

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ
по практической работе №1
по дисциплине «Операционные системы»
ТЕМА: ИССЛЕДОВАНИЕ СТРУКТУР ЗАГРУЗОЧНЫХ МОДУЛЕЙ

Студентка гр. 7383

Чемова К.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2019

Постановка задачи.

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

Ход работы.

В данной работе были использованы функции, представленные как шаблон в методических указаниях:

- TETR_TO_HEX – переводит число в шестнадцатеричную систему счисления из двоичной;
- BYTE_TO_HEX – переводит байтовое число из регистре AL в шестнадцатеричную систему счисления
- WRD_TO_HEX – переводит число из регистре AX в шестнадцатеричную систему счисления;
- BYTE_TO_DEC – переводит число в двоичную систему счисления;

Также были написаны функции:

- GET_PC_TYPE – определяет тип компьютера;
- GET_SYS_VERS – определяет версию системы;
- GET_OEM_NUM – определяет серийный номер OEM;
- GET_SERIAL_NUM – определяет серийный номер пользователя;
- PRINT_MSG– выводит на экран строку.

Программа определяет тип компьютера, версию системы, серийный номер OEM и серийный номер пользователя, а затем выводит информацию на экран. Исходный код .COM файла представлен в приложении А, код .EXE файла в приложении Б. Результат работы программы представлен на рис. 1-3.

```

C:\>LR1_COM.COM
PC Type: UB
System version: 5.0
OEM number: 255
User serial number:  FFFFFFFF

```

Рисунок 1 – Результат работы .COM файла

```

C:\>LR1_COM.EXE
UB          5 0          255          FF
FFFF
      0-PC Type:
      5 0          255          FFFFFFFF
0-PC Type:
255          FFFFFFFF
          FFFFFFFF      0-PC Type:
          0-PC Type:

```

Рисунок 2 – Результат работы «плохого» .EXE файла

```

C:\>LR1_EXE.EXE
PC Type: UB
System version: 5.0
OEM number: 255
User serial number:  FFFFFFFF

```

Рисунок 3 – Результат работы «хорошего» .EXE файла

Выводы.

В ходе выполнения лабораторной работы были исследованы различия структур загрузочных модулей типа .COM и .EXE. Было определено, что структура .EXE файла более сложна.

Ответы на контрольные вопросы.

Отличия исходных текстов COM и EXE программ

1. Сколько сегментов должна содержать COM-программа?

Один сегмент.

2. EXE-программа?

Как минимум из одного.

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

Должна быть включена директива `ORG 100h`, сдвигающая адресацию в программе на 256 байт для расположения PSP. Помимо этого, должна присутствовать директива `ASSUME`, ставящая в соответствие начало программы сегментам кода и данных. Отсутствие директивы `ASSUME` не дает программе компилироваться, поскольку невозможно найти начало сегмента кода.

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

Нет, не все. Нельзя использовать дальнюю (`far`) адресацию, поскольку в этих командах используется таблица настройки в которой содержатся адреса сегментов, содержащаяся только в `.EXE` файлах. это означает невозможность использования команды вида `mov register, segment`.

Отличия форматов файлов COM и EXE модулей.

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

Данный вид файлов содержит только машинный код и данные программы. На рис. 4 показано HEX представление файла. Из него видно, код располагается с нулевого адреса.

