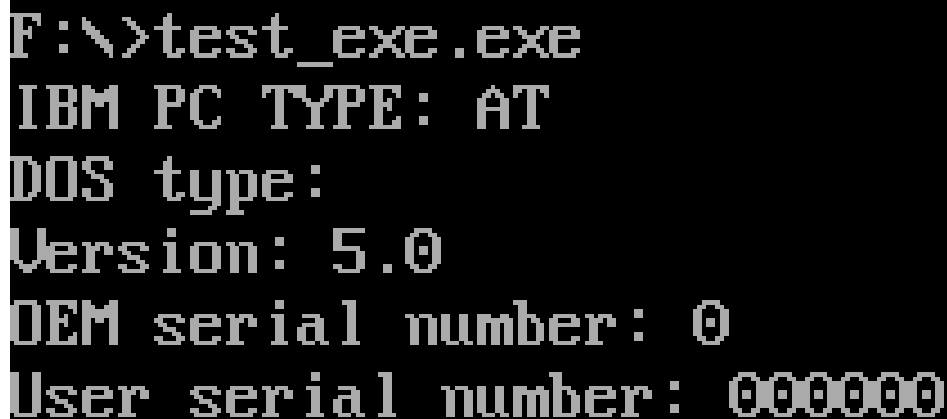


```
F:\>test.exe
IBM PC TYPE: AT
DOS type:
Version: 5.0
OEM serial number: 255
User serial number: 000000
```

Рис. 2: Результат работы “плохого” .EXE файла.

Как можно заметить, несмотря на то, что test.asm изначально создавался под .COM формат, он корректно отработал, даже будучи собранным под .EXE .

После сборок test.asm была осуществлена сборка test_exe.asm(он отличается от test.asm тем, что дополнительно содержит сегменты данных и стека, а также его сегмент кода начинается с загрузки адреса возврата в стек и инициализацией сегментного регистра DS).В результате был получен “хороший” .EXE файл. Результат его работы представлен на рис. 3.



```
F:\>test_exe.exe
IBM PC TYPE: AT
DOS type:
Version: 5.0
OEM serial number: 0
User serial number: 000000
```

Рис. 3: Результат работы “хорошего” .EXE файла.

Как видно из рисунка, хороший .EXE файл также отработал корректно.

Ответы на контрольные вопросы:

Отличия исходных текстов .COM и .EXE программ

1) Сколько сегментов должна содержать .COM программа?

.COM программа должна содержать только один сегмент.

2) EXE программа?

EXE программа может состоять из любого количества сегментов. При этом у нее всегда должен быть сегмент кода. Также обычно у .EXE программ есть сегменты данных и стека.

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

Директива ORG 100h для смещения всех адресов программы на 256(Т.к. в первые 256 байт сегмента записывается PSP), директива ASSUME для связывания меток с сегментом, в котором они находятся.

4) Все ли форматы команд можно использовать в .COM файле?

Нет. Программы с указанием адреса сегмента не могут быть выполнены, т.к. он становится известным только при загрузке программы, а .COM не может предоставить загрузчику перечня всех сегментных ссылок.

