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 ЗАГОЛОВОЧНЫХ МОДУЛЕЙ

Студент гр. 9381 _____ Любимов В.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Цель работы

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

Описание функций и структуры данных

1. TETR_TO_HEX – переводит число, представляемое четырьмя младшими битами в регистре AL, в 16-ричную цифру-символ.
2. BYTE_TO_HEX - переводит число, содержащиеся в регистре AL, в 16-ричные цифры, записывающиеся в регистры AL и AH.
3. WRD_TO_HEX – переводит слово в регистре AX в четыре 16-ричные цифры, записывающие по адресу, находящемуся в DI.
4. BYTE_TO_DEC - переводит число, содержащиеся в регистре AL, в 10-ричные цифры, записывающиеся по адресу, находящемуся в SI.
5. PRINT_MES – при помощи функции 9h из прерывания 21h выводит строку на экран.
6. DEFINE_PC_TYPE - выводит на экран информацию о типе PC. Для этого считывает значение предпоследнего байта ROM BIOS в AL. Сравнивает полученное код со значениями соответствующих типов PC. В случае успеха выводит на экран строку содержащую тип PC. Если полученный код не совпал не с одним из значений, соответствующих какому-либо типу PC, то при помощи функции BYTE_TO_HEX переводит полученный код в 16-ричное число и выводит его на экран.
7. DEFINE_OS_OEM_USER – выводит номер версии операционной системы серийный OEM номер и серийный номер пользователя. Для этого используется функция 30h прерывания 21h. Полученные в результате данные переводится в символьный вид при помощи функций BYTE_TO_DEC, BYTE_TO_HEX, WRD_TO_HEX в составе строк с пояснениями.

Ход выполнения работы

1. Был написан текст исходного .COM модуля lb1_com.asm, который определяет тип PC и версию системы. Затем были получены «плохой» .EXE модуль (последовательность команд: masm lb1_com.asm → link lb1_com.OBJ → lb1_com.exe) и «хороший» .COM модуль (последовательность команд: masm lb1_com.asm → link lb1_com.OBJ → exe2bin.exe lb1_com.exe lb1.com → lb1.com).

2. Был написан текст программы для «хорошего» .EXE модуля, после чего он был получен при помощи следующей последовательности команд: masm lb1_exe.asm → link lb1_exe.OBJ → lb1_exe.exe.

3. Структуры полученных загрузочных модулей были сравнены в 16-ричном виде при помощи файлового менеджера FAR.

4. Структуры полученных загрузочных модулей также были сравнены при помощи отладчика TD.EXE.

Ответы на контрольные вопросы

Отличия исходных текстов COM и EXE программ

1. Сколько сегментов должна содержать COM-программа?

COM-программа состоит из единственного сегмента, содержащего и данные и код.

2. EXE-программа?

EXE-программа может состоять из нескольких сегментов. Также данные, стек и код находятся в различных сегментах.

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

Начало сегмента определяется директивой SEGMENT.

Так как при загрузке COM-программы в память первые 256 байт (100h) выделяются под блок данных PSP, а затем располагает сам код программы. Для этого в коде программы необходимо использовать директиву ORG 100h.

Так как COM-программа состоит только из одного сегмента, то сегмент кода и стека объединяются в один при помощи директивы ASSUME.

Для завершения программы используется директива END.

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

Из-за того, что COM-программа состоит только из одного сегмента, использование far-переходов невозможно, а так как нет таблицы настроек, то нельзя использовать оператор SEG.