0000000000: E9 1C 01 50 43 20 54 79	70 65 3A 20 20 0D 0A 24	éL@PC Type: №\$
0000000010: 53 79 73 74 65 6D 20 76	65 72 73 69 6F 6E 3A 20	System version:
0000000020: 20 2E 20 20 0D 0A 24 4F	45 4D 20 6E 75 6D 62 65	. №\$OEM numbe
0000000030: 72 3A 20 20 20 20 20 20	0D 0A 24 55 73 65 72 20	r: №\$User
0000000040: 73 65 72 69 61 6C 20 6E	75 6D 62 65 72 3A 20 20	serial number:
0000000050: 20 20 20 20 20 20 20 20	20 20 20 20 20 0D 0A 24	№\$
0000000060: 04 0F 3C 09 76 02 04 07	04 30 C3 51 8A E0 E8 EF	♦α<ovθ♦♦0ÃQŠàèi
0000000070: FF 86 C4 B1 04 D2 E8 E8	E6 FF 59 C3 53 8A FC E8	ÿ†Ã±♦0èèæÿYÃSŠüè
0000000080: E9 FF 88 25 4F 88 05 4F	8A C7 E8 DE FF 88 25 4F	éÿ`%0`+0ŠÇèbÿ`%0
0000000090: 88 05 5B C3 51 52 32 E4	33 D2 B9 0A 00 F7 F1 80	ˆ♣[ÃQR2ã3D†☐ ÷ñ€
00000000A0: CA 30 88 14 4E 33 D2 3D	0A 00 73 F1 3C 00 74 04	Ê0`°JN3D=☐ sñ< t♦
00000000B0: 0C 30 88 04 5A 59 C3 06	53 50 BB 00 F0 8E C3 26	Q0`♦ZYÃ♣SP» ðŽÃ&
00000000C0: A1 FE FF 8A E0 E8 A3 FF	BB 03 01 89 47 09 58 5B	iþÿŠàèfÿ»♥0%GoX[
00000000D0: 07 50 56 BE 10 01 83 C6	10 E8 B8 FF 83 C6 03 8A	•PV%►0fA►è ÿfA♥Š
00000000E0: C4 E8 B0 FF 5E 58 C3 50	53 56 8A C7 BE 27 01 83	Ãè°ÿ^XÃPSVŠÇ%'0f
00000000F0: C6 0E E8 9F FF 5E 5B 58	C3 50 53 51 56 8A C3 E8	ÆÃèÿÿ^[XÃPSQVŠÃè
0000000100: 69 FF BF 3B 01 83 C7 16	89 05 8B C1 BF 3B 01 83	iÿ¿;0fÇ-☐+<Á¿;0f
0000000110: C7 1B E8 67 FF 5E 59 5B	58 C3 B4 09 CD 21 C3 E8	Ç←ègÿ^Y[XÃ´oÍ!Ãè
0000000120: 95 FF B4 30 CD 21 E8 A8	FF E8 BB FF E8 CA FF BA	•ÿ´0Í!è`ÿè»ÿèÊÿ°
0000000130: 03 01 E8 E5 FF BA 10 01	E8 DF FF BA 27 01 E8 D9	♥0èãÿ°►0èßÿ°'0èÛ
0000000140: FF BA 3B 01 E8 D3 FF 32	C0 B4 4C CD 21	ÿ°;0èÖÿ2À´LÍ!

Рисунок 4 – HEX представление .COM файла

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

В «плохом» EXE код и данные не разделены по сегментам, а перемешаны. Код располагается с адреса 300h, так как заголовок занимает 200h байт, а команда ORG 100h сдвигает код еще на 100h. С нулевого адреса располагается заголовок. В первых двух байтах можно увидеть символы MZ, означающие, что формат файла – 16-битный и его следует запускать в соответствии со структурой .EXE файлов. После заголовка идет таблица настройки. Без которой, файл загружался бы в память как .COM файл.

На рисунке 5 можно увидеть HEX представление «плохого» .EXE файла.