Отличие форматов файлов .COM и .EXE модулей

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

Вся .COM программа располагается в одном сегменте. В нем лежат данные и код, а также автоматически создается стек. Код начинается с адреса 0000, но будет загружен в основную память со смещением, начиная с адреса 100h.

```

C:\loc\TEST.COM
00000000: E9 BF 00 49 42 4D 20 50 43 20 54 59 50 45 3A 20 йї IBM PC TYPE:
00000001: 24 50 43 0D 0A 24 50 43 2F 58 54 0D 0A 24 41 54 $PC$PC/XT$AT
00000002: 0D 0A 24 50 53 32 2C 20 6D 6F 64 65 6C 20 33 30 $PS2, model 30
00000003: 0D 0A 24 50 53 32 20 6D 6F 64 65 6C 20 38 30 0D $PS2 model 80
00000004: 0A 24 50 43 6A 72 5C 6E 24 50 43 20 63 6F 6E 76 $PCjr\n$PC conv
00000005: 65 72 74 69 62 6C 65 0D 0A 24 00 00 00 00 00 00 ertible$
00000006: 00 00 00 00 2E 24 0D 0A 24 44 4F 53 20 74 79 70 $.DOS typ
00000007: 65 3A 0D 0A 24 56 65 72 73 69 6F 6E 3A 20 24 6C e:$Version: $l
00000008: 65 73 73 20 74 68 65 6E 20 32 2E 30 0D 0A 24 4F ess then 2.0$0
00000009: 45 4D 20 73 65 72 69 61 6C 20 6E 75 6D 62 65 72 EM serial number
0000000A: 3A 20 24 0D 0A 55 73 65 72 20 73 65 72 69 61 6C : $User serial
0000000B: 20 6E 75 6D 62 65 72 3A 20 20 20 20 20 20 0D number:
0000000C: 0A 24 E8 8F 00 E8 0D 00 32 C0 B4 4C CD 21 50 B4 $иЦ и 2ArLH!Pr
0000000D: 09 CD 21 58 C3 50 53 51 52 57 B4 30 CD 21 BA 69 oh!XGPSQRWr0H!ei
0000000E: 01 E8 EA FF BA 75 01 E8 E4 FF 3D 00 00 75 09 BA 0икаяеиудя= уое
0000000F: 7F 01 E8 D9 FF EB 29 90 51 8A EC B4 00 E8 B6 00 0ищял)ћQьмг и
00000010: BA 5A 01 E8 C8 FF BA 64 01 E8 C2 FF B0 00 8A E5 єZ0иИяд0иВя° ъе
00000011: E8 A3 00 BA 5A 01 E8 B5 FF BA 66 01 E8 AF FF 59 иJ єZ0имяеф0иІяY
00000012: B8 00 00 8A C7 E8 8E 00 BA 8F 01 E8 A0 FF BA 5A ё Ъзић єЦ0и яєZ
00000013: 01 E8 9A FF BF A3 01 83 C7 1B 8B C1 E8 F0 00 4F 0иьяіJ0f3←Бир 0
00000014: B8 00 00 8A C3 E8 DE 00 BA A3 01 E8 80 FF 5F 5A ё ЬгиЮ єJ0иТья_Z
00000015: 59 5B 58 C3 50 53 52 B4 C0 CD 15 06 1F 8B 47 02 Y[XGPSRG'AH$▲▼<G0
00000016: 0E 1F BA 03 01 E8 66 FF 3C FF 74 1C 3C FE 74 1E љѳєѳ0ифя<ятL<ют▲
00000017: 3C FB 74 1A 3C FC 74 1C 3C FA 74 1E 3C F8 74 20 <ыт→<ьтL<ьт▲<шт
00000018: 3C FD 74 22 3C F9 74 24 BA 11 01 EB 22 90 BA 16 <эт"<шт$є←0л"ћє=
00000019: 01 EB 1C 90 BA 1E 01 EB 16 90 BA 23 01 EB 10 90 0лLћє▲0л=ћє#0л►ћ
0000001A: BA 33 01 EB 0A 90 BA 42 01 EB 04 90 BA 49 01 E8 єЗ0лѳћєВ0л♦ћєІ0и
0000001B: 1C FF 5A 5B 58 C3 53 51 52 56 57 BE 00 00 BB 5A ЛяZ[XGPSQRVWs »Z
0000001C: 01 8B F8 B8 01 00 83 FF 00 75 07 C6 00 30 46 EB 0<шє0 гя и•Ж 0Фл
0000001D: 30 90 3B C7 73 07 BA 0A 00 F7 E2 EB F5 B9 0A 00 0ћ;Зс•єѳ чвлхNѳ
0000001E: F7 F1 8B C8 8B C7 F7 F1 05 30 00 88 00 46 83 F9 чс<И<Зчс♦0 є Fғщ
0000001F: 01 74 0E 8B C2 91 BA 00 00 BF 0A 00 F7 F7 91 EB 0т.ѳ•B'є іѳ чч'л
00000020: E5 C6 00 24 5F 5E 5A 59 5B C3 24 0F 3C 09 76 02 еЖ $_^ZY[Г$о<ov0
00000021: 04 07 04 30 C3 51 8A E0 E8 EF FF 86 C4 B1 04 D2 ♦♦♦0ГQьаипя†д†♦Т
00000022: E8 E8 E6 FF 59 C3 E8 EC FF 88 25 4F 88 05 C3 53 иижяYГимяє%0є♦ГS
00000023: 8A FC E8 E0 FF 88 25 4F 88 05 4F 8A C7 E8 D5 FF льиаяє%0є♦0љзиХя
00000024: 88 25 4F 88 05 5B C3 88 05 4F 8A C7 E8 D5 FF є%0є♦[Г