Отличия форматов файлов COM и EXE модулей

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

Внутри одного сегмента располагаются и код, и данные. Код начинается с адреса 0h.

```
C:\MASM\LB1.COM
0000000000: E9 F5 01 50 43 20 74 79  70 65 20 69 73 20 24 50  йх@PC type is $P
0000000010: 43 0D 0A 24 50 43 2F 58  54 0D 0A 24 41 54 0D 0A  C♪◙$PC/XT♪◙$AT♪◙
0000000020: 24 50 53 32 20 6D 6F 64  65 6C 20 33 30 0D 0A 24  $PS2 model 30♪◙$
0000000030: 50 53 32 20 6D 6F 64 65  6C 20 35 30 20 6F 72 20  PS2 model 50 or
0000000040: 36 30 0D 0A 24 50 53 32  20 6D 6F 64 65 6C 20 38  60♪◙$PS2 model 8
0000000050: 30 0D 0A 24 50 43 6A 72  0D 0A 24 50 43 20 43 6F  0♪◙$PCjr♪◙$PC Co
0000000060: 6E 76 65 72 74 69 62 6C  65 0D 0A 24 20 75 6E 6B  nvertible♪◙$ unk
0000000070: 6E 6F 77 6E 20 50 43 20  74 79 70 65 0D 0A 24 53  nown PC type♪◙$S
0000000080: 79 73 74 65 6D 20 76 65  72 73 69 6F 6E 20 69 73  ystem version is
0000000090: 20 20 2E 20 0D 0A 24 4F  45 4D 20 6E 75 6D 62 65    . ♪◙$OEM numbe
00000000A0: 72 20 69 73 20 20 20 0D  0A 24 55 73 65 72 20 6E  r is   ♪◙$User n
00000000B0: 75 6D 62 65 72 20 69 73  20 20 20 20 20 20 20 0D  umber is       ♪
00000000C0: 0A 24 24 0F 3C 09 76 02  04 07 04 30 C3 51 8A E0  ◙$$☼<☺v☻♦•♦0ГQЉa
00000000D0: E8 EF FF 86 C4 B1 04 D2  E8 E8 E6 FF 59 C3 53 8A  ипя†Д±♦ТиижяYГSЉ
00000000E0: FC E8 E9 FF 88 25 4F 88  05 4F 8A C7 E8 DE FF 88  ьийя€%O€♦OЉЗиЮя€
00000000F0: 25 4F 88 05 5B C3 51 52  32 E4 33 D2 B9 0A 00 F7  %O€♦[ГQR2д3ТЎ◙ ч
0000000100: F1 80 CA 30 88 14 4E 33  D2 3D 0A 00 73 F1 3C 00  сЂК0€¶N3Т=◙ sc<
0000000110: 74 04 0C 30 88 04 5A 59  C3 50 B4 09 CD 21 58 C3  t♦♀0€♦ZYГPґ☼H!ХГ
0000000120: 50 52 06 BA 03 01 E8 F0  FF B8 00 F0 8E C0 26 A0  PR♠є♥@иряё рЋА&
0000000130: FE FF 3C FF 74 31 3C FE  74 33 3C FB 74 2F 3C FC  юя<ят1<ют3<ыt/<ь
0000000140: 74 31 3C FA 74 33 3C FC  74 35 3C F8 74 37 3C FD  t1<ъt3<ьt5<шt7<э
0000000150: 74 39 3C F9 74 3B E8 74  FF BF 6C 01 88 05 88 65  t9<щt;иtяїl@€♦€e
0000000160: 01 BA 6C 01 EB 31 90 BA  0F 01 EB 2B 90 BA 14 01  ☺єl☺л1ђєо☺л+ђє¶☺
0000000170: EB 25 90 BA 1C 01 EB 1F  90 BA 21 01 EB 19 90 BA  л%ђєL☺л▼ђє!☺л↓ђє
0000000180: 30 01 EB 13 90 BA 45 01  EB 0D 90 BA 54 01 EB 07  0☺л!!ђєE☺л♪ђєT☺л•
0000000190: 90 BA 5B 01 EB 01 90 E8  7F FF 07 5A 58 C3 50 52  ђє[☺л☺ђиояя•ZXГPR
00000001A0: 56 57 53 BF 7F 01 B4 30  CD 21 8A F4 E8 47 FF AD  VWSїL☺ґ0H!ЉфиGя-
00000001B0: 88 65 12 8A C6 E8 3E FF  AD 88 65 14 BA 7F 01 E8  €e‡ЉЖи>я-€e¶єо☺и
00000001C0: 57 FF 8A C7 E8 06 FF BF  97 01 88 45 0F 88 65 10  WяЊЗи♠яї—@€Eo€e►
00000001D0: BA 97 01 E8 43 FF BF AA  01 8A C3 E8 EF FE 88 45  є—@иСяїЄ@ЊГипю€E
00000001E0: 0F 88 65 10 8B C1 83 C7  14 E8 F2 FE BA AA 01 E8  o€e►‹БѓЗ¶итюєЄ@и
00000001F0: 27 FF 5B 5F 5E 5A 58 C3  E8 25 FF E8 A0 FF 32 C0  'я[_^ZXГи%яи я2А
0000000200: B4 4C CD 21                                      ґLH!
```

Рисунок 1. COM файл в 16-ричном виде.

