

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

Студентка гр. 7383

Ханова Ю.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 ходе выполнения данной работы был создан набор функций и структур данных, описанных в табл. 1-2.

Таблица 1 – Описание функций.

Название функции	Назначение
System	Печатает тип ОС
Ver_OS	Печатает версию ОС
Write_oem	Печатает серийный номер OEM
Num_of_serial	Печатает серийный номер пользователя
Write	Вызывает функцию печати строки
TETR_TO_HEX	Вспомогательная функция для работы функции BYTE_TO_HEX
BYTE_TO_HEX	Переводит число AL в коды символов 16-ой с/с, записывая получившееся в BL и BH
WRD_TO_HEX	Переводит число AX в строку в 16-ой с/с, записывая получившееся в di, начиная с младшей цифры
BYTE_TO_DEC	Переводит байт из AL в десятичную с/с и записывает получившееся число по адресу SI, начиная с младшей цифры

Таблица 2 - Описание структур данных.

Название	Тип	Назначение
OS	db	Тип ОС
OS_VERS	db	Версия ОС
OS_OEM	db	Серийный номер OEM
SER_NUM	db	Серийный номер пользователя
PC	db	PC
PCXT	db	PC/XT
_AT	db	AT
PS2_30	db	PS2 модель 30
PS2_80	db	PS2 модель 80
PCjr	db	PCjr
PC_Cnv	db	PC Convertible

Были созданы файлы типов .COM и .EXE («плохой» и «хороший»)

Программа определяет и выводит на экран следующие значения в заданном порядке: тип ОС, версия ОС, серийный номер пользователя, серийный номер OEM. Результаты работы программы представлены на рис. 1-3.

```
C:\>lr1_1.com
Type OS: AT
Version OS: 5 0
Serial number: 000000
OEM: 255
```

Рисунок 1 – Результат выполнения программы lr1_1.com

```
C:\>lr1_1.exe

Type OS:

Version OS: 5 0

Serial number: 000000

DEM: 255
```

Рисунок 2 – Результат выполнения программы lr1_1.exe

```
C:\>lr1_2.exe
Type OS: AT
Version OS: 5 0
Serial number: 000000
DEM: 255
```

Рисунок 3 – Результат выполнения программы lr1_2.exe

Выводы.

В процессе выполнения данной лабораторной работы были исследованы различия в структурах исходных текстов модулей типов .COM и .EXE, структур файлов загрузочных модулей и способов их загрузки в основную память. Код программы lr1_1.asm и lr1_2.asm представлены в приложении А.

Ответы на контрольные вопросы.

Отличия исходных текстов COM и EXE программ

1. Сколько сегментов должна содержать COM-программа?

1 сегмент.

2. EXE-программа?

Минимум 1 сегмент.

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

В тексте COM-программы обязательно должна быть директива `ORG 100h`, которая сдвигает адресацию в программе на 256 байт для расположения PSP.

Так же, должна присутствовать директива `ASSUME`, ставящая в соответствие начало программы сегментам кода и данных (при отсутствии директивы `ASSUME`, программа не скомпилируется из-за невозможности обнаружения начала сегмента кода).

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

Нет, в COM-программе нельзя использовать команды вида `mov register, segment` и команды, содержащие дальнюю (`far`) адресацию, т.к. в этих командах используется таблица настройки в которой содержатся адреса сегментов. Такая таблица есть только в EXE-файлах, поэтому COM-программа не может использовать сегментную адресацию.

Отличия форматов файлов COM и EXE модулей

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

HEX-представление .COM файла показано на рис. 4.

