

**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.**

Факультет: КТИ

Дата выполнения работы: 14.02.2021

Студент гр. 9381

\_\_\_\_\_

Семенов А. Н.

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.

## **Задание.**

Написать тексты исходных **.COM** и **.EXE** модулей на языке ассемблера, которые определяют тип РС и версию системы. Сравнить исходные тексты. Произвести трансляцию и сборку этих модулей с получением хороших **.COM** и **.EXE** модулей и плохого **.EXE** модуля. Изучить полученные загрузочные модули, произвести их сравнительный анализ. Изучить загрузки модулей в основную память с помощью отладчика **TD.EXE**.

## **Функции и структуры данных.**

**TETR\_TO\_HEX** – переводит значение 4 младших битов регистра **al** в цифру 16-ричной системы счисления в виде символа, которая кладется в регистр **al**.

**BYTE\_TO\_HEX** – переводит значение байта, содержащегося в регистре **al** в двухсимвольное число в шестнадцатеричной системе счисления, которые кладется в регистр **ax**: код первого символа в **al**, второго в **ah**.

**BYTE\_TO\_DEC** – переводит значение байта, содержащегося в регистре **al** в символьное представление числа в десятичной системе счисления, которое кладется в память по адресу **si**.

**WRD\_TO\_HEX** – переводит значение регистра **AX** в шестнадцатеричное число в виде 4 символов, которые кладутся в память по адресу **di**.

## **Последовательность действий, выполняемых программой.**

1. В регистр **ES** и **BX** записываются соответственно адреса места в памяти, где хранится шестнадцатеричный байт – тип РС. При обращении к данному месту памяти в регистр **AL** кладется соответствующий байт.

2. Производятся сравнения байта в **AL** с существующими значениями, определение и печать на экран информации о типе РС в строковом виде.

3. Далее в регистр AH кладется число 30h и вызывается прерывание int 21h, которое при данном аргументе раскидывает следующие значения по регистрам:

AL - номер основной версии. Если 0, то < 2.0

AH - номер модификации

BH - серийный номер OEM (Original Equipment Manufacturer)

BL:CH - 24-битовый серийный номер пользователя.

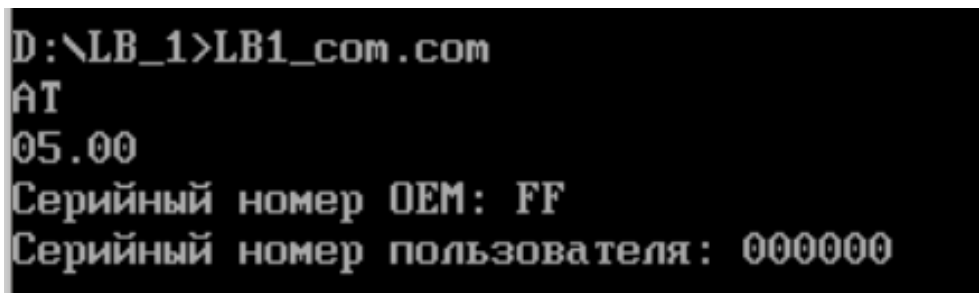
4. Затем в регистры SI и DI кладутся адреса памяти в строках вывода, предназначенные для вставки соответствующих номеров.

5. Производятся вызовы процедур, переводящих значения номеров в строковые форматы и записывающих их в соответствующие ячейки памяти по адресам, содержащимся в регистрах DI и SI. Перед каждым вызовом процедуры необходимый для перевода в символьную информацию номер кладется в регистр AL или AH.

6. Производится печать на экран строки с информацией о серийных номерах, номера версии и модификации.

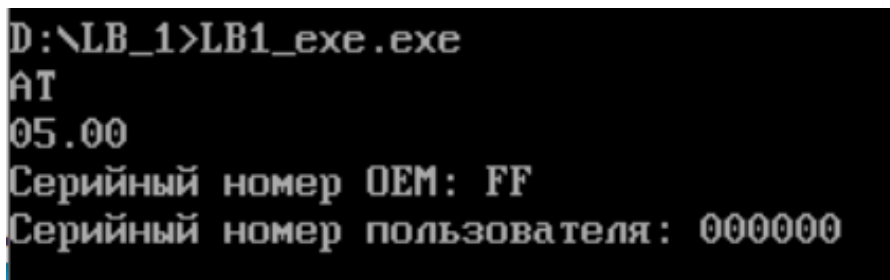
### **Ход работы и результаты исследования проблем.**