```

Рис. 4: Структура .COM файла.

2) Какова структура плохого .EXE? С какого адреса располагается код?

Плохой .EXE также состоит из одного сегмента. Перед этим сегментом располагается заголовок (0000h - 0200h), а затем 256 байтовое смещение (0200h – 0299h). Код начинается с адреса 00300h.

```

view TEST.EXE - Far 3.0.5959.0 x86
C:\oc\TEST.EXE
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 BF 00 49 42 4D 20 50 43 20 54 59 50 45 3A 20
0000000310: 24 50 43 0D 0A 24 50 43 2F 58 54 0D 0A 24 41 54
0000000320: 0D 0A 24 50 53 32 2C 20 6D 6F 64 65 6C 20 33 30
0000000330: 0D 0A 24 50 53 32 20 6D 6F 64 65 6C 20 38 30 0D
0000000340: 0A 24 50 43 6A 72 5C 6E 24 50 43 20 63 6F 6E 76
0000000350: 65 72 74 69 62 6C 65 0D 0A 24 00 00 00 00 00 00
0000000360: 00 00 00 00 2E 24 0D 0A 24 44 4F 53 20 74 79 70
0000000370: 65 3A 0D 0A 24 56 65 72 73 69 6F 6E 3A 20 24 6C
0000000380: 65 73 73 20 74 68 65 6E 20 32 2E 30 0D 0A 24 4F
0000000390: 45 4D 20 73 65 72 69 61 6C 20 6E 75 6D 62 65 72
00000003A0: 3A 20 24 0D 0A 55 73 65 72 20 73 65 72 69 61 6C
00000003B0: 20 6E 75 6D 62 65 72 3A 20 20 20 20 20 20 20 0D
00000003C0: 0A 24 E8 8F 00 E8 0D 00 32 C0 B4 4C CD 21 50 B4
00000003D0: 09 CD 21 58 C3 50 53 51 52 57 B4 30 CD 21 BA 69
00000003E0: 01 E8 EA FF BA 75 01 E8 E4 FF 3D 00 00 75 09 BA
00000003F0: 7F 01 E8 D9 FF EB 29 90 51 8A EC B4 00 E8 B6 00
0000000400: BA 5A 01 E8 C8 FF BA 64 01 E8 C2 FF B0 00 8A E5
0000000410: E8 A3 00 BA 5A 01 E8 B5 FF BA 66 01 E8 AF FF 59
0000000420: B8 00 00 8A C7 E8 8E 00 BA 8F 01 E8 A0 FF BA 5A
0000000430: 01 E8 9A FF BF A3 01 83 C7 1B 8B C1 E8 F0 00 4F
0000000440: B8 00 00 8A C3 E8 DE 00 BA A3 01 E8 80 FF 5F 5A
0000000450: 59 5B 58 C3 50 53 52 B4 C0 CD 15 06 1F 8B 47 02
0000000460: 0E 1F BA 03 01 E8 66 FF 3C FF 74 1C 3C FE 74 1E
0000000470: 3C FB 74 1A 3C FC 74 1C 3C FA 74 1E 3C F8 74 20
0000000480: 3C FD 74 22 3C F9 74 24 BA 11 01 EB 22 90 BA 16
0000000490: 01 EB 1C 90 BA 1E 01 EB 16 90 BA 23 01 EB 10 90
00000004A0: BA 33 01 EB 0A 90 BA 42 01 EB 04 90 BA 49 01 E8

```

Рис. 5: Структура “плохого” .EXE файла.