00000000: E9 AC 01 54 79 70 65 20	4F 53 3A 20 24 56 65 72	é-@Type OS: \$Ver
00000010: 73 69 6F 6E 20 4F 53 3A	20 20 20 2E 20 20 0D 0A	sion OS: . \$
00000020: 24 4F 45 4D 3A 20 20 20	20 0D 0A 24 53 65 72 69	\$OEM: \$Ser
00000030: 61 6C 20 6E 75 6D 62 65	72 3A 20 24 20 20 20 20	al number: \$
00000040: 24 0D 0A 24 50 43 0D 0A	24 50 43 2F 58 54 0D 0A	\$J\$PC\$PC/XT
00000050: 24 41 54 0D 0A 24 50 53	32 20 6D 6F 64 65 6C 20	\$AT\$PS2 model
00000060: 33 30 0D 0A 24 50 53 32	20 6D 6F 64 65 6C 20 38	30\$PS2 model 8
00000070: 30 0D 0A 24 50 43 6A 72	0D 0A 24 50 43 20 43 6F	0\$PCjr\$PC Co
00000080: 6E 76 65 72 74 69 62 6C	65 0D 0A 24 B4 09 CD 21	nvertible\$'oÍ!
00000090: C3 24 0F 3C 09 76 02 04	07 04 30 C3 B8 00 F0 8E	Å&ov♦♦♦♦Å, dŽ
000000A0: C0 26 A1 FE FF C3 BA 03	01 E8 E0 FF E8 ED FF 3C	Å&þÿÅ°♥øèàÿèÿ<
000000B0: FF 74 1C 3C FE 74 1E 3C	FB 74 1A 3C FC 74 1C 3C	ÿtL<ptÅ<ût-<ÿtL<
000000C0: FA 74 1E 3C F8 74 20 3C	FD 74 22 3C F9 74 24 BA	ÿtÅ<øt <ÿt"<ÿt\$°
000000D0: 44 01 EB 22 90 BA 49 01	EB 1C 90 BA 51 01 EB 16	Døë"Ø°IØèLØ°QØë-
000000E0: 90 BA 56 01 EB 10 90 BA	65 01 EB 0A 90 BA 74 01	Ø°VØë-Ø°eØëØ°tØ
000000F0: EB 04 90 BA 7B 01 E8 93	FF C3 B8 00 00 B4 30 CD	ë♦Ø°{øè"ÿÅ, 'øÍ
00000100: 21 BE 0D 01 83 C6 0C 50	E8 81 00 58 8A C4 83 C6	!%ØøfÅ9PøØ X\$ÅfÅ
00000110: 03 E8 78 00 BA 0D 01 E8	72 FF C3 B8 00 00 B4 30	♥èx °JØèrÿÅ, 'ø
00000120: CD 21 BE 21 01 83 C6 07	8A C7 E8 5F 00 BA 21 01	Í!%!øfÅ•\$Çè, °!ø
00000130: E8 59 FF C3 BA 2C 01 E8	52 FF 8A C3 E8 24 00 8B	èÿÿÅ°,øèrÿ\$Åè\$ <
00000140: D8 8A D3 B4 02 CD 21 8A	D7 CD 21 BF 3C 01 83 C7	ø\$Ø'øÍ!\$×Í!¿<øfÇ
00000150: 03 8B C1 E8 1E 00 BA 3C	01 E8 30 FF BA 41 01 E8	♥<Åè▲ °<øèøÿ°Aøè
00000160: 2A FF C3 51 8A E0 E8 28	FF 86 C4 B1 04 D2 E8 E8	*ÿÅQ\$Åè(ÿtÅ±øèè
00000170: 1F FF 59 C3 53 8A FC E8	E9 FF 88 25 4F 88 05 4F	ÿÿYÅ\$Süèèÿ`%O`+O
00000180: 8A C7 E8 DE FF 88 25 4F	88 05 5B C3 51 52 32 E4	\$Çèþÿ`%O`+ [ÅQR2ä
00000190: 33 D2 B9 0A 00 F7 F1 80	CA 30 88 14 4E 33 D2 3D	3Ø¹ ÷ñ€ÈØ`¶N3Ø=
000001A0: 0A 00 73 F1 3C 00 74 04	0C 30 88 04 5A 59 C3 E8	sñ< t♦øØ^♦ZYÅè
000001B0: F4 FE E8 45 FF E8 7C FF	E8 60 FF 32 C0 B4 4C CD	øþèEÿè ÿè`ÿ2Å`LÍ
000001C0: 21		!