1. Написание исходного текста исходного .COM модуля на языке ассемблера, файл: *LB1\_COM.ASM* (см. в приложении).
2. Трансляция исходного кода командой: *masm lb1\_com.asm*, отладка и получение объектного модуля *LB1\_COM.OBJ*.
3. Сборка объектного модуля командой: *link lb1\_com.obj*, и получение загрузочного модуля *LB1\_COM.EXE* – плохого .EXE модуля.
4. Получение хорошего .COM модуля: *LB1\_COM.COM*, с помощью команды: *exe2bin lb1\_com.exe*.
5. Запуск программы в терминале ДОС: *lb1\_com.com:*



```
D:\LB_1>LB1_com.com
AT
05.00
Серийный номер OEM: FF
Серийный номер пользователя: 000000
```

6. Написание исходного текста исходного .EXE модуля на языке ассемблера, файл: *LB1\_EXE.ASM* (см. в приложении).
7. Трансляция исходного кода командой: *masm lbl\_exe.asm*, отладка и получение объектного модуля *LB1\_EXE.OBJ*.
8. Сборка объектного модуля командой: *link lbl\_exe.obj*, и получение загрузочного модуля *LB1\_EXE.EXE* – хорошего .EXE модуля.
9. Запуск программы в терминале ДОС: *lbl\_exe.exe*:



```
D:\LB_1>LB1_exe.exe
AT
05.00
Серийный номер OEM: FF
Серийный номер пользователя: 000000
```

10. Сравнение исходных текстов: *LB1\_COM.ASM* и *LB1\_EXE.ASM* (ответы на вопросы «Отличие исходных текстов .COM и .EXE программ):

- 1) COM-программа содержит всего один сегмент.

- 2) EXE-программа может содержать от одного до нескольких сегментов.

Например, в DOS программа содержит 3 сегмента: сегменты данных, кода и стека.

- 3) Директивы *ORG* и *ASSUME* должны обязательно присутствовать в тексте COM-программы. Команда *ORG 100h* помещает в регистр *IP* смещение 256 байт относительно начала *PSP*-сектора, находящегося в начале программы. Таким образом, в *IP* записывается адрес начала кода программы, так как размер *PSP* – 256 байт. Без этой директивы программа начнет исполнять код *PSP*. Директива же *ASSUME* указывает, с каким сегментом связан тот или иной сегментный регистр. Позволяет ассемблеру проверить допустимость ссылок. Без нее программа не скомпилируется.

- 4) В COM-программе можно использовать не все виды команд. Например, запрещается использовать команду *SEG*, так как в COM-программе отсутствует таблица настройки и данных о каких-либо иных сегментах, кроме единственного основного, нет.

11. Запуск FAR и исследование файла загрузочного модуля *LB1\_COM.COM* и плохого *LB1\_COM.EXE* в шестнадцатеричном виде. Их сравнение с загрузочным модулем *LB1\_EXE.EXE*:

Ответы на вопросы «Отличия форматов файлов COM и EXE модулей»:

1) Код (данные) располагаются с нулевого адреса 0h, так как код, данные и стек расположены в одном сегменте. (см. файл *LB1\_COM.COM*)

2) В файле плохого EXE код данные и стек расположены в одном сегменте. С нулевого адреса располагается заголовок и таблица настройки адресов. Код с данными начинается с адреса 300h. (см. файл *LB1\_COM.EXE*)