000000000: 4D 5A 4D 00 03 00 00 00	20 00 00 00 FF FF 00 00	MZM ♥ ŷŷ
000000010: 00 00 00 00 00 01 00 00	3E 00 00 00 01 00 FB 50	@ > @ ŷP
000000020: 6A 72 00 00 00 00 00 00	00 00 00 00 00 00 00 00	jr
000000030: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000040: 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 00	
000000060: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000070: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000080: 00 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 00	
0000000B0: 00 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 00	
0000000E0: 00 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	
000000100: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000110: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000120: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000130: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000140: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000150: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000160: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000170: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000180: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000190: 00 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	
0000001C0: 00 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	
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	
000000210: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000220: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000230: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000240: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000250: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000260: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000270: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000280: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000000290: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000002A0: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000002B0: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
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 1C 01 50 43 20 54 79	70 65 3A 20 20 0D 0A 24	éL0PC Type: J\$
000000310: 53 79 73 74 65 6D 20 76	65 72 73 69 6F 6E 3A 20	System version:
000000320: 20 2E 20 20 0D 0A 24 4F	45 4D 20 6E 75 6D 62 65	. J\$OEM numbe
000000330: 72 3A 20 20 20 20 20 20	0D 0A 24 55 73 65 72 20	r: J\$User
000000340: 73 65 72 69 61 6C 20 6E	75 6D 62 65 72 3A 20 20	serial number:
000000350: 20 20 20 20 20 20 20 20	20 20 20 20 20 0D 0A 24	J\$
000000360: 04 0F 3C 09 76 02 04 07	04 30 C3 51 8A E0 E8 EF	♦◁ov♦♦♦Q5àèi
000000370: FF 86 C4 B1 04 D2 E8 E8	E6 FF 59 C3 53 8A FC E8	ŷ†Ä±♦DèèæŷYÄS5üè
000000380: E9 FF 88 25 4F 88 05 4F	8A C7 E8 DE FF 88 25 4F	éŷ`%0`+OŠCèbŷ`%0
000000390: 88 05 5B C3 51 52 32 E4	33 D2 B9 0A 00 F7 F1 80	^+[[ÄQR2ä3D¹ ÷ñ€
0000003A0: CA 30 88 14 4E 33 D2 3D	0A 00 73 F1 3C 00 74 04	Ê0`JN3D= sñ< t♦
0000003B0: 0C 30 88 04 5A 59 C3 06	53 50 BB 00 F0 8E C3 26	90`♦ZYÄSP» ðŽÄ&
0000003C0: A1 FE FF 8A E0 E8 A3 FF	BB 03 01 89 47 09 58 5B	iþŷ5àèŷ»♥0%GoX[
0000003D0: 07 50 56 BE 10 01 83 C6	10 E8 B8 FF 83 C6 03 8A	•PV%◁ofA◁è.ŷfA♥Š
0000003E0: C4 E8 B0 FF 5E 58 C3 50	53 56 8A C7 BE 27 01 83	Äè°ŷ^XÄPSVŠC%`of
0000003F0: C6 0E E8 9F FF 5E 5B 58	C3 50 53 51 56 8A C3 E8	Äèèŷŷ^ [XÄPSQVŠÄè
000000400: 69 FF BF 3B 01 83 C7 16	89 05 8B C1 BF 3B 01 83	iŷ¿;ofC`%+◁Ä¿;of
000000410: C7 1B E8 67 FF 5E 59 5B	58 C3 B4 09 CD 21 C3 E8	Ç◁ègŷ^Y[XÄ`oÍ!Äè
000000420: 95 FF B4 30 CD 21 E8 A8	FF E8 BB FF E8 CA FF BA	•ŷ`0Í!è`ŷè»ŷèÊŷ°
000000430: 03 01 E8 E5 FF BA 10 01	E8 DF FF BA 27 01 E8 D9	♥0èäŷ°◁0èßŷ°`0èÜ
000000440: FF BA 3B 01 E8 D3 FF 32	C0 B4 4C CD 21	ŷ°;0èÜŷ2Ä`LÍ!

Рисунок 5 – HEX представление «плохого» .EXE файла

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

В отличие от «плохого» .EXE файла, в «хорошем» код, стек и данные выделены в отдельные сегменты. Для «хорошего» .EXE файла в директиве `ORG 100h` нет необходимости, загрузчик автоматически расположит программу после PSP. Код программы начинается с `200h`.