Рисунок 4 – HEX представление .COM файла

COM-файл содержит только код и данные. В файле код располагается с нулевого адреса.

2. Какова структура файла «плохого» EXE? С какого адреса располагается код? Что располагается с адреса 0? HEX-представление «плохого» .EXE файла показано на рис. 5-6.

0000000000: 4D 5A C1 00 03 00 00 00	20 00 00 00 FF FF 00 00	MZÁ ♥	ÿÿ
0000000010: 00 00 00 00 00 01 00 00	3E 00 00 00 01 00 FB 50	⊙ > ⊙ ûP	
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 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		

Рисунок 5 – HEX-представление «плохого» .EXE (1)

0000000300: E9 AC 01 54 79 70 65 20	4F 53 3A 20 24 56 65 72	é-@Type OS: \$Ver
0000000310: 73 69 6F 6E 20 4F 53 3A	20 20 20 2E 20 20 0D 0A	sion OS: . ♪
0000000320: 24 4F 45 4D 3A 20 20 20	20 0D 0A 24 53 65 72 69	\$OEM: ♪\$Seri
0000000330: 61 6C 20 6E 75 6D 62 65	72 3A 20 24 20 20 20 20	al number: \$
0000000340: 24 0D 0A 24 50 43 0D 0A	24 50 43 2F 58 54 0D 0A	\$♪\$PC♪\$PC/XT♪
0000000350: 24 41 54 0D 0A 24 50 53	32 20 6D 6F 64 65 6C 20	\$AT♪\$PS2 model
0000000360: 33 30 0D 0A 24 50 53 32	20 6D 6F 64 65 6C 20 38	30♪\$PS2 model 8
0000000370: 30 0D 0A 24 50 43 6A 72	0D 0A 24 50 43 20 43 6F	0♪\$PCjr♪\$PC Co
0000000380: 6E 76 65 72 74 69 62 6C	65 0D 0A 24 B4 09 CD 21	nvertible♪\$´oÍ!
0000000390: C3 24 0F 3C 09 76 02 04	07 04 30 C3 B8 00 F0 8E	Ã\$<ov♦♦♦0Ã, ðŽ
00000003A0: C0 26 A1 FE FF C3 BA 03	01 E8 E0 FF E8 ED FF 3C	À&¡þÿÃ°♥œàÿëíÿ<
00000003B0: FF 74 1C 3C FE 74 1E 3C	FB 74 1A 3C FC 74 1C 3C	ÿtL<pt<ût-><ütL<
00000003C0: FA 74 1E 3C F8 74 20 3C	FD 74 22 3C F9 74 24 BA	út<øt <ÿt" <üt\$°
00000003D0: 44 01 EB 22 90 BA 49 01	EB 1C 90 BA 51 01 EB 16	D0ë"⌘°I0ëL⌘°Q0ë-
00000003E0: 90 BA 56 01 EB 10 90 BA	65 01 EB 0A 90 BA 74 01	⌘°V0ë-⌘°e0ë⌘°t0
00000003F0: EB 04 90 BA 7B 01 E8 93	FF C3 B8 00 00 B4 30 CD	ë♦⌘°{0ë"ÿÃ, ´0Í
0000000400: 21 BE 0D 01 83 C6 0C 50	E8 81 00 58 8A C4 83 C6	!%♪ofÆ9Pè⌘ XŠÄfÆ
0000000410: 03 E8 78 00 BA 0D 01 E8	72 FF C3 B8 00 00 B4 30	♥èx °♪0èrÿÃ, ´0
0000000420: CD 21 BE 21 01 83 C6 07	8A C7 E8 5F 00 BA 21 01	Í!%!ofÆ•ŠÇè °!0
0000000430: E8 59 FF C3 BA 2C 01 E8	52 FF 8A C3 E8 24 00 8B	èYÿÃ°,0èRÿŠÃè\$ <
0000000440: D8 8A D3 B4 02 CD 21 8A	D7 CD 21 BF 3C 01 83 C7	ØŠÓ´0Í!Š×Í!¿<0fÇ
0000000450: 03 8B C1 E8 1E 00 BA 3C	01 E8 30 FF BA 41 01 E8	♥<Ãè▲ °<0è0ÿ°A0è
0000000460: 2A FF C3 51 8A E0 E8 28	FF 86 C4 B1 04 D2 E8 E8	*ÿÃQŠàè(ÿ†Ã±♦0èè
0000000470: 1F FF 59 C3 53 8A FC E8	E9 FF 88 25 4F 88 05 4F	▼ÿYÃŠSüèéÿ`%0`+0
0000000480: 8A C7 E8 DE FF 88 25 4F	88 05 5B C3 51 52 32 E4	ŠÇèþÿ`%0`+ [ÃQR2ä
0000000490: 33 D2 B9 0A 00 F7 F1 80	CA 30 88 14 4E 33 D2 3D	30¹☐ ÷ñ€É0`JN30=
00000004A0: 0A 00 73 F1 3C 00 74 04	0C 30 88 04 5A 59 C3 E8	☐ sñ< t♦90`♦ZYÃè
00000004B0: F4 FE E8 45 FF E8 7C FF	E8 60 FF 32 C0 B4 4C CD	òpèEÿè ÿè`ÿ2Ã`LÍ
00000004C0: 21		!