3) В хорошем EXE код данные и стек расположены в различных сегментах, в отличие от плохого, в определенном порядке расположения. (см. файл *LB1\_EXE.EXE*).

```
view LB1_COM.COM - Far 3.0.5700.0 x64
D:\Sasha\OS\DOS\LB_1\LB1_COM.COM
0000000000: E9 16 01 8D A5 AA AE E0 E0 A5 AA E2 AD EB A9 20 щ=Некорректный
0000000010: E4 AE E0 AC A0 E2 20 E2 A8 AF A0 20 50 43 3A 20 формат типа PC:
0000000020: A2 E1 E2 E0 A5 E2 A8 AB AE E1 EC 3A 20 00 00 0D встретилось:
0000000030: 0A 24 50 43 0D 0A 24 50 43 2F 58 54 0D 0A 24 41 PC/XT
0000000040: 54 0D 0A 24 50 43 32 20 AC AE A4 A5 AB EC 20 33 T PC2 модель 3
0000000050: 30 0D 0A 24 50 43 32 20 AC AE A4 A5 AB EC 20 38 PC2 модель 8
0000000060: 30 0D 0A 24 50 43 6A 72 0D 0A 24 50 43 20 43 6F PCjr PC Co
0000000070: 6E 76 65 72 74 69 62 6C 65 0D 0A 24 30 30 2E 30 nvertible.
0000000080: 30 0D 0A 24 91 A5 E0 A8 A9 AD EB A9 20 AD AE AC PC Серийный ном
0000000090: A5 E0 20 4F 45 4D 3A 20 00 00 0D 0A 91 A5 E0 A8 ер OEM:
00000000A0: A9 AD EB A9 20 AD AE AC A5 E0 20 AF AE AB EC A7 Серийный номер польз
00000000B0: AE A2 A0 E2 A5 AB EF 3A 20 00 00 00 00 00 00 0D ователя:
00000000C0: 0A 24 24 0F 3C 09 76 02 04 07 04 30 C3 51 8A E0 PC
00000000D0: E8 EF FF 86 C4 B1 04 D2 E8 E8 E6 FF 59 C3 51 52 шя Ж-тшщ Y QR
00000000E0: 32 E4 33 D2 B9 0A 00 F7 F1 80 CA 30 88 14 4E 33 2фЗт üA-иИЖNЗ
00000000F0: D2 3D 0A 00 73 F1 3C 00 74 04 0C 30 88 04 5A 59 T= së< tQИZY
0000000100: C3 53 8A FC E8 C6 FF 88 25 4F 88 05 4F 8A C7 E8 SKNш И%ИИ+OKш
0000000110: BB FF 88 25 4F 88 05 5B C3 B8 00 F0 8E C0 BB FE ИИ%ИИ+ И ЭО L
0000000120: FF 26 8A 07 3C FF 75 06 BA 32 01 EB 4C 90 3C FE &K< u 2ыLP<
0000000130: 75 06 BA 37 01 EB 42 90 3C FB 75 02 EB F4 3C FC u 7ыBP<v uыI<N
0000000140: 75 06 BA 3F 01 EB 32 90 3C FA 75 06 BA 44 01 EB u ?ы2P< u Dы
0000000150: 28 90 3C F8 75 06 BA 54 01 EB 1E 90 3C FD 75 06 (P<° u TыP<u
0000000160: BA 64 01 EB 14 90 3C F9 75 06 BA 6B 01 EB 0A 90 dыP< u kыP
0000000170: E8 5A FF A3 2D 01 BA 03 01 B4 09 CD 21 B4 30 CD шZ г-@||@-o=!-@=
0000000180: 21 8B D0 BE 7C 01 46 E8 54 FF 8A C6 BE 7F 01 46 !л|||@FшT K|@OF
0000000190: E8 4B FF BA 7C 01 B4 09 CD 21 8A C7 E8 2E FF A3 шK ||@-o=!Kш. г
00000001A0: 98 01 8A C3 E8 26 FF A3 B9 01 8B C1 BF B9 01 83 ШOKш& г|@л-||@Г
00000001B0: C7 05 E8 4C FF BA 84 01 B4 09 CD 21 32 C0 B4 4C ||шL ||д@-o=!2 L
00000001C0: CD 21 =!
```

Рис. 1. Файл *LB1\_COM.COM*

```

view LB1_COM.EXE - Far 3.0.5700.0 x64
D:\Sasha\OS\DOS\LB_1\LB1_COM.EXE
00000000: 4D 5A C2 00 03 00 00 00 20 00 00 00 FF FF 00 00 MZT ♥
00000010: 00 00 C5 F5 00 01 00 00 1E 00 00 00 01 00 00 00 +i @ ▲ @
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
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
1Help 2Text 3Quit 4Dump 5 6Edit 7Search