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

Данные и код расположены также в одном сегменте, но начиная с адреса 300h. Начиная с адреса 0h, расположена таблица настроек.

```
C:\MASM\LB1_COM.EXE
0000000000: 4D 5A 04 01 03 00 00 00  20 00 00 00 FF FF 00 00
0000000010: 00 00 53 58 00 01 00 00  1E 00 00 00 01 00 00 00
0000000020: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0000000030: 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
0000000050: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0000000060: 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 00
0000000080: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0000000090: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000000A0: 00 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 00
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: 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 F5 01 50 43 20 74 79  70 65 20 69 73 20 24 50  йх@PC type is $P
0000000310: 43 0D 0A 24 50 43 2F 58  54 0D 0A 24 41 54 0D 0A  C♪◙$PC/XT♪◙$AT♪◙
0000000320: 24 50 53 32 20 6D 6F 64  65 6C 20 33 30 0D 0A 24  $PS2 model 30♪◙$
0000000330: 50 53 32 20 6D 6F 64 65  6C 20 35 30 20 6F 72 20  PS2 model 50 or
0000000340: 36 30 0D 0A 24 50 53 32  20 6D 6F 64 65 6C 20 38  60♪◙$PS2 model 8
0000000350: 30 0D 0A 24 50 43 6A 72  0D 0A 24 50 43 20 43 6F  0♪◙$PCjr♪◙$PC Co
0000000360: 6E 76 65 72 74 69 62 6C  65 0D 0A 24 20 75 6E 6B  nvertible♪◙$ unk
0000000370: 6E 6F 77 6E 20 50 43 20  74 79 70 65 0D 0A 24 53  nown PC type♪◙$S
0000000380: 79 73 74 65 6D 20 76 65  72 73 69 6F 6E 20 69 73  ystem version is
0000000390: 20 20 2E 20 0D 0A 24 4F  45 4D 20 6E 75 6D 62 65    . ♪◙$OEM numbe
00000003A0: 72 20 69 73 20 20 20 0D  0A 24 55 73 65 72 20 6E  r is   ♪◙$User n
00000003B0: 75 6D 62 65 72 20 69 73  20 20 20 20 20 20 20 0D  umber is       ♪
00000003C0: 0A 24 24 0F 3C 09 76 02  04 07 04 30 C3 51 8A E0  ◙$$☼<○v☻♦•♦0ГQЉa
00000003D0: E8 EF FF 86 C4 B1 04 D2  E8 E8 E6 FF 59 C3 53 8A  ипя†Дұ♦ТиижяУГSЉ
00000003E0: FC E8 E9 FF 88 25 4F 88  05 4F 8A C7 E8 DE FF 88  ьийя€%О€♦ОЉЗиЮя€
00000003F0: 25 4F 88 05 5B C3 51 52  32 E4 33 D2 B9 0A 00 F7  %О€♦[ГQR2д3ТN№◙ ч
0000000400: F1 80 CA 30 88 14 4E 33  D2 3D 0A 00 73 F1 3C 00  сЂК0€¶N3Т=◙ sc<
0000000410: 74 04 0C 30 88 04 5A 59  C3 50 B4 09 CD 21 58 C3  t♦♀0€♦ZYГPґ▫ОН!ХГ
0000000420: 50 52 06 BA 03 01 E8 F0  FF B8 00 F0 8E C0 26 A0  PR♠є♥0иряё рЋА&
0000000430: FE FF 3C FF 74 31 3C FE  74 33 3C FB 74 2F 3C FC  юя<ят1<ют3<ыт/<ь
0000000440: 74 31 3C FA 74 33 3C FC  74 35 3C F8 74 37 3C FD  t1<ъt3<ьt5<шt7<э
0000000450: 74 39 3C F9 74 3B E8 74  FF BF 6C 01 88 05 88 65  t9<щt;итяїl0€♦€е
0000000460: 01 BA 6C 01 EB 31 90 BA  0F 01 EB 2B 90 BA 14 01  ☺єl☺л1ђєо☺л+ђє¶☺
0000000470: EB 25 90 BA 1C 01 EB 1F  90 BA 21 01 EB 19 90 BA  л%ђєL☺л▼ђє!☺л↓ђє
0000000480: 30 01 EB 13 90 BA 45 01  EB 0D 90 BA 54 01 EB 07  0☺л‼ђєЕ☺л♪ђєТ☺л•
0000000490: 90 BA 5B 01 EB 01 90 E8  7F FF 07 5A 58 C3 50 52  ђє[☺л☺ђиоя•ZXГPR
00000004A0: 56 57 53 BF 7F 01 B4 30  CD 21 8A F4 E8 47 FF AD  VWSї⌂☺Ґ0Н!ЊфиGя-
00000004B0: 88 65 12 8A C6 E8 3E FF  AD 88 65 14 BA 7F 01 E8  €e↕ЊЖи>я-€е¶єо☺и
00000004C0: 57 FF 8A C7 E8 06 FF BF  97 01 88 45 0F 88 65 10  WяЊЗи♠яї—0€Ео€е►
00000004D0: BA 97 01 E8 43 FF BF AA  01 8A C3 E8 EF FE 88 45  є—0иСяїЄ☺ЊГипю€Е
00000004E0: 0F 88 65 10 8B C1 83 C7  14 E8 F2 FE BA AA 01 E8  о€е►‹БѓЗ¶итюєЄ☺и
00000004F0: 27 FF 5B 5F 5E 5A 58 C3  E8 25 FF E8 A0 FF 32 C0  'я[_^ZXГи%яи я2А
0000000500: B4 4C CD 21                                       ґLH!
```