Рисунок 6 – HEX-представление «плохого» .EXE (2)

В «плохом» EXE код и данные не разделены по сегментам, а перемешаны (на скриншоте перед данными видно метку перехода E9 AE 01). Код располагается с адреса 300h, т.к. заголовок занимает 200h байт (байты 8 и 9 указывают, сколько параграфов занимает заголовок) и команда ORG 100h «сдвигает» код на дополнительные 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. За заголовком следует таблица настройки. Если их убрать, то файл будет загружаться в память как COM-файл.

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

HEX-представление «хорошего» .EXE файла показано на рис 7-8.

00000000: 4D 5A CA 01 03 00 01 00	20 00 00 00 FF FF 00 00	MZK000	яя
00000010: 00 02 00 00 23 01 29 00	3E 00 00 00 01 00 FB 50	0 #0) >	0 ыP
00000020: 6A 72 00 00 00 00 00 00	00 00 00 00 00 00 00 00	jr	
00000030: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 24 01		\$0
00000040: 29 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		

Рисунок 7 – HEX-представление «хорошего» .EXE (1)


```

0000000400: 54 79 70 65 20 4F 53 3A 20 24 56 65 72 73 69 6F Type OS: $Versio
0000000410: 6E 20 4F 53 3A 20 20 20 2E 20 20 0D 0A 24 4F 45 n OS: .  $OE
0000000420: 4D 3A 20 20 20 20 0D 0A 24 53 65 72 69 61 6C 20 M:  $Serial
0000000430: 6E 75 6D 62 65 72 3A 20 24 20 20 20 24 0D 0A number: $  $
0000000440: 24 50 43 0D 0A 24 50 43 2F 58 54 0D 0A 24 41 54 $PC $PC/XT $AT
0000000450: 0D 0A 24 50 53 32 20 6D 6F 64 65 6C 20 33 30 0D $PS2 model 30
0000000460: 0A 24 50 53 32 20 6D 6F 64 65 6C 20 38 30 0D 0A $PS2 model 80
0000000470: 24 50 43 6A 72 0D 0A 24 50 43 20 43 6F 6E 76 65 $PCjr $PC Conve
0000000480: 72 74 69 62 6C 65 0D 0A 24 00 00 00 00 00 00 00 rtible $
0000000490: B4 09 CD 21 C3 24 0F 3C 09 76 02 04 07 04 30 C3 гоH!Г$<ov0♦♦♦Г
00000004A0: B8 00 F0 8E C0 26 A1 FE FF C3 BA 00 00 E8 E0 FF ë рhA&ÿюяГе иая