```

```

view LB1_COM.EXE - Far 3.0.5700.0 x64
D:\Sasha\OS\DOS\LB_1\LB1_COM.EXE
0000002C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000002D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000002E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000002F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000300: E9 16 01 8D A5 AA AE E0 E0 A5 AA E2 AD EB A9 20 щ=0Некорректный
000000310: E4 AE E0 AC A0 E2 20 E2 A8 AF A0 20 50 43 3A 20 формат типа PC:
000000320: A2 E1 E2 E0 A5 E2 A8 AB AE E1 EC 3A 20 00 00 0D встретилось:
000000330: 0A 24 50 43 0D 0A 24 50 43 2F 58 54 0D 0A 24 41 PCPC/XTA
000000340: 54 0D 0A 24 50 43 32 20 AC AE A4 A5 AB EC 20 33 TPC2 модель 3
000000350: 30 0D 0A 24 50 43 32 20 AC AE A4 A5 AB EC 20 38 PC2 модель 8
000000360: 30 0D 0A 24 50 43 6A 72 0D 0A 24 50 43 20 43 6F PCjrPC Co
000000370: 6E 76 65 72 74 69 62 6C 65 0D 0A 24 30 30 2E 30 nvertible$0.0
000000380: 30 0D 0A 24 91 A5 E0 A8 A9 AD EB A9 20 AD AE AC PCСерийный ном
000000390: A5 E0 20 4F 45 4D 3A 20 00 00 0D 0A 91 A5 E0 A8 ер OEM:
0000003A0: A9 AD EB A9 20 AD AE AC A5 E0 20 AF AE AB EC A7 иный номер польз
0000003B0: AE A2 A0 E2 A5 AB EF 3A 20 00 00 00 00 00 00 0D ователя:
0000003C0: 0A 24 24 0F 3C 09 76 02 04 07 04 30 C3 51 8A E0 $<ov♦♦0QКр
0000003D0: E8 EF FF 86 C4 B1 04 D2 E8 E8 E6 FF 59 C3 51 52 шя Ж-тшшц Y|QR
0000003E0: 32 E4 33 D2 B9 0A 00 F7 F1 80 CA 30 88 14 4E 33 2ф3т|üëA-0иN3
0000003F0: D2 3D 0A 00 73 F1 3C 00 74 04 0C 30 88 04 5A 59 т= së< t♦90иZY
000000400: C3 53 8A FC E8 C6 FF 88 25 4F 88 05 4F 8A C7 E8 SKнш- И%ОиOKш
000000410: BB FF 88 25 4F 88 05 5B C3 B8 00 F0 8E C0 BB FE и И%Ои+| 7EO-и
000000420: FF 26 8A 07 3C FF 75 06 BA 32 01 EB 4C 90 3C FE &K< u||2ыLP<
000000430: 75 06 BA 37 01 EB 42 90 3C FB 75 02 EB F4 3C FC u||7ыBP<√uыi<и
000000440: 75 06 BA 3F 01 EB 32 90 3C FA 75 06 BA 44 01 EB u||?ы2P<-u||Dыы
000000450: 28 90 3C F8 75 06 BA 54 01 EB 1E 90 3C FD 75 06 (P<°u||TыиP<иu
000000460: BA 64 01 EB 14 90 3C F9 75 06 BA 6B 01 EB 0A 90 ||dыыP<-u||kыыP
000000470: E8 5A FF A3 2D 01 BA 03 01 B4 09 CD 21 B4 30 CD шZ г-@||♥@|о=|!|о=
000000480: 21 8B D0 BE 7C 01 46 E8 54 FF 8A C6 BE 7F 01 46 !л||OFшT К||оF
000000490: E8 4B FF BA 7C 01 B4 09 CD 21 8A C7 E8 2E FF A3 шK |||@|о=|K|ш. г
0000004A0: 98 01 8A C3 E8 26 FF A3 B9 01 8B C1 BF B9 01 83 шOK|ш& г||@л-||@Г
0000004B0: C7 05 E8 4C FF BA 84 01 B4 09 CD 21 32 C0 B4 4C ||шL ||д@|о=|2||L
0000004C0: CD 21 !=
1Help 2Text 3Quit 4Dump 5 6Edit 7Search

```

Рис. 2. Файл LB1\_COM.EXE





12. Запуск отладчика TD.EXE и загрузка .COM модуля в основную память:

1) Модуль .COM имеет следующий формат:

