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: 14.0	2.2021
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 PC 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 ах: 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 PC. 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 PC в строковом виде.
- 3. Далее в регистр АН 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

АН - номер модификации

BH - серийный номер OEM (Original Equipment Manufacturer)

BL:CX - 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 AX.
- 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 (рис. 1):

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

Рис. 1. Запуск программы <u>lb1 сот.сот</u> в терминале ДОС

- 6. Написание исходного текста исходного .EXE модуля на языке ассемблера, файл: *LB1 EXE.ASM* (см. в приложении).
- 7. Трансляция исходного кода командой: *masm lb1_exe.asm*, отладка и получение объектного модуля *LB1 EXE.OBJ*.
- 8. Сборка объектного модуля командой: $link\ lb1_exe.obj$, и получение загрузочного модуля $LB1\ EXE.EXE$ хорошего .EXE модуля.
- 9. Запуск программы в терминале ДОС (рис. 2):

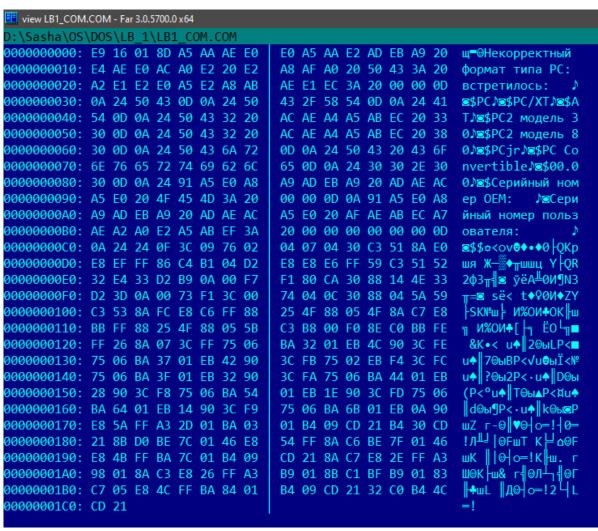
```
D:\LB_1>LB1_exe.exe
AT
05.00
Серийный номер ОЕМ: FF
Серийный номер пользователя: 000000
```

Рис. 2. Запуск программы <u>lb1_exe.exe</u> в терминале ДОС

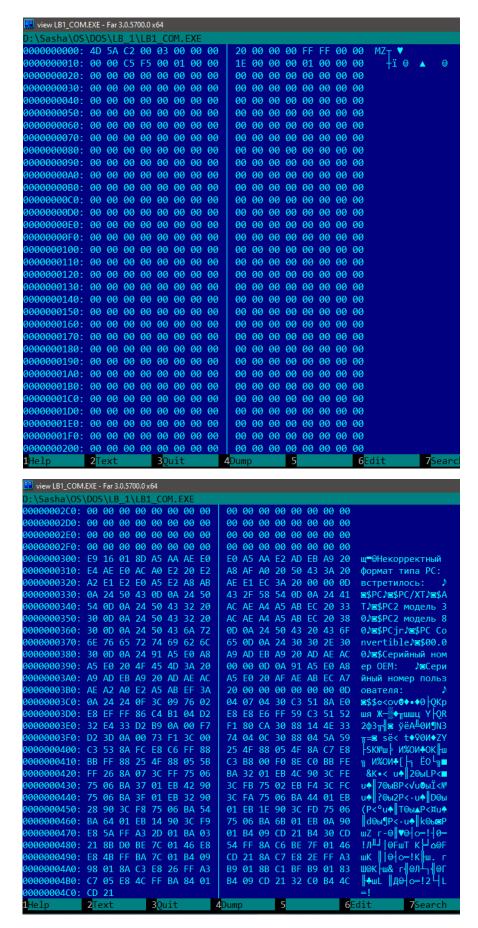
- 10. Сравнение исходных текстов: *LB1_COM.ASM* и *LB1_EXE.ASM* (ответы на вопросы «Отличие исходных текстов .COM и .EXE программ»):
- 1) Сколько сегментов должна содержать СОМ-программа? СОМ-программа содержит всего один сегмент.
- 2) <u>Сколько сегментов должна содержать EXE-программа?</u> *EXE-программа* может содержать от одного до нескольких сегментов. Например, в DOS программа содержит 3 сегмента: сегменты данных, кода и стека.