00000004B0: E8 ED FF 3C FF 74 1C 3C FE 74 1E 3C FB 74 1A 3C иня<ятL<ют▲<ыт→<
00000004C0: FC 74 1C 3C FA 74 1E 3C F8 74 20 3C FD 74 22 3C ьтL<ьт▲<шт<эт" <
00000004D0: F9 74 24 BA 41 00 EB 22 90 BA 46 00 EB 1C 90 BA щт$εA л"ñеF ллñе
00000004E0: 4E 00 EB 16 90 BA 53 00 EB 10 90 BA 62 00 EB 0A N л-ñеS л-ñеб л
00000004F0: 90 BA 71 00 EB 04 90 BA 78 00 E8 93 FF C3 B8 00 ñеq лñеx и"яГё
0000000500: 00 B4 30 CD 21 BE 0A 00 83 C6 0C 50 E8 81 00 58 г0H!s fЖ9PиГ X
0000000510: 8A C4 83 C6 03 E8 78 00 BA 0A 00 E8 72 FF C3 B8 ЬдГЖVиx е ияГё
0000000520: 00 00 B4 30 CD 21 BE 1E 00 83 C6 07 8A C7 E8 5F г0H!s fЖ•Ьзи_
0000000530: 00 BA 1E 00 E8 59 FF C3 BA 29 00 E8 52 FF 8A C3 ε▲ иYяГе) иРяЬГ
0000000540: E8 24 00 8B D8 8A D3 B4 02 CD 21 8A D7 CD 21 BF и$ <ШЬг0H!ьЧN!i
0000000550: 39 00 83 C7 03 8B C1 E8 1E 00 BA 39 00 E8 30 FF 9 f3▼<Би▲ е9 иоя
0000000560: BA 3E 00 E8 2A FF C3 51 8A E0 E8 28 FF 86 C4 B1 ε> и*яГQЬai(ятД±
0000000570: 04 D2 E8 E8 1F FF 59 C3 53 8A FC E8 E9 FF 88 25 ♦Тии▼яYГSЬийяε%
0000000580: 4F 88 05 4F 8A C7 E8 DE FF 88 25 4F 88 05 5B C3 Oε+OЬзиюяε%Oε+[Г
0000000590: 51 52 32 E4 33 D2 B9 0A 00 F7 F1 80 CA 30 88 14 QR2д3TN чсТК0εJ
00000005A0: 4E 33 D2 3D 0A 00 73 F1 3C 00 74 04 0C 30 88 04 N3T= sс< т♦Q0ε♦
00000005B0: 5A 59 C3 B8 20 00 8E D8 E8 EF FE E8 40 FF E8 77 ZYГё Ѓшипиюи@яиw
00000005C0: FF E8 5B FE 32 C0 B4 4C CD 21 яиГя2AГH!

```

Рисунок 8 – HEX-представление «хорошего» .EXE (2)

В отличие от «плохого» EXE, в «хорошем» код, стек и данные выделены в отдельные сегменты. Код программы начинается с 400h, т.к. дополнительно выделено под стек 200 байт (100 слов). Для «хорошего» EXE в директиве org 100h нет необходимости, т.к. загрузчик автоматически расположит программу после PSP.

Результат загрузки COM модуля в основную память представлен на рис. 9.