- С нулевого адреса располагается специальный блок PSP (префикс программного сегмента), размер которого – 100h (256 байт).
- Далее с адреса 100h располагается код и данные в одном сегменте.
- Все сегментные регистры DS, CS, ES и SS указывают на нулевой адрес относительно начала памяти под программу, т. е. на PSP.
- Директива ORG 100h заносит в регистр IP значение 100h, чтобы точка входа в программу была в месте начала кода: CS:IP.
- Размер всей памяти, отведенный под программу – 64 кб, при этом стек заполняется «сверху вниз», начиная с адреса FFFFh (64 кб), относительно начала сегмента программы. Сегментный регистр стека имеет адрес 0h, а регистр SP указывает на верхушку стека в начале программы, т. е. имеет адрес FFFFh, относительно начала сегмента программы.

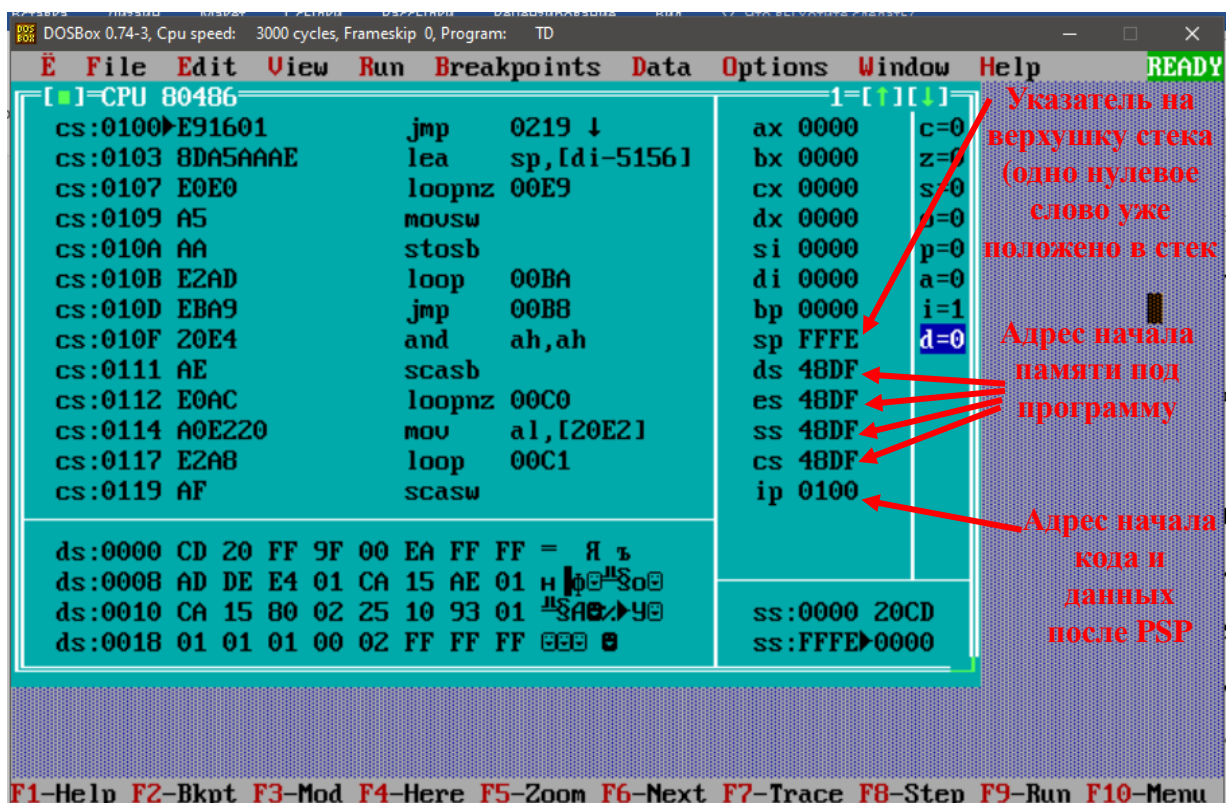


Рис. 4. Запуск программы .COM в отладчике TD



2) План загрузки модуля .COM в основную память:

- I.** Определяется сегментный адрес свободного участка памяти для загрузки программы (обычно MS-DOS загружает программу в младшие адреса памяти, если при редактировании не указана загрузка в старшие адреса);
- II.** Создаются два блока памяти - блок памяти для переменных среды, а также блок памяти для PSP и программы;
- III.** В блок памяти переменных среды помещается путь к файлу программы;
- IV.** Заполняются поля префикса сегмента программы PSP в соответствии с характеристиками программы (количество памяти, доступное программе, адрес сегмента блока памяти, содержащего переменные среды и т. д.);
- V.** Адрес области Disk Transfer Area (DTA) устанавливается на вторую половину PSP (PSP:0080);
- VI.** Анализируются параметры запуска программы на предмет наличия в первых двух параметрах идентификаторов дисковых устройств. По результатам анализа устанавливается содержимое регистра AX при входе в программу. Если первый или второй параметры не содержат правильного идентификатора дискового устройства, то соответственно в регистры AL и AH записывается значение FFh.
- VII.** Сегментные регистры CS, DS, ES, SS устанавливаются на начало PSP;
- VIII.** Регистр SP устанавливается на конец сегмента PSP;
- IX.** Вся область памяти после PSP распределяется программе;
- X.** В стек записывается слово 0000;
- XI.** Указатель команд IP устанавливается на 100h (начало программы).

13. Запуск отладчика TD.EXE и загрузка хорошего .EXE модуля в основную память:

1) Хороший .EXE после его загрузки в основную память имеет следующую структуру:

- С нулевого адреса начинается PSP-блок размером 256 байт.
- С адреса 100h начинается блок сегмента стека. Адрес начала этого блока записывается в сегментный регистр SS.
- За сегментом стека следует сегмент данных. Его адрес необходимо положить в регистр DS в процессе выполнения программы.
- За сегментом данных следует сегмент кода, адрес которого кладется в регистр CS.

Регистры DS и ES перед началом работы программы указывают на начало блока PSP.