0000000000: 4D 5A 77 01 02 00 01 00	20 00 00 00 FF FF 06 00	MZw@ @	ÿÿ
0000000010: 20 00 00 00 BF 00 08 00	3E 00 00 00 01 00 FB 50	¿ >	û P
0000000020: 6A 72 00 00 00 00 00 00	00 00 00 00 00 00 00 00	jr	
0000000030: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 C3 00		Ã
0000000040: 08 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: 50 43 20 54 79 70 65 3A	20 20 0D 0A 24 53 79 73	PC Type:	\$Sys
0000000210: 74 65 6D 20 76 65 72 73	69 6F 6E 3A 20 20 2E 20	tem version: .	
0000000220: 20 0D 0A 24 4F 45 4D 20	6E 75 6D 62 65 72 3A 20	\$OEM number:	
0000000230: 20 20 20 20 20 0D 0A 24	55 73 65 72 20 73 65 72	\$User ser	
0000000240: 69 61 6C 20 6E 75 6D 62	65 72 3A 20 20 20 20 20	ial number:	
0000000250: 20 20 20 20 20 20 20 20	20 20 0D 0A 24 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: 04 0F 3C 09 76 02 04 07	04 30 C3 51 8A E0 E8 EF	♦<ov♦♦♦0ÃQŠàèi	
0000000290: FF 86 C4 B1 04 D2 E8 E8	E6 FF 59 C3 53 8A FC E8	ÿtÃ±♦ðèèæÿYÃSŠüè	
00000002A0: E9 FF 88 25 4F 88 05 4F	8A C7 E8 DE FF 88 25 4F	éÿ^%0^+0ŠÇèbÿ^%0	
00000002B0: 88 05 5B C3 51 52 32 E4	33 D2 B9 0A 00 F7 F1 80	^+ [ÃQR2ä3ð¹ ÷ñ€	
00000002C0: CA 30 88 14 4E 33 D2 3D	0A 00 73 F1 3C 00 74 04	Ê0^JN3ð= sñ< t♦	
00000002D0: 0C 30 88 04 5A 59 C3 06	53 50 BB 00 F0 8E C3 26	ð0^♦ZYÃ+SP» ðŽÃ&	
00000002E0: A1 FE FF 8A E0 E8 A3 FF	BB 00 00 89 47 09 58 5B	iþÿŠàèfÿ» %GoX[
00000002F0: 07 50 56 BE 0D 00 83 C6	10 E8 B8 FF 83 C6 03 8A	•PV%» fÆ»è.ÿfA♥Š	
0000000300: C4 E8 B0 FF 5E 58 C3 50	53 56 8A C7 BE 24 00 83	Ãè°ÿ^XÃPSVŠC%\$ f	
0000000310: C6 0E E8 9F FF 5E 5B 58	C3 50 53 51 56 8A C3 E8	Æðèÿÿ^[XÃPSQVŠÃè	
0000000320: 69 FF BF 38 00 83 C7 16	89 05 8B C1 BF 38 00 83	iÿ¿8 fÇ=»+Ã¿8 f	
0000000330: C7 1B E8 67 FF 5E 59 5B	58 C3 B4 09 CD 21 C3 1E	Ç<ègÿ^Y[XÃ´oÍ!Ã▲	
0000000340: 2B C0 B8 00 00 8E D8 2B	C0 E8 8B FF B4 30 CD 21	+Ã. Žð+Ãè<ÿ´0Í!	
0000000350: E8 9E FF E8 B1 FF E8 C0	FF BA 00 00 E8 DB FF BA	èžÿè±ÿèÃÿ° èÜÿ°	
0000000360: 0D 00 E8 D5 FF BA 24 00	E8 CF FF BA 38 00 E8 C9	» èÕÿ°\$ èÿÿ°8 èÉ	
0000000370: FF 32 C0 B4 4C CD 21		ÿ2Ã´LÍ!	

Рисунок 6 – HEX представление «хорошего» .EXE файла

Загрузка COM модуля в основную память.

На рис. 7 показан отладчик, в котором запущен .COM файл.