- 3) Какие директивы должны обязательно быть в тексте COM-программы? 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) Все ли форматы команд можно использовать в СОМ-программе? В СОМ-программе можно использовать не все виды команд. Например, запрещается использовать команду SEG, так как в СОМ-программе отсутствует таблица настройки и данных о каких-либо иных сегментов, 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) Какова структура файла СОМ? С какого адреса располагается код? Код (данные) располагаются с нулевого адреса 0h, так как код, данные и стек расположены в одном сегменте. Структура файла СОМ отображена на рисунке 3 (см. файл LB1 COM.COM, рис. 3).
- 2) Какова структура файла «плохого» EXE? С какого адреса располагается код? Что располагается с адреса 0? 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. Структура файла «плохого» EXE отображена на рисунке 4 (см. файл LB1_COM.EXE, рис. 4).

3) Какова структура файла «хорошего» ЕХЕ? Чем он отличается от «плохого» ЕХЕ? В хорошем ЕХЕ код данные и стек расположены в различных сегментах, в отличие от плохого, где они находятся в одном сегменте в определенном порядке расположения. Также хороший ЕХЕ модуль в отличие от плохого не содержит директивы ORG 100h, которая выделяет память под PSP-сегмент, размером в 256 байт, поэтому код начинается с адреса 220h. В плохом же ЕХЕ есть такая директива и код начинается ровно после выделенной памяти под PSP, из чего следует, что он начинается с адреса 300h. Структура файла «хорошего» ЕХЕ отображена на рисунке 5 (см. файл LB1_EXE.EXE, рис. 5).