Рисунок 2. «Плохой» EXE файл в 16-ричном виде.

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

В «хорошем» EXE файле код и данные расположены в отдельных сегментах. Также память под PSP не выделяется, так как отсутсвует директива ORG 100h.

```
C:\MASM\LB1_EXE.EXE
0000000000: 4D 5A 19 00 03 00 01 00   20 00 11 00 FF FF 22 00
0000000010: 00 01 45 F0 10 00 00 00   1E 00 00 00 01 00 4A 01
0000000020: 00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
0000000030: 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
0000000050: 00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
0000000060: 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 00
0000000080: 00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
0000000090: 00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
00000000A0: 00 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 00
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: E9 36 01 24 0F 3C 09 76   02 04 07 04 30 C3 51 8A
0000000220: E0 E8 EF FF 86 C4 B1 04   D2 E8 E8 E6 FF 59 C3 53
0000000230: 8A FC E8 E9 FF 88 25 4F   88 05 4F 8A C7 E8 DE FF
0000000240: 88 25 4F 88 05 5B C3 51   52 32 E4 33 D2 B9 0A 00
0000000250: F7 F1 80 CA 30 88 14 4E   33 D2 3D 0A 00 73 F1 3C
0000000260: 00 74 04 0C 30 88 04 5A   59 C3 50 B4 09 CD 21 58
0000000270: C3 50 52 06 BA 0A 00 E8   F0 FF B8 00 F0 8E C0 26
0000000280: A0 FE FF 3C FF 74 31 3C   FE 74 33 3C FB 74 2F 3C
0000000290: FC 74 31 3C FA 74 33 3C   FC 74 35 3C F8 74 37 3C
```

```
C:\MASM\LB1_EXE.EXE
0000000290: FC 74 31 3C FA 74 33 3C   FC 74 35 3C F8 74 37 3C  ьt1<ъt3<ьt5<шt7<
00000002A0: FD 74 39 3C F9 74 3B E8   74 FF BF 73 00 88 05 88  эt9<щt;итяїs €♦€
00000002B0: 65 01 BA 73 00 EB 31 90   BA 16 00 EB 2B 90 BA 1B  e☺єs л1ђє━ л+ђє←
00000002C0: 00 EB 25 90 BA 23 00 EB   1F 90 BA 28 00 EB 19 90   л%ђє# л▼ђє( л↓ђ
00000002D0: BA 37 00 EB 13 90 BA 4C   00 EB 0D 90 BA 5B 00 EB  є7 л‼ђєL л♪ђє[ л
00000002E0: 07 90 BA 62 00 EB 01 90   E8 7F FF 07 5A 58 C3 50  •ђєb л☺ђиая•ZXГP
00000002F0: 52 56 57 53 BF 86 00 B4   30 CD 21 8A F4 E8 47 FF  RVWSї† ґ0Н!љфиGя
0000000300: AD 88 65 12 8A C6 E8 3E   FF AD 88 65 14 BA 86 00  -€e↕ЉЖи>я-€e¶є†
0000000310: E8 57 FF 8A C7 E8 06 FF   BF 9E 00 88 45 0F 88 65  иWяЊЗи♠яїћ €Eo€e
0000000320: 10 BA 9E 00 E8 43 FF BF   B1 00 8A C3 E8 EF FE 88  ►єћ иСяї± ЊГипю€
0000000330: 45 0F 88 65 10 8B C1 83   C7 14 E8 F2 FE BA B1 00  Eo€e►‹БґЗ¶итює±
0000000340: E8 27 FF 5B 5F 5E 5A 58   C3 B8 15 00 8E D8 E8 20  и'я[_^ZXГё§ ЋШи
0000000350: FF E8 9B FF 32 C0 B4 4C   CD 21 50 43 20 74 79 70  яи›я2АґLH!PC typ
0000000360: 65 20 69 73 20 24 50 43   0D 0A 24 50 43 2F 58 54  e is $PC♪◙$PC/XT
0000000370: 0D 0A 24 41 54 0D 0A 24   50 53 32 20 6D 6F 64 65  ♪◙$AT♪◙$PS2 mode
0000000380: 6C 20 33 30 0D 0A 24 50   53 32 20 6D 6F 64 65 6C  l 30♪◙$PS2 model
0000000390: 20 35 30 20 6F 72 20 36   30 0D 0A 24 50 53 32 20   50 or 60♪◙$PS2
00000003A0: 6D 6F 64 65 6C 20 38 30   0D 0A 24 50 43 6A 72 0D  model 80♪◙$PCjr♪
00000003B0: 0A 24 50 43 20 43 6F 6E   76 65 72 74 69 62 6C 65  ◙$PC Convertible
00000003C0: 0D 0A 24 20 75 6E 6B 6E   6F 77 6E 20 50 43 20 74  ♪◙$ unknown PC t
00000003D0: 79 70 65 0D 0A 24 53 79   73 74 65 6D 20 76 65 72  ype♪◙$System ver
00000003E0: 73 69 6F 6E 20 69 73 20   20 2E 20 0D 0A 24 4F 45  sion is  . ♪◙$OE
00000003F0: 4D 20 6E 75 6D 62 65 72   20 69 73 20 20 20 0D 0A  M number is   ♪◙
0000000400: 24 55 73 65 72 20 6E 75   6D 62 65 72 20 69 73 20  $User number is
0000000410: 20 20 20 20 20 20 0D 0A   24                        ♪◙$
```

Рисунок 3. «Хороший» EXE файл в 16-ричном виде.

Загрузка COM модуля в основную память

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

В памяти размещается PSP, а затем с адреса 100h размещается код.