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

В начале файла располагается заголовок, а далее идут сегменты в порядке их описания. Хороший .EXE отличается от плохого, во-первых, отсутствием дополнительного 256 байтного смещения (т.к. в нем нет директивы ORG 100h), а

во-вторых, количеством сегментов- у хорошего .EXE их 3(стек, данные, код), а у плохого — только один(код).

```
view TEST_EXE.EXE - Far 3.0.5959.0 x86
C:\oc\TEST_EXE.EXE
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: 49 42 4D 20 50 43 20 54 59 50 45 3A 20 24 50 43 IBM PC TYPE: $PC
0000000230: 0D 0A 24 50 43 2F 58 54 0D 0A 24 41 54 0D 0A 24 $PC/XT$AT$
0000000240: 50 53 32 2C 20 6D 6F 64 65 6C 20 33 30 0D 0A 24 PS2, model 30$
0000000250: 50 53 32 20 6D 6F 64 65 6C 20 38 30 0D 0A 24 50 PS2 model 80$P
0000000260: 43 6A 72 5C 6E 24 50 43 20 63 6F 6E 76 65 72 74 Cjr\n$PC convert
0000000270: 69 62 6C 65 0D 0A 24 00 00 00 00 00 00 00 00 00 ible$
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: 44 4F 53 20 74 79 70 65 3A 0D 0A 24 56 65 72 73 .,$$
00000002F0: 69 6F 6E 3A 20 24 6C 65 73 73 20 74 68 65 6E 20 DOS type:$Vers
0000000300: 32 2E 30 0D 0A 24 4F 45 4D 20 73 65 72 69 61 6C ion: $less then
0000000310: 20 6E 75 6D 62 65 72 3A 20 24 0D 0A 55 73 65 72 2.0$OEM serial
0000000320: 20 73 65 72 69 61 6C 20 6E 75 6D 62 65 72 3A 20 number: $User
0000000330: 20 20 20 20 20 20 0D 0A 24 00 00 00 00 00 00 00 serial number:
0000000340: 1E 2B C0 50 B8 02 00 8E D8 E8 8A 00 E8 08 00 CB $
0000000350: 50 B4 09 CD 21 58 C3 50 53 51 52 57 B4 30 CD 21 ▲+APёё ЁШиль и Л
ProH!XГPQRWr0H!
```

Рис.5: Структура “хорошего” .EXE файла.

