

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ
по лабораторной работе №1
по дисциплине «Операционные системы»
Тема: Исследование структур загрузочных модулей

Студент гр. 8383

Колмыков В.Д.

Преподаватель

Губкин А.Ф.

Санкт-Петербург

2020

Цель работы.

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

Процедуры, используемые в работе.

Название процедуры	Описание процедуры
OS_VER_PROC	Процедура распознавания и вывода версии ОС
PC_TYPE_PROC	Процедура распознавания и вывода типа ПК
WRITE	Процедура вывода содержимого по смещению DX
WRITE_OS_VERSION	Процедура вывода версии ОС
WRITE_DEC	Процедура вывода слова AX в 10-иричной с.с.
WRITE_SERIAL	Процедура вывода серийного номера пользователя
WRITE_HEX_BYTE	Процедура вывода байта AL в 16-иричной с.с.
WRITE_DEC_BYTE	Процедура вывода байта AL в 10-иричной с.с. в формате, требуемым заданием

Ход работы.

Был написан текст исходного **.COM** модуля, который определяет тип ПК и версию системы. Код программы представлен в приложении А. После написания текста, из него были построены **.COM** модуль (результат выполнения которого показан на рис. 1) а также «плохой» **.EXE** модуль

(результат выполнения показан на рис. 2). В ходе линковки **.EXE** модуля LINK-ер выдал предупреждение об отсутствии стека (см. рис. 3).

```
C:\>tasm LAB1_COM.ASM,,,;
Turbo Assembler Version 3.1 Copyright (c) 1988, 1992 Borland International

Assembling file: LAB1_COM.ASM
Error messages: None
Warning messages: None
Passes: 1
Remaining memory: 461k

C:\>tlink /t LAB1_COM.OBJ,,,;
Turbo Link Version 5.1 Copyright (c) 1992 Borland International

C:\>LAB1_COM.COM
AT
OS version is 05.00
OEM is 255
Serial number is 000000
C:\>
```

Рисунок 1 – Результат выполнения **.COM** модуля

```
C:\>LAB1_COM.EXE

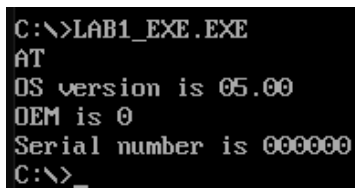
OS version is 05.00
OEM is 255
Serial number is 000000
C:\>
```

Рисунок 2 – Результат выполнения «плохого» **.EXE** модуля

```
C:\>tlink LAB1_COM.OBJ,,,;
Turbo Link Version 5.1 Copyright (c) 1992 Borland International
Warning: No stack
```

Рисунок 3 – Предупреждение во время линковки

Был написан текст для «хорошего» **.EXE** модуля. Код программы представлен в приложении Б. После написания текста, из него был построен **.EXE** модуль (результат выполнения которого показан на рис. 4).



```
C:\>LAB1_EXE.EXE
AT
OS version is 05.00
OEM is 0
Serial number is 000000
C:\>_
```

Рисунок 4 – Результат выполнения «хорошего» **.EXE** модуля

Отличия исходных текстов COM и EXE программ

- 1) COM программа должна содержать ровно один сегмент.
- 2) EXE программа может содержать сколько угодно сегментов (не меньше одного)
- 3) COM программа должна начинаться с директивы `ORG 100h`, устанавливающая значение `IP 100h`, так как первые 256 байт занимает PSP, код программы расположен после него. В EXE программе PSP расположен вне сегмента кода, и указывать смещение не нужно.
- 4) В COM программах нельзя использовать команды вида `mov <регистр> <сегмент>`, так как в этих программах используется только один сегмент и надобности в этих командах нет. Также запрещены 64-битные команды.