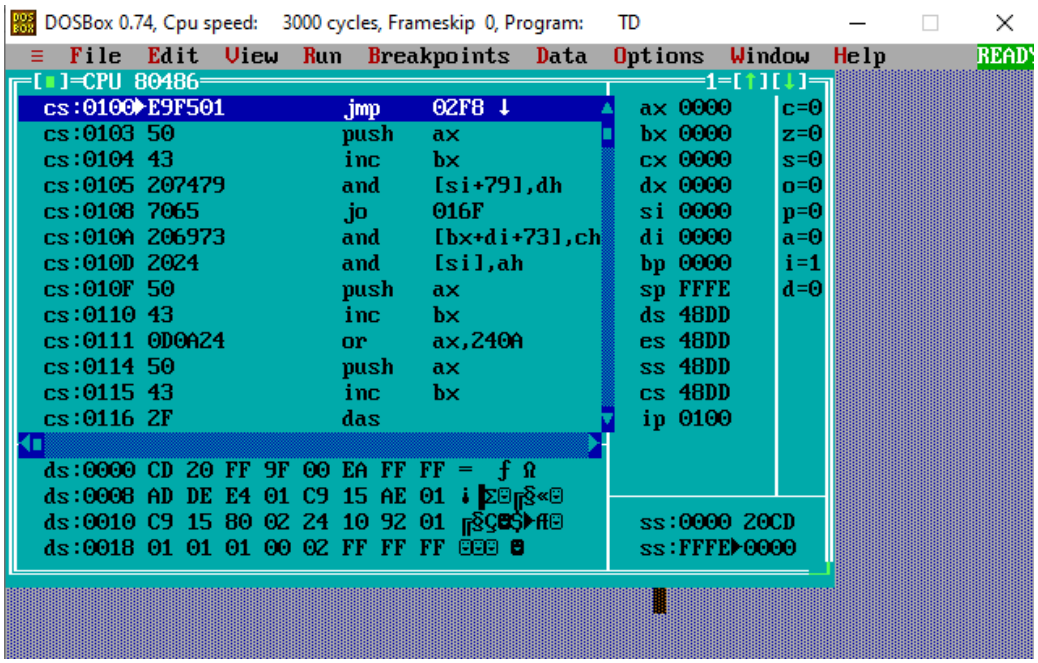

Рисунок 4. COM файл, открытый в отладчике TD.

2. Что располагается с адреса 0?

Начиная с этого адреса располагается PSP длиной в 100h байт.

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

Все сегментные регистры указывают на начало PSP, то есть имеют одинаковое значение.

```
ds 48DD
es 48DD
ss 48DD
cs 48DD
```

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

Автоматически созданный стек занимает память с адресами от FFFEh до 0000h.

Загрузка «хорошего» EXE модуля в основную память

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

Из заголовка EXE модуля считывается информация, в результате чего регистры DS и ES указывают на начало PSP, регистр CS указывает на начало сегмента кода, регистр SS указывает на начало сегмента стека, а IP располагается смещение точки входа.

2. На что указывают регистры DS и ES?

Эти регистры указывают на начало PSP.

3. Как определяется стек?

При помощи директивы .stack объявляется сегмент стека. При этом регистр SS будет указывать на начало, а регистр SP – на конец объявленного сегмента.

4. Как определяется точка входа?

При помощи директивы END, операндом которой является метка, указывающая на начало выполняемого кода.

План загрузки модуля .COM в основную память

1. В основной памяти ищется свободный сегмент.
2. Когда такой сегмент найден, в первые 100h байт загружается PSP. За PSP загружается сам .COM модуль.

Вывод

Исследованы различия в структурах исходных текстов модулей .COM и .EXE, структур файлов загрузочных модулей и способов их загрузки в основную память.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