Puc. 3. Файл LB1_COM.COM



Puc. 4. Файл LB1_COM.EXE

```
M7000 0
000000000: 4D 5A E4 01 02 00 01 00
                                      20 00 00 00 FF FF 00 00
0000000010: 18 00 F4 CB 57 00 0E 00
                                      1E 00 00 00 01 00 5C 00
                                                               ↑ Ï<sub>TT</sub>W ♬ ▲
000000020: 0E 00 00 00 00 00 00 00
                                      00 00 00 00 00 00 00 1
000000030: 00 00 00 00 00 00 00 00
                                     00 00 00 00 00 00 00 00
0000000040: 00 00 00 00 00 00 00 00
                                     00 00 00 00 00 00 00 00
000000050: 00 00 00 00 00 00 00 00
                                      00 00 00 00 00 00 00
000000060: 00 00 00 00 00 00 00 00
                                     00 00 00 00 00 00 00 00
0000000070: 00 00 00 00 00 00 00 00
                                      00 00 00 00 00 00 00
0000000080: 00 00 00 00 00 00 00
                                     00 00 00 00 00 00 00 00
000000090: 00 00 00 00 00 00 00 00
                                     00 00 00 00 00 00 00 00
0000000A0: 00 00 00 00 00 00 00
                                     00 00 00 00 00 00 00 00
00000000B0: 00 00 00 00 00 00 00
                                     00 00 00 00 00 00 00 00
0000000C0: 00 00 00 00 00 00 00 00
                                      00 00 00 00 00 00 00 00
0000000D0: 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
0000000F0: 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
0000000130: 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
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
0000001A0: 00 00 00 00 00 00 00 00
                                     00 00 00 00 00 00 00 00
0000001B0: 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
0000001D0: 00 00 00 00 00 00 00 00
                                      00 00 00 00 00 00 00 00
0000001E0: 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
000000200: 00 00 00 00 00 00 00 00
                                     00 00 00 00 00 00 00 00
                                               5
           2Dump
                       3Quit
                                                            6Edit
                                                                       7Search
 view LB1_EXE.EXE - Far 3.0.5700.0 x64
     ha\OS\DOS\LB_1\LB1_EXE.EXE
0000001E0: 00 00 00 00 00 00 00 00
                                     00 00 00 00 00 00 00 00
0000001F0: 00 00 00 00 00 00 00 00
                                     00 00 00 00 00 00 00 00
000000200: 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: 8D A5 AA AE E0 E0 A5 AA
                                     E2 AD EB A9 20 E4 AE E0
                                                              Некорректный фор
000000230: AC A0 E2 20 E2 A8 AF A0
                                     20 50 43 3A 20 A2 E1 E2
                                                              мат типа РС: вст
000000240: E0 A5 E2 A8 AB AE E1 EC
                                     3A 20 00 00 0D 0A 24 50
                                                              ретилось: №$Р
                                                              CJE$PC/XTJE$ATJE
0000000250: 43 0D 0A 24 50 43 2F 58
                                     54 0D 0A 24 41 54 0D 0A
0000000260: 24 50 43 32 20 AC AE A4
                                     A5 AB EC 20 33 30 0D 0A
                                                              $РС2 модель 30 №
0000000270: 24 50 43 32 20 AC AE A4
                                     A5 AB EC 20 38 30 0D 0A
                                                              $РС2 модель 80 №
000000280: 24 50 43 6A 72 0D 0A 24
                                     50 43 20 43 6F 6E 76 65
                                                              $PCjr⊅≥$PC Conve
000000290: 72 74 69 62 6C 65 0D 0A
                                     24 30 30 2F 30 30 0D 0A
                                                              rtible №$00.00 №
00000002A0: 24 91 A5 E0 A8 A9 AD EB
                                     A9 20 AD AE AC A5 E0 20
                                                              $Серийный номер
0000002B0: 4F 45 4D 3A 20 00 00 0D
                                     0A 91 A5 E0 A8 A9 AD EB
                                                              ОЕМ: №Серийны
0000002C0: A9 20 AD AE AC A5 E0 20
                                     AF AE AB EC A7 AE A2 A0
                                                              й номер пользова
0000002D0: E2 A5 AB EF 3A 20 00 00
                                     00 00 00 00 0D 0A 24 00
                                                              теля:
                                                                         .N⊠$
                                                              $≎<0∨€♦•♦0 -QКршя
00000002E0: 24 0F 3C 09 76 02 04 07
                                     04 30 C3 51 8A E0 E8 EF
                                                              00000002F0: FF 86 C4 B1 04 D2 E8 E8
                                     E6 FF 59 C3 51 52 32 E4
000000300: 33 D2 B9 0A 00 F7 F1 80
                                     CA 30 88 14 4E 33 D2 3D
                                                             0C 30 88 04 5A 59 C3 53
000000310: 0A 00 73 F1 3C 00 74 04
000000320: 8A FC E8 C6 FF 88 25 4F
                                     88 05 4F 8A C7 E8 BB FF
0000000330: 88 25 4F 88 05 5B C3 1E
                                     2B CØ 50 B8 Ø2 ØØ 8E D8
                                                             7 EO 1 ■ &K • < u • 4 → BP <
Vu@ui! < Nºu • 4 → BP <
Vu@ui! < Nºu • 4 → BP <
0000000340: B8 00 F0 8E C0 BB FE FF
                                     26 8A 07 3C FF 75 06 BA
000000350: 2F 00 EB 4C 90 3C FE 75
                                     06 BA 34 00 EB 42 90 3C
000000360: FB 75 02 EB F4 3C FC 75
                                     06 BA 3C 00 EB 32 90 3C
                                                              -u^||A ы(P<°u^||Q
000000370: FA 75 06 BA 41 00 EB 28
                                     90 3C F8 75 06 BA 51 00
0000000380: EB 1E 90 3C FD 75 06 BA
                                     61 00 EB 14 90 3C F9 75
                                                              ы≜Р<¤и∳∥а ы¶Р<∙и
                                                              000000390: 06 BA 68 00 EB 0A 90 E8
                                     51 FF A3 2A 00 BA 00 00
0000003A0: B4 09 CD 21 B4 30 CD 21
                                     8B DØ BE 79 ØØ 46 E8 4B
0000003B0: FF 8A C6 BE 7C 00 46 E8
                                     42 FF BA 79 00 B4 09 CD
0000003C0: 21 8A C7 E8 25 FF A3 95
                                     00 8A C3 E8 1D FF A3 B6
0000003D0: 00 8B C1 BF B6 00 83 C7
                                     05 E8 43 FF BA 81 00 B4
0000003E0: 09 CD 21 CB
                                                         6Edit
         2Dump
                                4Text 5
                                                                     7Search
```

view LB1_EXE.EXE - Far 3.0.5700.0 x64

Рис. 5. Файл LB1 EXE.EXE

- 12. Запуск отладчика TD.EXE и загрузка .COM модуля в основную память. Ответы на вопросы «Загрузка COM-модуля в основную память»:
 - 1) Какой формат загрузки модуля СОМ? С какого адреса располагается код? Модуль .COM имеет следующий формат (рис. 6):
 - 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.