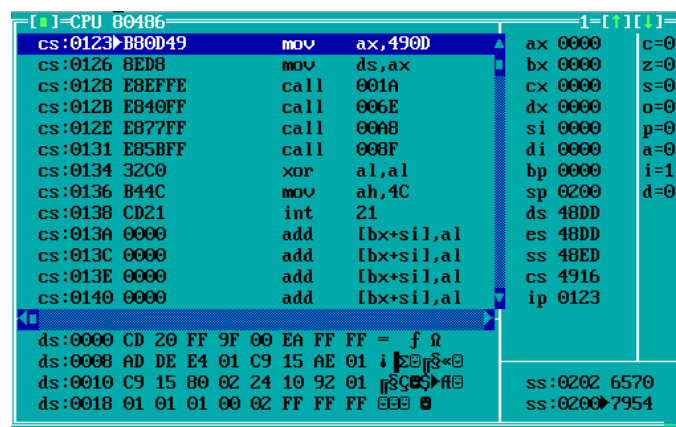


Рисунок 9 – Результат загрузки .COM в основную память

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

Формат загрузки модуля COM:

1. Выделение сегмента памяти для модуля
2. Установка всех сегментных регистров на начало выделенного сегмента памяти
3. Построение в первых 100h байтах памяти PSP
4. Загрузка содержимого COM-файла и присваивание регистру IP значения 100h.
5. Регистр SP устанавливается в конец сегмента

Код начинается с адреса, содержащимся в CS, в нашем случае это 48DD.

2. Что располагается с адреса 0?

С нулевого адреса располагается PSP.

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

Все сегментные регистры (CS, DS, ES, SS) в данном случае равны 48DD и 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=FFFEh – на его конец. Стек может дойти до кода/данных программы при достаточном количестве элементов.

Адреса расположены в диапазоне 0000h-FFFEh. Стек растет от больших адресов к меньшим.

Результат загрузки «хорошего» EXE модуля в основную память представлен на рис. 10.

CPU 80486				1=			
cs:0100	E9AC01	jmp	02AF ↓	ax	0000	c=0	
cs:0103	54	push	sp	bx	0000	z=0	
cs:0104	7970	jns	0176	cx	0000	s=0	
cs:0106	65204F53	and	gs:[bx+53],cl	dx	0000	o=0	
cs:010A	3A20	cmp	ah,[bx+sil]	si	0000	p=0	
cs:010C	2456	and	al,56	di	0000	a=0	
cs:010E	657273	jb	gs:0184	bp	0000	i=1	
cs:0111	696F6E204F	imul	bp,[bx+6E],4F	sp	FFFE	d=0	
cs:0116	53	push	bx	ds	48DD		
cs:0117	3A20	cmp	ah,[bx+sil]	es	48DD		
cs:0119	2020	and	[bx+sil],ah	ss	48DD		
cs:011B	2E2020	and	cs:[bx+sil],ah	cs	48DD		
cs:011E	0D0A24	or	ax,240A	ip	0100		
ds:0000	CD 20 FF 9F 00 EA FF FF	= f 0					
ds:0008	AD DE E4 01 C9 15 AE 01	↓ 20 15 01					
ds:0010	C9 15 80 02 24 10 92 01	↓ 10 92 01					
ds:0018	01 01 01 00 02 FF FF FF	00 02 FF FF					
				ss:0000	20CD		
				ss:FFFE	0000		

Рисунок 10– Результат загрузки «хорошего» .EXE в основную память

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

SS=48DD – начало сегмента стека, CS=48DD – начало сегмента команд.

2. На что указывают регистры DS и ES?

На начало PSP.

3. Как определяется стек?