lb1_exe.asm

```
DOSSEG

	.model small

	.stack 100h


.data

pc_type db 'PC type is ', '$'

is_pc db 'PC', 0dh, 0ah, '$'

is_xt db 'PC/XT', 0dh, 0ah, '$'

is_at db 'AT', 0dh, 0ah, '$'

is_ps3 db 'PS2 model 30', 0dh, 0ah, '$'

is_ps5 db 'PS2 model 50 or 60', 0dh, 0ah, '$'

is_ps8 db 'PS2 model 80', 0dh, 0ah, '$'

is_pcjr db 'PCjr', 0dh, 0ah, '$'

is_pc_con db 'PC Convertible', 0dh, 0ah, '$'

is_unknownpc db ' unknown PC type', 0dh, 0ah, '$'

sys_ver db 'System version is  . ', 0dh, 0ah, '$'

oem_num db 'OEM number is   ', 0dh, 0ah, '$'

user_num db 'User number is       ', 0dh, 0ah, '$'


.code

START: JMP BEGIN

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

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

PRINT_MES PROC near
        push ax
        mov ah, 09h
        int 21h
        pop ax
        ret
PRINT_MES ENDP

DEFINE_PC_TYPE PROC near
        push ax
        push dx
        push es

        mov dx, offset pc_type
        call PRINT_MES

        mov ax, 0f000h
```

```
mov es, ax
mov al, es:[0fffeh]

cmp al, 0ffh
je type_pc
cmp al, 0feh
je type_xt
cmp al, 0fbh
je type_xt
cmp al, 0fch
je type_at
cmp al, 0fah
je type_30
cmp al, 0fch
je type_50
cmp al, 0f8h
je type_80
cmp al, 0fdh
je type_jr
cmp al, 0f9h
je type_conv

;unknown pc type
call BYTE_TO_HEX
mov di, offset is_unknownpc
mov [di], al
mov [di+1], ah
```

```
mov dx, offset is_unknownpc
jmp output_type

type_pc:
mov dx, offset is_pc
jmp output_type
type_xt:
mov dx, offset is_xt
jmp output_type
type_at:
mov dx, offset is_at
jmp output_type
type_30:
mov dx, offset is_ps3
jmp output_type
type_50:
mov dx, offset is_ps5
jmp output_type
type_80:
mov dx, offset is_ps8
jmp output_type
type_jr:
mov dx, offset is_pcjr
jmp output_type
type_conv:
mov dx, offset is_pc_con
jmp output_type
```

```
        output_type:
        call PRINT_MES

        pop es
        pop dx
        pop ax
        ret
DEFINE_PC_TYPE ENDP

DEFINE_OS_OEM_USER PROC near
        push ax
        push dx
        push si
        push di
        push bx

        mov di, offset sys_ver
        mov ah, 30h
        int 21h

        mov dh, ah
        call BYTE_TO_DEC
        lodsw
        mov [di+18], ah
        mov al, dh
```

```
call BYTE_TO_DEC
lodsw