При помощи приложения Far были открыты все созданные файлы загрузочных модулей в шестнадцатеричном виде. Результаты представлены на рис. 5 – 7.

```

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: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000310: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000320: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000330: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000340: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000350: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000360: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000370: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000380: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000390: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000003A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000003B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000003C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000003D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000003E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000003F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000400: 0D 0A 4F 53 20 76 65 72 73 69 6F 6E 20 69 73 20
0000000410: 24 2E 24 0D 0A 4F 45 4D 20 69 73 20 24 0D 0A 53
0000000420: 65 72 69 61 6C 20 6E 75 6D 62 65 72 20 69 73 20
0000000430: 24 0D 0A 50 43 20 74 79 70 65 20 69 73 20 24 50
0000000440: 43 24 50 43 2F 58 54 24 41 54 24 50 53 32 20 6D
0000000450: 6F 64 65 6C 20 33 30 24 50 53 32 20 6D 6F 64 65
0000000460: 6C 20 35 30 20 6F 72 20 36 30 24 50 53 32 20 6D
0000000470: 6F 64 65 6C 20 38 30 24 50 43 6A 72 24 50 43 20
0000000480: 43 6F 6E 76 65 72 74 69 62 6C 65 24 75 6E 6B 6E
0000000490: 6F 77 6E 20 24 00 00 00 00 00 00 00 00 00 00 00
00000004A0: E9 2A 01 52 50 B4 30 CD 21 BA 00 00 E8 92 00 E8
00000004B0: 96 00 BA 13 00 E8 89 00 BA 13 00 8A C7 B4 00 E8
00000004C0: 9D 00 BA 1D 00 E8 79 00 E8 B6 00 58 5A C3 50 06
00000004D0: 52 B8 00 F0 8E C0 26 A0 FE FF 3C FF 74 2C 3C FE
00000004E0: 74 2E 3C FB 74 2A 3C FC 74 2C 3C FA 74 2E 3C FC
00000004F0: 74 30 3C F8 74 32 3C FD 74 34 3C F9 74 36 BA 8C
0000000500: 00 E8 3D 00 E8 92 00 E8 34 90 BA 3F 00 E8 2B 90
0000000510: BA 42 00 E8 25 90 BA 48 00 E8 1F 90 BA 4B 00 E8
0000000520: 19 90 BA 58 00 E8 13 90 BA 6B 00 E8 0D 90 BA 78
0000000530: 00 E8 07 90 BA 7D 00 E8 01 90 E8 04 00 5A 07 58
0000000540: C3 50 B4 09 CD 21 58 C3 53 52 50 B3 0A E8 66 00
0000000550: BA 11 00 E8 E8 FF 8A C4 E8 5B 00 58 5A 5B C3 50
0000000560: 51 52 53 B8 0A 00 33 C9 33 D2 F7 F3 52 41 85 C0
0000000570: 75 F6 B4 02 5A 80 C2 30 CD 21 E2 F8 5B 5A 59 58
0000000580: C3 50 53 51 52 8A C3 E8 0F 00 8A C5 E8 0A 00 8A
0000000590: C1 E8 05 00 5A 59 5B 58 C3 50 53 52 B4 00 B3 10
00000005A0: F6 F3 8B D0 B4 02 80 C2 30 CD 21 8A D6 80 C2
00000005B0: CD 21 5A 5B 58 C3 50 B4 00 F6 F3 8B D0 B4 02 80
00000005C0: C2 30 CD 21 8A D6 80 C2 30 CD 21 58 C3 1E 2B C0
00000005D0: 50 B8 20 00 8E D8 E8 F5 FE E8 C7 FE 32 C0 B4 4C
00000005E0: CD 21

```

```

OS version is
$. OEM is $S
erial number is
$PC type is $P
C$PC/XT$AT$PS2 m
odel 30$PS2 mode
l 50 or 60$PS2 m
odel 80$PCjr$PC
Convertible$unkn
own $
й*0RP_0H!е и' и
- е!! и% е!! _3_ и
_е+ иу и! XZГР+
Rè p_A& юя<ят,<ю
т.<ьт*<ьт,<ьт.<ь
т0<шт2<эт4<шт6е_
и= и' л4_е? л+_
еВ л%_еН л%_еК л
↓_еХ л!!_еК л%_еХ
л*_е} л0_и+ Z+X
ГР_оН!ХГSRP_ииф
е+ иля_ди[ XZ[ГР
QRS>Э ЗЙЗТчуРА:А
иц_0Z_В0Н!вш[ZYX
ГПСQR_Гю_Еи_
Би+ ZY[ХГPSR_ _
цу<Р_0_В0Н!_Ц_В0
Н!Z[ХГР_цу<Р_0_
В0Н!_Ц_В0Н!ХГ+А
Рё _ШихюиЗю2А_L
Н!