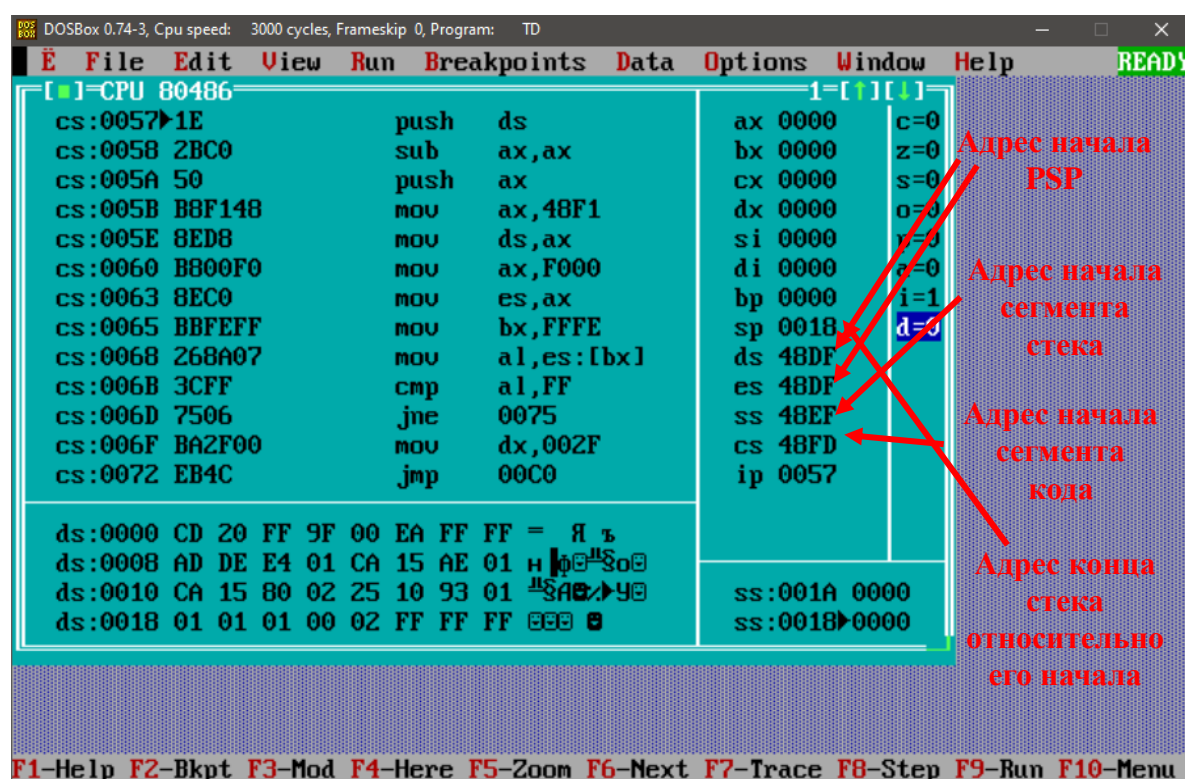


Рис. 5. Запуск программы .EXE в отладчике TD

2) Регистры DS и ES указывают на начало блока PSP.

3) Стек определяется следующим образом: регистр SS указывает на начало сегмента стека, а регистр SP – на конец сегмента стека относительно его начала.

4) Точка входа определяется названием метки в коде программы, стоящей после директивы END, с которой требуется начать выполнение программы. В данном случае такой меткой является начала главной процедуры Main.

#### **Вывод.**

В ходе лабораторной работы было проведено исследование различий в структурах исходных текстов модулей типов **.COM** и **.EXE**, структур файлов загрузочных модулей и способов их загрузки в основную память на примере собственной программы на языке Ассемблера, определяющей тип РС и версию системы.

# ПРИЛОЖЕНИЕ А

## ИСХОДНЫЙ КОД ПРОГРАММЫ

### 1. Файл LB1\_COM.ASM (исходный текст для .COM модуля):

```
LB_1 SEGMENT
    ASSUME CS:LB_1, DS:LB_1, ES:NOTHING, SS:NOTHING
    ORG 100H

START: JMP BEGIN

; Данные:
Message db 'Некорректный формат типа PC: встретилось: '
Number  dw 0
        db 0DH,0AH,'$'

FF db 'PC', 0DH,0AH,'$'
FE db 'PC/XT', 0DH,0AH,'$'
FC db 'AT', 0DH,0AH,'$'
FA db 'PC2 модель 30', 0DH,0AH,'$'
;FC db 'PC2 модель 50 или 60', 0DH,0AH,'$'
F8 db 'PC2 модель 80', 0DH,0AH,'$'
FD db 'PCjr', 0DH,0AH,'$'
F9 db 'PC Convertible', 0DH,0AH,'$'

Version_xx dw '00'
        db '.'
Version_yy dw '00'
        db 0DH,0AH,'$'

Serial_m db 'Серийный номер OEM: '
Serial_n dw 0
db 0DH,0AH, 'Серийный номер пользователя: '
Serial_user db 6 dup(0)
db 0DH,0AH, '$'

; Код (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
    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

```

```

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_1
    or AL,30h
    mov [SI],AL
end_1: pop DX
    pop CX
    ret
BYTE_TO_DEC 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

```

; Код (программа):

```

BEGIN:
        mov AX, 0F000h
        mov ES, AX
        mov BX, 0FFFEh
        mov AL, ES:[BX]
cmp_1:   cmp AL, 0FFh
        jne cmp_2
        mov dx, offset FF
        jmp end_case
cmp_2:   cmp AL, 0FEh
        jne cmp_3
dop:    mov dx, offset FE
        jmp end_case
cmp_3:   cmp AL, 0FBh
        jne cmp_4
        jmp dop
cmp_4:   cmp AL, 0FCh
        jne cmp_5
        mov dx, offset FC
        jmp end_case
cmp_5:   cmp AL, 0FAh
        jne cmp_6
        mov dx, offset FA
        jmp end_case
cmp_6:   cmp AL, 0F8h
        jne cmp_7
        mov dx, offset F8
        jmp end_case
cmp_7:   cmp AL, 0FDh
        jne cmp_8
        mov dx, offset FD
        jmp end_case
cmp_8:   cmp AL, 0F9h
        jne default
        mov dx, offset F9