Загрузка COM модуля в основную память

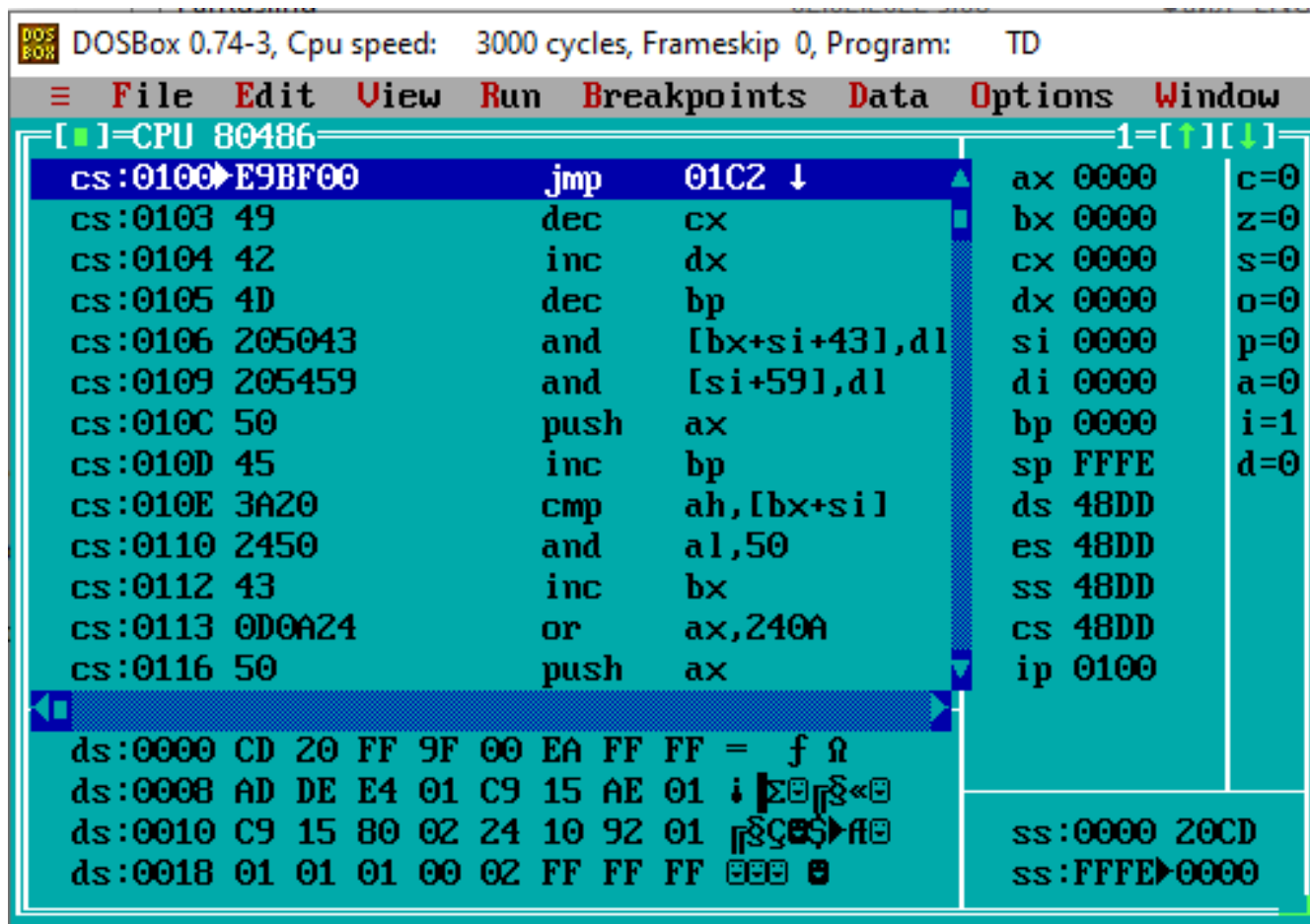


Рис. 6: отладка .COM.

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

В основной памяти выделяется один сегмент, после чего в его первые 256 бит записывается PSP. После PSP уже заносится непосредственно код программы (т.е. начиная с адреса 100h).

2) Что располагается с адреса 0?
PSP.

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

Все сегментные регистры (CS, DS и SS), указывают на начало единственного сегмента, т.е. на начало PSP.

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

Начало стека расположено в конце сегмента программы, т.е. по адресу SS::FFFE. Стек заполняется снизу вверх, до адреса SS:0000 (после этого стек выйдет из выделенного под программу сегмента, что вызовет ошибку).