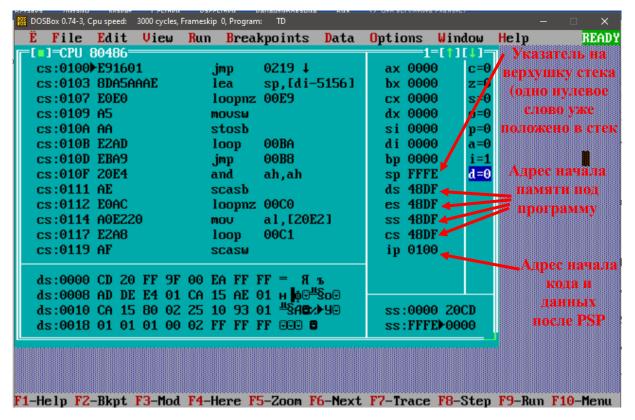


Рис. 6. Запуск программы .COM в отладчике TD

- 2) <u>Что располагается с адреса 0?</u> С нулевого адреса располагается специальный блок PSP (префикс программного сегмента), размер которого 100h (256 байт).
- 3) <u>Какие значения имеют сегментные регистры? На какие области памяти</u> они указывают? Все сегментные регистры DS, CS, ES и SS указывают на нулевой адрес относительно начала памяти под программу, т. е. на PSP (см. рис. 6).
- 4) <u>Как определяется стек? Какую область памяти он занимает? Какие адреса?</u> 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.

План загрузки модуля .СОМ 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 АХ 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;
 - Х. В стек записывается слово 0000;
 - **ХІ.** Указатель команд IP устанавливается на 100h (начало программы).

- 13. Запуск отладчика TD.EXE и загрузка хорошего .EXE модуля в основную память. Ответы на вопросы «Загрузка хорошего EXE модуля в основную память:
 - 1) <u>Как загружается «хороший» ЕХЕ? Какие значения имеют сегментные</u> регистры? *Хороший .EXE после его загрузки в основную память имеет следующую структуру (рис. 7):*
 - 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.