В исходном коде модуля стек определяется при помощи директивы STACK, а при исполнении в регистр SS записывается адрес начала сегмента стека, а в SP – 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_1.asm

```
TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100H
START: JMP BEGIN
;ДАННЫЕ
OS db 'Type OS: $'
OS_VERS db 'Version OS: . ',0DH,0AH,'$'
OS_OEM db 'OEM: ',0DH,0AH,'$'
SER_NUM db 'Serial number: ','$'
STRING db ' $'
ENDSTR db 0DH,0AH,'$'

PC db 'PC',0DH,0AH,'$'
PCXT db 'PC/XT',0DH,0AH,'$'
_AT db 'AT',0DH,0AH,'$'
PS2_30 db 'PS2 model 30',0DH,0AH,'$'
PS2_80 db 'PS2 model 80',0DH,0AH,'$'
PCjr db 'PCjr',0DH,0AH,'$'
PC_Cnv db 'PC Convertible',0DH,0AH,'$'

Write PROC near
    mov AH,09h
    int 21h
    ret
Write ENDP

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

System PROC near
    mov ax,0F000h
    mov es,ax
    mov ax,es:0FFFEh
    ret
System ENDP

;определение типа ОС
```

```

Write_system PROC near
    mov dx, OFFSET OS
    call Write
    call System

    cmp al,0FFh
    je PC_mtk
    cmp al,0FEh
    je PCXT_mtk
    cmp al,0FBh
    je PCXT_mtk
    cmp al,0FCh
    je AT_mtk
    cmp al,0FAh
    je PS2_30_mtk
    cmp al,0F8h
    je PS2_80_mtk
    cmp al,0FDh
    je PCjr_mtk
    cmp al,0F9h
    je PC_Cnv_mtk

PC_mtk:
    mov dx, OFFSET PC
    jmp to_write
PCXT_mtk:
    mov dx, OFFSET PCXT
    jmp to_write
AT_mtk:
    mov dx, OFFSET _AT
    jmp to_write
PS2_30_mtk:
    mov dx, OFFSET PS2_30
    jmp to_write
PS2_80_mtk:
    mov dx, OFFSET PS2_80
    jmp to_write
PCjr_mtk:
    mov dx, OFFSET PCjr
    jmp to_write
PC_Cnv_mtk:
    mov dx, OFFSET PC_Cnv

to_write:
    call Write
    ret

```

Write_system ENDP

;определение версии системы

Ver_OS PROC near

 ;получение данных

 mov ax,0

 mov ah,30h

 int 21h

 ;запись в строку OS_VERS номер основной версии ОС

 mov si,offset OS_VERS

 add si,12

 push ax

 call BYTE_TO_DEC

 pop ax

 mov al,ah

 add si,3

 call BYTE_TO_DEC

 ;запись версии ОС в консоль

 mov dx,offset OS_VERS

 call Write

 ret

Ver_OS ENDP

;определение оем

Write_oem PROC near

 mov ax,0

 mov ah,30h

 int 21h

 mov si,offset OS_OEM

 add si,7

 mov al,bh

 call BYTE_TO_DEC

 mov dx,offset OS_OEM

 call Write

 ret

Write_oem ENDP

Num_of_serial PROC near

 ;запись серийного номера пользователя

 mov dx,offset SER_NUM

 call Write

 mov al,bl

 call BYTE_TO_HEX

 mov bx,ax

```

        mov dl,b1
        mov ah,02h
        int 21h
        mov dl,bh
        int 21h
        mov di,offset STRING
        add di,3
        mov ax,cx
        call WRD_TO_HEX
        mov dx,offset STRING
        call Write

        mov dx,offset ENDSTR
        call Write

        ret
Num_of_serial 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
        pop CX
        ret
BYTE_TO_HEX ENDP

; перевод в 16с/с 16-ти разрядного числа
; в AX - число, DI - адрес последнего символа
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

; перевод в 10с/с, SI - адрес поля младшей цифры
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

BEGIN:
    call Write_system
    call Ver_OS
    call Num_of_serial
    call Write_oem
    xor AL,AL
    mov AH,4Ch
    int 21H

TESTPC ENDS
END START