Загрузка хорошего .EXE модуля в основную память

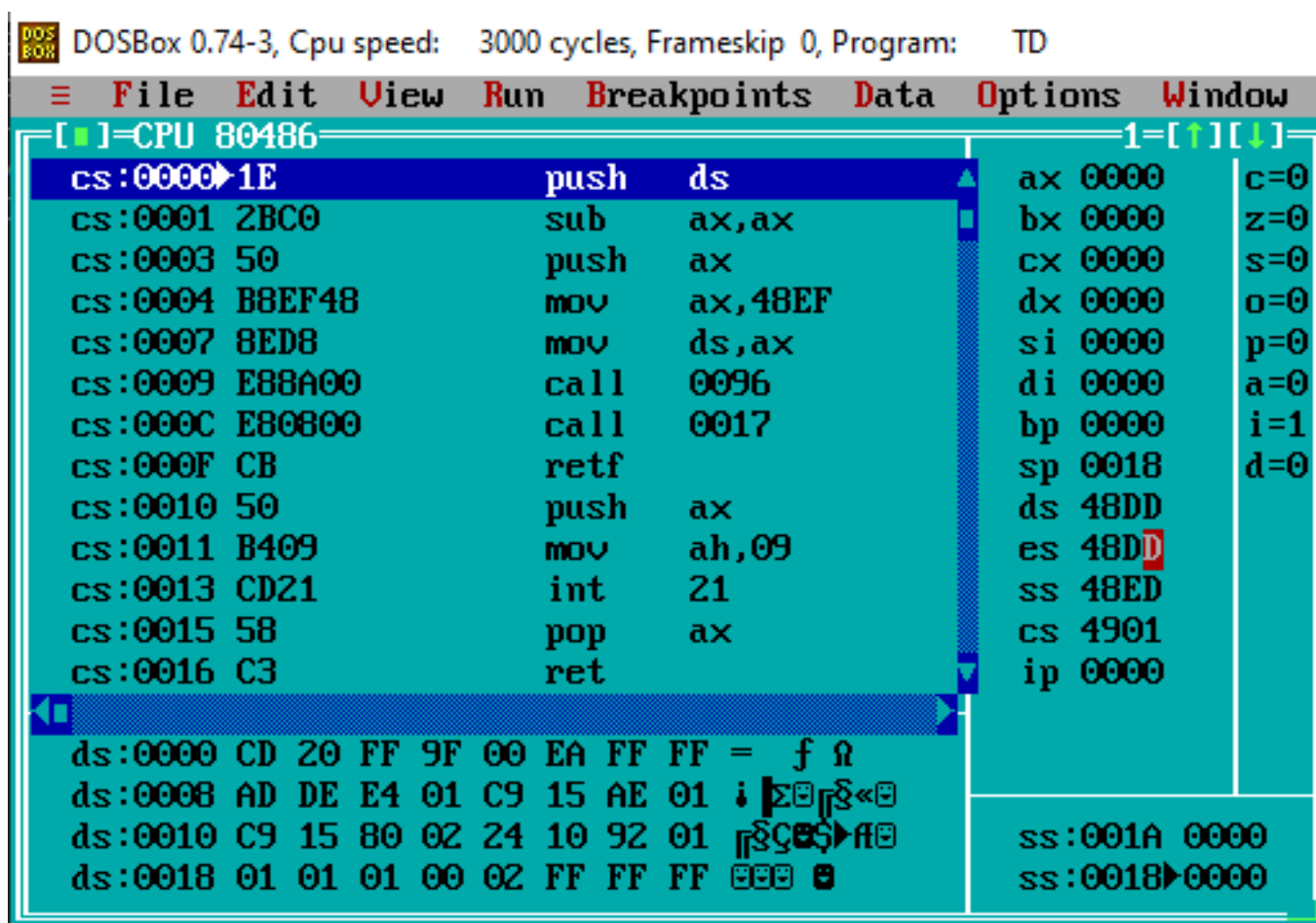


Рис. 7: отладка “хорошего” .EXE.

- 1) Как загружается хороший .EXE? Какие значения имеют сегментные регистры?

Сначала в основную память загружается PSP, после чего загружается сам .exe модуль(в соответствии с информацией в

заголовке). Изначально CS хранит адрес начала сегмента кода, SS – сегмента стека, а DS и ES – начала PSP.

2) На что указывают регистры DS и ES?

На начало PSP

3) Как определяется стек?

Инициализировать регистр SS можно с помощью директивы STACK, или перемещением в него адреса сегмента с помощью команды mov.

4) Как определяется точка входа?

Директивой END.

Выводы.

Были исследованы различия в структурах исходных текстов модулей типов **.COM** и **.EXE**, а также структуры файлов загрузочных модулей и способы их загрузки в основную память.