mov [di+20], ah

mov dx, offset sys_ver
call PRINT_MES

mov al, bh
call BYTE_TO_HEX
mov di, offset oem_num
mov [di+15], al
mov [di+16], ah
mov dx, offset oem_num
call PRINT_MES

mov di, offset user_num

mov al, bl
call BYTE_TO_HEX
mov [di+15], al
mov [di+16], ah

mov ax, cx
add di, 20
call WRD_TO_HEX
```

```
	mov dx, offset user_num
	call PRINT_MES

	pop bx
	pop di
	pop si
	pop dx
	pop ax
	ret
DEFINE_OS_OEM_USER ENDP

;------------------------------
BEGIN:
	mov ax, @data
	mov ds, ax
	call DEFINE_PC_TYPE
	call DEFINE_OS_OEM_USER
	xor AL,AL
	mov AH,4Ch
	int 21H
END START
```

lb1_com.asm

```
TESTPC SEGMENT
 ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
	ORG 100H
START: JMP BEGIN
```

```
; ДАННЫЕ
pc_type db 'PC type is ', '$'
is_pc db 'PC', 0dh, 0ah, '$'
is_xt db 'PC/XT', 0dh, 0ah, '$'
is_at db 'AT', 0dh, 0ah, '$'
is_ps3 db 'PS2 model 30', 0dh, 0ah, '$'
is_ps5 db 'PS2 model 50 or 60', 0dh, 0ah, '$'
is_ps8 db 'PS2 model 80', 0dh, 0ah, '$'
is_pcjr db 'PCjr', 0dh, 0ah, '$'
is_pc_con db 'PC Convertible', 0dh, 0ah, '$'
is_unknownpc db ' unknown PC type', 0dh, 0ah, '$'
sys_ver db 'System version is  . ', 0dh, 0ah, '$'
oem_num db 'OEM number is   ', 0dh, 0ah, '$'
user_num db 'User number is       ', 0dh, 0ah, '$'

;ПРОЦЕДУРЫ
;------------------------------------------------
TETR_TO_HEX PROC near
       and AL,0Fh
       cmp AL,09
       jbe NEXT
       add AL,07
       NEXT:
       add AL,30h
       ret
TETR_TO_HEX ENDP
```

```
;-------------------------------
BYTE_TO_HEX PROC near
; байт в AL переводится в два символа шестн. числа в AX
       push CX
       mov AH,AL
       call TETR_TO_HEX
       xchg AL,AH
       mov CL,4
       shr AL,CL
       call TETR_TO_HEX ;в AL старшая цифра
       pop CX ;в AH младшая
       ret
BYTE_TO_HEX ENDP

;-------------------------------
WRD_TO_HEX PROC near
;перевод в 16 с/с 16-ти разрядного числа
; в AX - число, DI - адрес последнего символа
       push BX
       mov BH,AH
       call BYTE_TO_HEX
       mov [DI],AH
       dec DI
       mov [DI],AL
       dec DI
       mov AL,BH
```

```
        call BYTE_TO_HEX
        mov [DI],AH
        dec DI
        mov [DI],AL
        pop BX
        ret
WRD_TO_HEX ENDP

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