```

Рисунок 5 – Содержимое файла «хорошего» EXE модуля

0000000000: E9 BF 01 0D 0A 4F 53 20	76 65 72 73 69 6F 6E 20	йi0)OS version
0000000010: 69 73 20 24 2E 24 0D 0A	4F 45 4D 20 69 73 20 24	is \$.\$.OEM is \$
0000000020: 0D 0A 53 65 72 69 61 6C	20 6E 75 6D 62 65 72 20)Serial number
0000000030: 69 73 20 24 0D 0A 50 43	20 74 79 70 65 20 69 73	is \$)PC type is
0000000040: 20 24 50 43 24 50 43 2F	58 54 24 41 54 24 50 53	\$PC\$PC/XT\$AT\$PS
0000000050: 32 20 6D 6F 64 65 6C 20	33 30 24 50 53 32 20 6D	2 model 30\$PS2 m
0000000060: 6F 64 65 6C 20 35 30 20	6F 72 20 36 30 24 50 53	odel 50 or 60\$PS
0000000070: 32 20 6D 6F 64 65 6C 20	38 30 24 50 43 6A 72 24	2 model 80\$PCjr\$
0000000080: 50 43 20 43 6F 6E 76 65	72 74 69 62 6C 65 24 75	PC Convertible\$u
0000000090: 6E 6B 6E 6F 77 6E 20 24	52 50 B4 30 CD 21 BA 03	nknown \$RP_0H!e▼
00000000A0: 01 E8 92 00 E8 96 00 BA	16 01 E8 89 00 BA 16 01	0и' и- eи% eи
00000000B0: 8A C7 B4 00 E8 9D 00 BA	20 01 E8 79 00 E8 B6 00	_з_и_ e иу и
00000000C0: 58 5A C3 50 06 52 B8 00	F0 8E C0 26 A0 FE FF 3C	XZГРRè p_A& юя<
00000000D0: FF 74 2C 3C FE 74 2E 3C	FB 74 2A 3C FC 74 2C 3C	ят,<ют.<ът*<ът,<
00000000E0: FA 74 2E 3C FC 74 30 3C	F8 74 32 3C FD 74 34 3C	ът.<ът0<шт2<эт4<
00000000F0: F9 74 36 BA 8F 01 E8 3D	00 E8 92 00 EB 34 90 BA	шт6e_0и= и' л4_e
0000000100: 42 01 EB 2B 90 BA 45 01	EB 25 90 BA 4B 01 EB 1F	В0л+_eE0л%_eK0л▼
0000000110: 90 BA 4E 01 EB 19 90 BA	5B 01 EB 13 90 BA 6E 01	_eN0л↓_e[0л!!_ен0
0000000120: EB 0D 90 BA 7B 01 EB 07	90 BA 80 01 EB 01 90 E8	л↓_e{0л*_e_0л0_и
0000000130: 04 00 5A 07 58 C3 50 B4	09 CD 21 58 C3 53 52 50	♦ Z*ХГР_оН!ХГSRP
0000000140: B3 0A E8 66 00 BA 14 01	E8 EB FF 8A C4 E8 5B 00	_иф eи0иля_ди[
0000000150: 58 5A 5B C3 50 51 52 53	BB 0A 00 33 C9 33 D2 F7	XZ[ГPQRS>З ЗЙЗТч
0000000160: F3 52 41 85 C0 75 F6 B4	02 5A 80 C2 30 CD 21 E2	yRA:Ауц_0Z_В0H!в
0000000170: F8 5B 5A 59 58 C3 50 53	51 52 8A C3 E8 0F 00 8A	ш[ZYXГP\$QR_Гио_
0000000180: C5 E8 0A 00 8A C1 E8 05	00 5A 59 5B 58 C3 50 53	Еи_Би+ ZY[ХГP\$
0000000190: 52 B4 00 B3 10 F6 F3 8B	D0 B4 02 80 C2 30 CD 21	R_►цу<P_0_В0H!
00000001A0: 8A D6 80 C2 30 CD 21 5A	5B 58 C3 50 B4 00 F6 F3	_ц_В0H!Z[ХГР_ цу
00000001B0: 8B D0 B4 02 80 C2 30 CD	21 8A D6 80 C2 30 CD 21	<P_0_В0H!_ц_В0H!
00000001C0: 58 C3 E8 FE FE E8 D0 FE	32 C0 B4 4C CD 21	ХГиюиРю2A_LH!