```



```

        jmp end_case

default:

        call BYTE_TO_HEX
        mov number, AX
        mov dx, offset Message

end_case:

        mov AH,09h
        int 21h


        mov AH, 30h
        int 21h
        mov DX, AX
        mov SI, offset Version_xx
        inc SI
        call BYTE_TO_DEC
        mov AL, DH
        mov SI, offset Version_yy
        inc SI
        call BYTE_TO_DEC


        mov DX, offset Version_xx
        mov AH,09h
        int 21h


        mov AL, BH
        call BYTE_TO_HEX
        mov Serial_n, AX


        mov AL, BL
        call BYTE_TO_HEX
        mov Serial_user, AX
        mov AX, CX
        mov DI, offset Serial_user
        add DI, 5
        call WRD_TO_HEX


        mov DX, offset Serial_m
        mov AH,09h
        int 21h


        xor AL,AL
        mov AH,4Ch
        int 21H

```

LB\_1 ENDS

END START

## 2. Файл *LBI\_EXE.ASM* (исходный текст для **.EXE** модуля):

```
AStack    SEGMENT    STACK

            DW 12 DUP(?)    ; Отводится 12 слов памяти

AStack    ENDS

; Данные программы

DATA      SEGMENT

    Message db 'Некорректный формат типа PC: встретилось: '
    Number  dw 0
            db 0DH,0AH,'$'

    FF db 'PC', 0DH,0AH,'$'
    FE db 'PC/XT', 0DH,0AH,'$'
    FC db 'AT', 0DH,0AH,'$'
    FA db 'PC2 модель 30', 0DH,0AH,'$'
    ;FC db 'PC2 модель 50 или 60', 0DH,0AH,'$'
    F8 db 'PC2 модель 80', 0DH,0AH,'$'
    FD db 'PCjr', 0DH,0AH,'$'
    F9 db 'PC Convertible', 0DH,0AH,'$'

    Version_xx dw '00'
            db '.'
    Version_yy dw '00'
            db 0DH,0AH,'$'

    Serial_m db 'Серийный номер OEM: '
    Serial_n dw 0
            db 0DH,0AH, 'Серийный номер пользователя: '
    Serial_user db 6 dup(0)
            db 0DH,0AH, '$'

DATA      ENDS

; Код программы

CODE      SEGMENT

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

    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

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_1
        or AL,30h
        mov [SI],AL
end_1:  pop DX
        pop CX
        ret
BYTE_TO_DEC 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

Main      PROC  FAR
            push  DS
            sub   AX,AX
            push  AX
            mov   AX, DATA
            mov   DS, AX

            mov  AX, 0F000h
            mov  ES, AX
            mov  BX, 0FFFEh
            mov  AL, ES:[BX]
cmp_1:      cmp  AL, 0FFh
            jne  cmp_2
            mov  dx, offset FF
            jmp  end_case
cmp_2:      cmp  AL, 0FEh
            jne  cmp_3
dop:        mov  dx, offset FE
            jmp  end_case
cmp_3:      cmp  AL, 0FBh
            jne  cmp_4
            jmp  dop
cmp_4:      cmp  AL, 0FCh
            jne  cmp_5
            mov  dx, offset FC
            jmp  end_case
cmp_5:      cmp  AL, 0FAh
            jne  cmp_6

```

```

        mov dx, offset FA
        jmp end_case
cmp_6:   cmp AL, 0F8h
        jne cmp_7
        mov dx, offset F8
        jmp end_case
cmp_7:   cmp AL, 0FDh
        jne cmp_8
        mov dx, offset FD
        jmp end_case
cmp_8:   cmp AL, 0F9h
        jne default
        mov dx, offset F9
        jmp end_case
default:
        call BYTE_TO_HEX
        mov number, AX
        mov dx, offset Message
end_case:
        mov AH, 09h
        int 21h

        mov AH, 30h
        int 21h
        mov DX, AX
        mov SI, offset Version_xx
        inc SI
        call BYTE_TO_DEC
        mov AL, DH
        mov SI, offset Version_yy
        inc SI
        call BYTE_TO_DEC

        mov DX, offset Version_xx
        mov AH, 09h
        int 21h

        mov AL, BH
        call BYTE_TO_HEX
        mov Serial_n, AX

        mov AL, BL
        call BYTE_TO_HEX
        mov Serial_user, AX

```

```

        mov AX, CX
        mov DI, offset Serial_user
        add DI, 5
        call WRD_TO_HEX

        mov DX, offset Serial_m
        mov AH, 09h
        int 21h

        ret
Main      ENDP
CODE      ENDS
        END Main

```