```
        mov [SI],AL
        end_1:
        pop DX
        pop CX
        ret
BYTE_TO_DEC ENDP

PRINT_MES PROC near
        push ax
        mov ah, 09h
        int 21h
        pop ax
        ret
PRINT_MES ENDP

DEFINE_PC_TYPE PROC near
        push ax
        push dx
        push es

        mov dx, offset pc_type
        call PRINT_MES

        mov ax, 0f000h
        mov es, ax
        mov al, es:[0fffeh]
```

```
cmp al, 0ffh
je type_pc
cmp al, 0feh
je type_xt
cmp al, 0fbh
je type_xt
cmp al, 0fch
je type_at
cmp al, 0fah
je type_30
cmp al, 0fch
je type_50
cmp al, 0f8h
je type_80
cmp al, 0fdh
je type_jr
cmp al, 0f9h
je type_conv

;unknown pc type
call BYTE_TO_HEX
mov di, offset is_unknownpc
mov [di], al
mov [di+1], ah
mov dx, offset is_unknownpc
jmp output_type
```

```
type_pc:
mov dx, offset is_pc
jmp output_type
type_xt:
mov dx, offset is_xt
jmp output_type
type_at:
mov dx, offset is_at
jmp output_type
type_30:
mov dx, offset is_ps3
jmp output_type
type_50:
mov dx, offset is_ps5
jmp output_type
type_80:
mov dx, offset is_ps8
jmp output_type
type_jr:
mov dx, offset is_pcjr
jmp output_type
type_conv:
mov dx, offset is_pc_con
jmp output_type



output_type:
```

```
        call PRINT_MES

        pop es
        pop dx
        pop ax
        ret
DEFINE_PC_TYPE ENDP

DEFINE_OS_OEM_USER PROC near
        push ax
        push dx
        push si
        push di
        push bx

        ;версия ОС
        mov di, offset sys_ver
        mov ah, 30h
        int 21h

        mov dh, ah
        call BYTE_TO_DEC
        lodsw
        mov [di+18], ah
        mov al, dh

        call BYTE_TO_DEC
```

```
lodsw
mov [di+20], ah

mov dx, offset sys_ver
call PRINT_MES

;OEM номер, находящийся в bh

mov al, bh
call BYTE_TO_HEX
mov di, offset oem_num
mov [di+15], al
mov [di+16], ah
mov dx, offset oem_num
call PRINT_MES

;номер пользователя, находящийся в bl:cx
mov di, offset user_num

;обработаем bl
mov al, bl
call BYTE_TO_HEX
mov [di+15], al
mov [di+16], ah

;обработаем cx
mov ax, cx
```

```
        add di, 20
        call WRD_TO_HEX

        mov dx, offset user_num
        call PRINT_MES

        pop bx
        pop di
        pop si
        pop dx
        pop ax
        ret
DEFINE_OS_OEM_USER ENDP

;------------------------------
; КОД
BEGIN:
        call DEFINE_PC_TYPE
        call DEFINE_OS_OEM_USER
        ; Выход в DOS
        xor AL,AL
        mov AH,4Ch
        int 21H
TESTPC ENDS
        END START ;конец модуля, START - точка входа
```