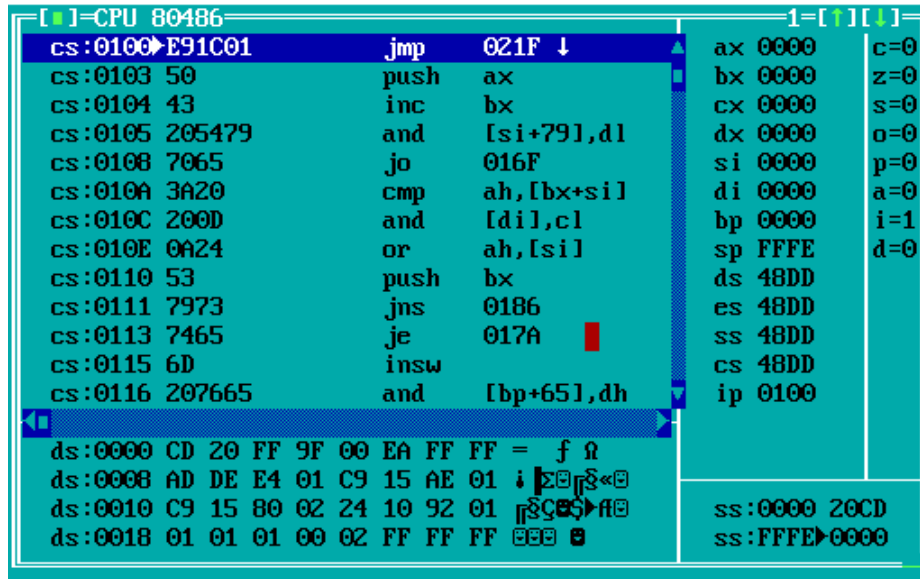


Рисунок 7 – .COM файл в отладчике

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

Сначала система выделяет свободный сегмент памяти, затем заносит его адрес во все сегментные регистры. Потом происходит построение PSP в первых 100h байтах памяти. Загружается содержимое .COM файла и присваивается регистру IP значение 100h. Указатель на стек SP ссылается на конец выделенной памяти, поэтому запись данных стека осуществляется снизу вверх.

Начало кода определяется директивой ORG 100h от начала выделенного фрагмента. Код располагается с адреса 48DD:0100h.

2. Что располагается с адреса 0?

По этому адресу располагается PSP.

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

Все сегментные регистры имеют значения 48DD, это видно из рис. 7. Они указывают на начало PSP.

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

Стек занимает весь сегмент COM-программы, его начало находится в конце сегмента. SS указывает на начало сегмента, а SP=FFFFh – на его конец. Стек может дойти до кода или данных программы при достаточном количестве элементов. 10 Адреса расположены в диапазоне 0000h-FFFFh. Стек растет от больших адресов к меньшим. Это означает, что стек может при большом количестве элементов переписать код или данные программы.

Загрузка «хорошего» EXE модуля в основную память.

На рис. 8 показан отладчик, в котором запущен .EXE файл.