Рисунок 6 – Содержисое файла СОМ модуля

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 BF 01 0D 0A 4F 53 20	76 65 72 73 69 6F 6E 20	йi0)OS version
0000000310: 69 73 20 24 2E 24 0D 0A	4F 45 4D 20 69 73 20 24	is \$.\$.OEM is \$
0000000320: 0D 0A 53 65 72 69 61 6C	20 6E 75 6D 62 65 72 20)Serial number
0000000330: 69 73 20 24 0D 0A 50 43	20 74 79 70 65 20 69 73	is \$)PC type is
0000000340: 20 24 50 43 24 50 43 2F	58 54 24 41 54 24 50 53	\$PC\$PC/XT\$AT\$PS
0000000350: 32 20 6D 6F 64 65 6C 20	33 30 24 50 53 32 20 6D	2 model 30\$PS2 m
0000000360: 6F 64 65 6C 20 35 30 20	6F 72 20 36 30 24 50 53	odel 50 or 60\$PS
0000000370: 32 20 6D 6F 64 65 6C 20	38 30 24 50 43 6A 72 24	2 model 80\$PCjr\$
0000000380: 50 43 20 43 6F 6E 76 65	72 74 69 62 6C 65 24 75	PC Convertible\$u
0000000390: 6E 6B 6E 6F 77 6E 20 24	52 50 B4 30 CD 21 BA 03	nknown \$RP_0H!e▼
00000003A0: 01 E8 92 00 E8 96 00 BA	16 01 E8 89 00 BA 16 01	0и' и- eи% eи
00000003B0: 8A C7 B4 00 E8 9D 00 BA	20 01 E8 79 00 E8 B6 00	_з_и_ e иу и
00000003C0: 58 5A C3 50 06 52 B8 00	F0 8E C0 26 A0 FE FF 3C	XZГРRè p_A& юя<
00000003D0: FF 74 2C 3C FE 74 2E 3C	FB 74 2A 3C FC 74 2C 3C	ят,<ют.<ът*<ът,<
00000003E0: FA 74 2E 3C FC 74 30 3C	F8 74 32 3C FD 74 34 3C	ът.<ът0<шт2<эт4<
00000003F0: F9 74 36 BA 8F 01 E8 3D	00 E8 92 00 EB 34 90 BA	шт6e_0и= и' л4_e
0000000400: 42 01 EB 2B 90 BA 45 01	EB 25 90 BA 4B 01 EB 1F	В0л+_eE0л%_eK0л▼
0000000410: 90 BA 4E 01 EB 19 90 BA	5B 01 EB 13 90 BA 6E 01	_eN0л↓_e[0л!!_ен0
0000000420: EB 0D 90 BA 7B 01 EB 07	90 BA 80 01 EB 01 90 E8	л↓_e{0л*_e_0л0_и
0000000430: 04 00 5A 07 58 C3 50 B4	09 CD 21 58 C3 53 52 50	♦ Z*ХГР_оН!ХГSRP
0000000440: B3 0A E8 66 00 BA 14 01	E8 EB FF 8A C4 E8 5B 00	_иф eи0иля_ди[
0000000450: 58 5A 5B C3 50 51 52 53	BB 0A 00 33 C9 33 D2 F7	XZ[ГPQRS>З ЗЙЗТч
0000000460: F3 52 41 85 C0 75 F6 B4	02 5A 80 C2 30 CD 21 E2	yRA:Ауц_0Z_В0H!в
0000000470: F8 5B 5A 59 58 C3 50 53	51 52 8A C3 E8 0F 00 8A	ш[ZYXГP\$QR_Гио_
0000000480: C5 E8 0A 00 8A C1 E8 05	00 5A 59 5B 58 C3 50 53	Еи_Би+ ZY[ХГP\$
0000000490: 52 B4 00 B3 10 F6 F3 8B	D0 B4 02 80 C2 30 CD 21	R_►цу<P_0_В0H!
00000004A0: 8A D6 80 C2 30 CD 21 5A	5B 58 C3 50 B4 00 F6 F3	_ц_В0H!Z[ХГР_ цу
00000004B0: 8B D0 B4 02 80 C2 30 CD	21 8A D6 80 C2 30 CD 21	<P_0_В0H!_ц_В0H!
00000004C0: 58 C3 E8 FE FE E8 D0 FE	32 C0 B4 4C CD 21	ХГиюиРю2A_LH!