```


lr1_2.asm

```
STACK SEGMENT STACK
    DW 0100H DUP(?)
STACK ENDS

DATA SEGMENT
; ДАННЫЕ
OS db 'Type OS: $'
OS_VERS db 'Version OS:  . ',0DH,0AH,$'
OS_OEM db 'OEM: ',0DH,0AH,$'
SER_NUM db 'Serial number: ', '$'
STRING db '    $'
ENDSTR db 0DH,0AH,$'

PC db 'PC',0DH,0AH,$'
PCXT db 'PC/XT',0DH,0AH,$'
_AT db 'AT',0DH,0AH,$'
PS2_30 db 'PS2 model 30',0DH,0AH,$'
PS2_80 db 'PS2 model 80',0DH,0AH,$'
PCjr db 'PCjr',0DH,0AH,$'
PC_Cnv db 'PC Convertible',0DH,0AH,$'
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA, ES:NOTHING, SS:STACK
Write PROC near
    mov AH,09h
    int 21h
    ret
Write ENDP

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

System PROC near
    mov ax,0F000h
    mov es,ax
    mov ax,es:0FFFEh
    ret
System ENDP

Write_system PROC near
    mov dx, OFFSET OS
    call Write
    call System

    cmp al,0FFh
    je PC_mtk
    cmp al,0FEh
    je PCXT_mtk
    cmp al,0FBh
    je PCXT_mtk
    cmp al,0FCh
    je AT_mtk
    cmp al,0FAh
```

```

je PS2_30_mtk
cmp al,0F8h
je PS2_80_mtk
cmp al,0FDh
je PCjr_mtk
cmp al,0F9h
je PC_Cnv_mtk

PC_mtk:
    mov dx, OFFSET PC
    jmp to_write
PCXT_mtk:
    mov dx, OFFSET PCXT
    jmp to_write
AT_mtk:
    mov dx, OFFSET _AT
    jmp to_write
PS2_30_mtk:
    mov dx, OFFSET PS2_30
    jmp to_write
PS2_80_mtk:
    mov dx, OFFSET PS2_80
    jmp to_write
PCjr_mtk:
    mov dx, OFFSET PCjr
    jmp to_write
PC_Cnv_mtk:
    mov dx, OFFSET PC_Cnv

to_write:
    call Write
    ret
Write_system ENDP

Ver_OS PROC near
    mov ax,0
    mov ah,30h
    int 21h

    mov si,offset OS_VERS
    add si,12
    push ax
    call BYTE_TO_DEC

    pop ax
    mov al,ah
    add si,3
    call BYTE_TO_DEC

    mov dx,offset OS_VERS
    call Write
    ret
Ver_OS ENDP

Write_oem PROC near
    mov ax,0
    mov ah,30h
    int 21h

    mov si,offset OS_OEM

```

```

        add si,7
        mov al,bh
        call BYTE_TO_DEC

        mov dx,offset OS_OEM
        call Write
        ret
Write_oem ENDP

Num_of_serial PROC near

        mov dx,offset SER_NUM
        call Write
        mov al,bl
        call BYTE_TO_HEX
        mov bx,ax
        mov dl,bl
        mov ah,02h
        int 21h
        mov dl,bh
        int 21h
        mov di,offset STRING
        add di,3
        mov ax,cx
        call WRD_TO_HEX
        mov dx,offset STRING
        call Write

        mov dx,offset ENDSTR
        call Write

        ret
Num_of_serial 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
        pop CX
        ret
BYTE_TO_HEX ENDP

; перевод в 16с/с 16-ти разрядного числа
; в AX - число, DI - адрес последнего символа
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

; перевод в 10с/с, SI - адрес поля младшей цифры
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

BEGIN:
        mov ax,DATA
        mov ds,ax
        call Write_system
        call Ver_OS
        call Num_of_serial
        call Write_oem
        xor AL,AL
        mov AH,4Ch
        int 21H
CODE ENDS
END BEGIN

```