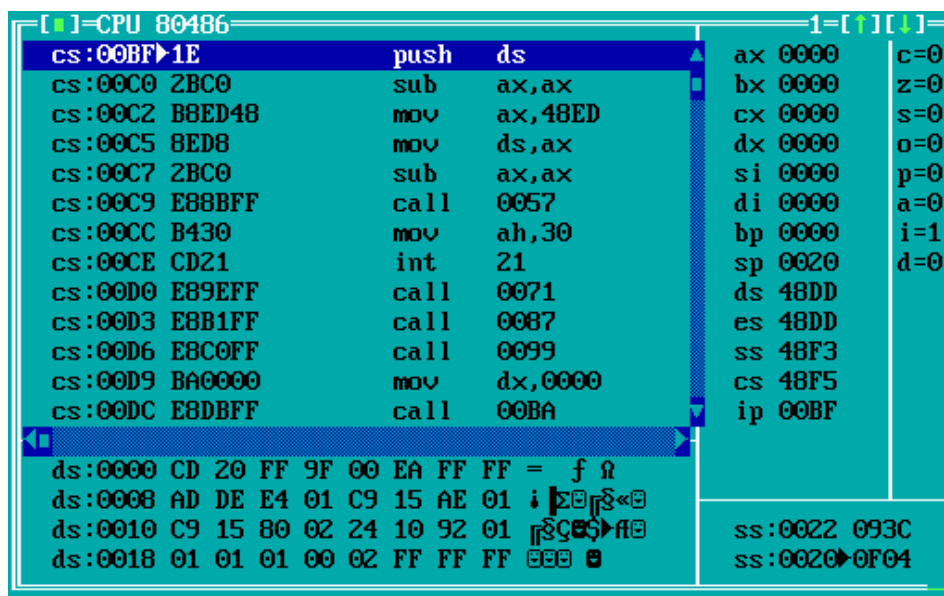


Рисунок 7 – .EXE файл в отладчике

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

Сначала загружается PSP, а дальше устанавливаются сегментные регистры. DS и ES устанавливаются на начало PSP (48DD). SS – на начало сегмента стека (48F3), CS – на начало сегмента кода (48F5).

2. На что указывают регистры DS и ES?

Они указывают на начало PSP.

3. Как определяется стек?

В исходном коде модуля стек определяется при помощи директивы `STACK`. При запуске программы в `SS` заносится адрес сегмента стека, его начало, а в `SP` – адрес его вершины.

4. Как определяется точка входа?

Точка входа в программу определяется с помощью директивы `END`. После этой директивы указывается метка, куда переходит программа при запуске.

ПРИЛОЖЕНИЕ А

LR1_COM.ASM

TESTPC SEGMENT

```
        ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING;
        ORG     100H
START:  JMP     BEGIN
;ДАННЫЕ
PC_TYPE      db      'PC Type: ',0DH,0AH,'$'
SYS_VERS     db      'System version: . ',0DH,0AH,'$'
OEM_NUM      db      'OEM number: ',0DH,0AH,'$'
SER_NUM      db      'User serial number: ',0DH,0AH,'$'
;ПРОЦЕДУРЫ
;-----
TETR_TO_HEX  PROC  near
        add     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 переводиться в два символа шестнадцатеричного 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 CC 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 10 CC, 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_1
        or      AL, 30h
        mov     [SI], AL
end_1: pop     DX
        pop     CX
        ret
BYTE_TO_DEC ENDP
;-----
GET_PC_TYPE PROC near
        push    ES
        push    BX
        push    AX
        mov     BX, 0F000H
        mov     ES, BX
        mov     AX, ES:[0FFFFH]
        mov     AH, AL
        call    BYTE_TO_HEX
        lea     BX, PC_TYPE
        mov     [BX+9], AX
        pop     AX
        pop     BX
        pop     ES
GET_PC_TYPE ENDP
;-----
GET_SYS_VERS PROC near
        push    AX
        push    SI
        lea     SI, SYS_VERS
        add     SI, 16      ;основная версия
        call    BYTE_TO_DEC
        add     SI, 3       ;модификация
        mov     AL, Ah
        call    BYTE_TO_DEC

```

```

        pop     SI
        pop     AX
        ret
GET_SYS_VERS ENDP
;-----
GET_OEM_NUM PROC  near
        push    AX
        push    BX
        push    SI
        mov     AL, BH
        lea     SI, OEM_NUM
        add     SI, 14
        call    BYTE_TO_DEC
        pop     SI
        pop     BX
        pop     AX
        ret
GET_OEM_NUM ENDP
;-----
GET_SERIAL_NUM PROC  near
        push    AX
        push    BX
        push    CX
        push    SI
        mov     AL, BL
        call    BYTE_TO_HEX
        lea     DI, SER_NUM
        add     DI, 22
        mov     [DI], AX
        mov     AX, CX
        lea     DI, SER_NUM
        add     DI, 27
        call    WRD_TO_HEX
        pop     SI
        pop     CX
        pop     BX
        pop     AX
        ret
GET_SERIAL_NUM ENDP
;-----
PRINT_MSG PROC  near
        mov     AH, 09h
        int     21h
        ret
PRINT_MSG ENDP
;-----
BEGIN:
;Определение типа ПК и версии ОС
        call    GET_PC_TYPE
        mov     AH, 30h
        int     21h
        call    GET_SYS_VERS
        call    GET_OEM_NUM