Рисунок 7 – Содержимое файла «плохого» EXE модуля

Отличие форматов файлов COM и EXE модулей

- 1) COM файл содержит только машинный код и данные программы. Код начинается с адреса 0h, но при загрузке происходит смещение на 100h.
- 2) В «плохом» EXE модуле машинный код и данные содержатся в одном сегменте. Код начинается со смещения 300h. До машинного кода идет таблица настроек.
- 3) В «хорошем» EXE модуле данные, стек и машинный код находятся в разных сегментах. От «плохого» EXE он так же отличается тем, что мы можем наблюдать выделенный стек.

При помощи отладчика TD COM файл был загружен в основную память. Результат продемонстрирован на рис. 8.

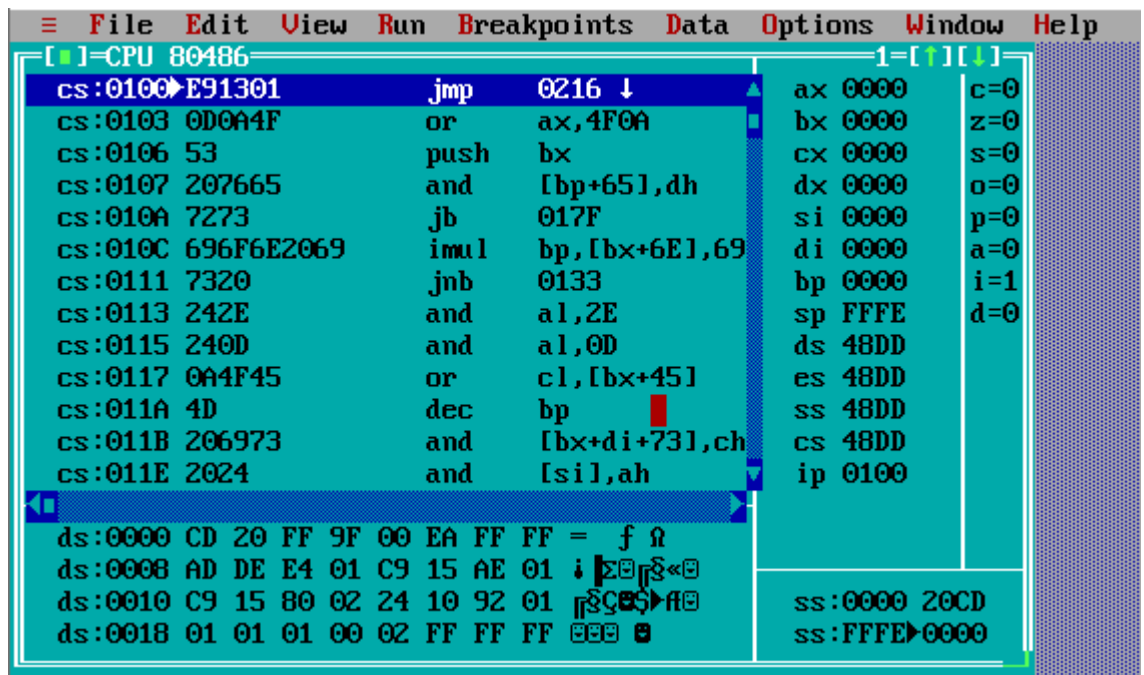


Рисунок 8 – Результат загрузки COM файла в память

Загрузка COM модуля в основную память

- 1) Определяется сегментный адрес свободного участка ОП, в который можно загрузить программу. Создается блок памяти. В поля PSP заносятся значения. Загружается COM файл со смещением 100h. Сегментные регистры устанавливаются на адрес сегмента PSP, регистр SP указывает на конец сегмента, туда записывается 0000h. С ростом

стека значение SP будет уменьшаться. Счетчик команд принимает значение 100h. Программа запускается.

- 2) Со смещения 0h в программном сегменте начинается PSP.
- 3) Сегментные регистры указывают на адрес сегмента PSP.
- 4) Стек занимает все доступное место (из 64кб) после PSP и машинного кода. Регистр SP изначально указывает на адрес FFFeh.

При помощи отладчика в основную память был записан так же «хороший» EXE файл. Результат продемонстрирован на рис. 9.