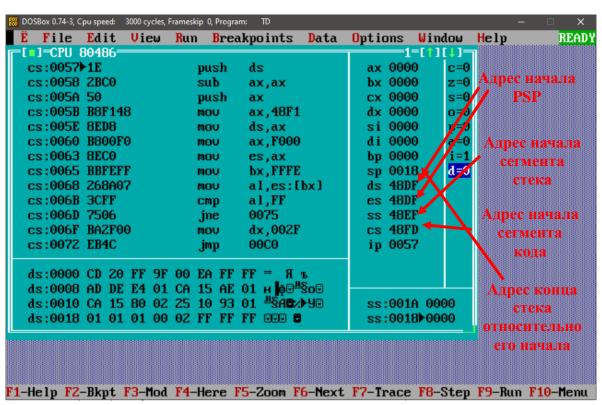


Рис. 7. Запуск программы .EXE в отладчике TD

- 2) <u>На что указывают регистры DS и ES?</u> Регистры DS и ES указывают на начало блока PSP.
- 3) <u>Как определяется стек?</u> Стек определяется следующим образом: регистр SS указывает на начало сегмента стека, а регистр SP на конец сегмента стека относительно его начала.
- 4) <u>Как определяется точка входа?</u> 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 РС: встретилось: '
Number dw 0
            db 0DH, 0AH, '$'
FF db 'PC', ODH, OAH, '$'
FE db 'PC/XT', ODH, OAH, '$'
FC db 'AT', ODH, OAH, '$'
FA db 'PC2 модель 30', 0DH, 0AH, '$'
;FC db 'PC2 модель 50 или 60', ODH, OAH, '$'
F8 db 'PC2 модель 80', 0DH,0AH,'$'
FD db 'PCjr', ODH, OAH, '$'
F9 db 'PC Convertible', ODH, OAH, '$'
Version xx dw '00'
               db '.'
Version yy dw '00'
               db 0DH, 0AH, '$'
Serial m db 'Серийный номер ОЕМ: '
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, OFh
            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_l
     or AL,30h
     mov [SI],AL
end_l: 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, OF000h
                  mov ES, AX
                  mov BX, OFFFEh
                  mov AL, ES:[BX]
                  cmp AL, OFFh
      cmp_1:
                  jne cmp_2
                  mov dx, offset FF
                  jmp end_case
                  cmp AL, OFEh
      cmp 2:
                  jne cmp 3
      dop: mov dx, offset FE
                  jmp end case
                  cmp AL, OFBh
      cmp 3:
                  jne cmp_4
                  jmp dop
                  cmp AL, OFCh
      cmp_4:
                  jne cmp_5
                  mov dx, offset FC
                  jmp end case
      cmp_5:
                  cmp AL, OFAh
                  jne cmp 6
                  mov dx, offset FA
                  jmp end_case
                  cmp AL, 0F8h
      cmp_6:
                  jne cmp 7
```

mov dx, offset F8

jmp end_case

cmp_7: cmp AL, OFDh

jne cmp_8

mov dx, offset FD

jmp end_case

cmp_8: cmp AL, OF9h

jne default

mov dx, offset F9

jmp end case

default:

call BYTE_TO_HEX

mov DI, offset number

mov word ptr DS:[DI], 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 DI, offset Serial n

mov word ptr DS:[DI], AX

mov AL, BL

call BYTE TO HEX

```
mov DI, offset Serial user
                  mov word ptr DS:[DI], 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
2. Файл LB1 EXE.ASM (исходный текст для .EXE модуля):
          SEGMENT STACK
          DW 12 DUP(?) ; Отводится 12 слов памяти
; Данные программы
         SEGMENT
      Message db 'Некорректный формат типа РС: встретилось: '
      Number dw 0
           db 0DH,0AH,'$'
      FF db 'PC', ODH, OAH, '$'
      FE db 'PC/XT', ODH, OAH, '$'
      FC db 'AT', ODH, OAH, '$'
      FA db 'PC2 модель 30', 0DH, 0AH, '$'
      ;FC db 'PC2 модель 50 или 60', ODH, OAH, '$'
      F8 db 'PC2 модель 80', 0DH,0AH,'$'
      FD db 'PCjr', ODH, OAH, '$'
      F9 db 'PC Convertible', ODH, OAH, '$'
      Version xx dw '00'
```

LB_1 ENDS END START

AStack

AStack

DATA

ENDS

db '.'

```
Version yy dw '00'
              db 0DH, 0AH, '$'
     Serial m db 'Серийный номер ОЕМ: '
      Serial n dw 0
      db 0DH,0AH, 'Серийный номер пользователя: '
      Serial user db 6 dup(0)
      db 0DH,0AH, '$'
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_l
     or AL,30h
     mov [SI],AL
end_l: 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, OF000h mov ES, AX mov BX, OFFFEh mov AL, ES:[BX] cmp AL, OFFh cmp 1: jne cmp 2 mov dx, offset FF jmp end case cmp 2: cmp AL, OFEh jne cmp 3 dop: mov dx, offset FE jmp end_case $cmp_3:$ cmp AL, OFBh jne cmp_4 jmp dop cmp AL, OFCh $cmp_4:$ jne cmp 5 mov dx, offset FC jmp end_case cmp AL, OFAh $cmp_5:$ jne cmp 6 mov dx, offset FA jmp end case cmp AL, 0F8h cmp 6: jne cmp_7 mov dx, offset F8 jmp end_case $cmp_7:$ cmp AL, OFDh jne cmp_8 mov dx, offset FD jmp end_case cmp AL, OF9h cmp_8: jne default mov dx, offset F9 jmp end case default: call BYTE_TO_HEX mov DI, offset number mov word ptr DS:[DI], 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 DI, offset Serial n

mov word ptr DS:[DI], AX

mov AL, BL

call BYTE_TO_HEX

mov DI, offset Serial_user

mov word ptr DS:[DI], 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