```

```

        call    GET_SERIAL_NUM
;-----
;Вывод информации
        lea     DX, PC_TYPE
        call    PRINT_MSG
        lea     DX, SYS_VERS
        call    PRINT_MSG
        lea     DX, OEM_NUM
        call    PRINT_MSG
        lea     DX, SER_NUM
        call    PRINT_MSG
;-----
;Выход в DOS
        xor     AL, AL
        mov     AH, 4Ch
        int     21h
TESTPC ENDS
        END     START ;Конец модуля, START - точка входа
;-----

```


ПРИЛОЖЕНИЕ Б

LR1_EXE.ASM

```
DATA SEGMENT
PC_TYPE      db      'PC Type:  ',0DH,0AH,'$'
SYS_VERS     db      'System version:  .  ',0DH,0AH,'$'
OEM_NUM      db      'OEM number:      ',0DH,0AH,'$'
SER_NUM      db      'User serial number:          ',0DH,0AH,'$'
DATA ENDS

AStack SEGMENT STACK
        DW 010H DUP(?)
AStack ENDS
CODE SEGMENT
        ASSUME CS:CODE, DS:DATA, ES:NOTHING, SS:AStack
;ПРОЦЕДУРЫ
;-----
TETR_TO_HEX  PROC  near
        add     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 переводиться в два символа шестнадцатеричного 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 CC 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 10 CC, 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_l: pop     DX
        pop     CX
        ret
BYTE_TO_DEC ENDP
;-----
GET_PC_TYPE PROC    near
        push    ES
        push    BX
        push    AX
        mov     BX, 0F000H
        mov     ES, BX
        mov     AX, ES:[0FFFEH]
        mov     AH, AL
        call    BYTE_TO_HEX
        lea     BX, PC_TYPE
        mov     [BX+9], AX
        pop     AX
        pop     BX
        pop     ES
GET_PC_TYPE ENDP
;-----
GET_SYS_VERS PROC    near
        push    AX
        push    SI
        lea     SI, SYS_VERS
        add     SI, 16      ;основная версия
        call    BYTE_TO_DEC

```

```

        add     SI, 3           ;модификация
        mov     AL, Ah
        call    BYTE_TO_DEC
        pop     SI
        pop     AX
        ret

```

```
GET_SYS_VERS  ENDP
```

```
;-----
```

```
GET_OEM_NUM  PROC  near
        push    AX
        push    BX
        push    SI
        mov     AL, BH
        lea     SI, OEM_NUM
        add     SI, 14
        call    BYTE_TO_DEC
        pop     SI
        pop     BX
        pop     AX
        ret

```

```
GET_OEM_NUM  ENDP
```

```
;-----
```

```
GET_SERIAL_NUM  PROC  near
        push    AX
        push    BX
        push    CX
        push    SI
        mov     AL, BL
        call    BYTE_TO_HEX
        lea     DI, SER_NUM
        add     DI, 22
        mov     [DI], AX
        mov     AX, CX
        lea     DI, SER_NUM
        add     DI, 27
        call    WRD_TO_HEX
        pop     SI
        pop     CX
        pop     BX
        pop     AX
        ret

```

```
GET_SERIAL_NUM  ENDP
```

```
;-----
```

```
PRINT_MSG  PROC  near
        mov     AH, 09h
        int     21h
        ret

```

```
PRINT_MSG  ENDP
```

```
;-----
```

```
MAIN  PROC  near
        push    DS
        sub     AX, AX
        mov     AX, DATA

```

```

        mov     DS, AX
        sub     AX, AX
;Определение типа ПК и версии ОС
        call    GET_PC_TYPE
        mov     AH, 30h
        int     21h
        call    GET_SYS_VERS
        call    GET_OEM_NUM
        call    GET_SERIAL_NUM
;-----
;Вывод информации
        lea     DX, PC_TYPE
        call    PRINT_MSG
        lea     DX, SYS_VERS
        call    PRINT_MSG
        lea     DX, OEM_NUM
        call    PRINT_MSG
        lea     DX, SER_NUM
        call    PRINT_MSG
;-----
;Выход в DOS
        xor     AL, AL
        mov     AH, 4Ch
        int     21h
MAIN     ENDP
CODE     ENDS
        END MAIN
;-----

```