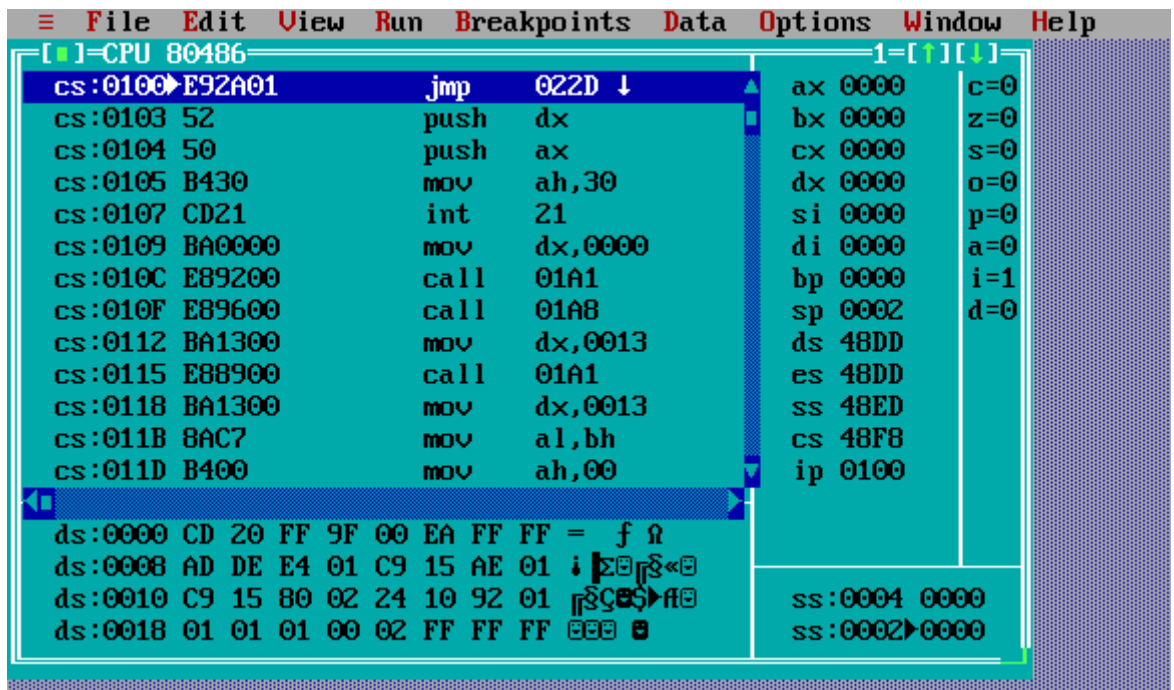


Рисунок 9 – Результат загрузки EXE файла в память

Загрузка «хорошего» EXE модуля в основную память

- 1) Определяется сегментный адрес свободного участка ОП, в который можно загрузить программу. Создается блок памяти для PSP и программы. В PSP заносятся соответствующие значения. В рабочую область загрузчика считывается форматированная часть заголовка файла. Определяется смещение начала загрузочного модуля в EXE файле. Вычисляется сегментный адрес (START_SEG) для загрузки. В память считывается загрузочный модуль. Таблица настройки порциями считывается в рабочую память. Для каждого элемента

таблицы настройки к полю сегмента прибавляется сегментный адрес начального сегмента (в результате элемент таблицы указывает на нужное слово в памяти). В регистры ES и DS записывается сегментный адрес начала PSP. Регистр CS указывает на начало сегмента кода, SP – на начало стека (значение из заголовка). Управление передается загруженной задаче по адресу из заголовка.

- 2) Регистры ES и DS указывают на начало PSP.
- 3) Стек определяется при помощи директивы ASSUME CS:MYSTACK, где MYSTACK – сегмент, отведенный под стек:

```
MYSTACK SEGMENT STACK
```

```
    DW 100H
```

```
MYSTACK ENDS
```

- 4) Точка входа определяется при помощи директивы END <метка для входа>

Выводы.

В ходе выполнения 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.

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ ДЛЯ СОМ ФАЙЛА

```
LAB1  SEGMENT
        ASSUME  CS:LAB1, DS:LAB1, ES:NOTHING, SS:NOTHING
        ORG 100H

START: JMP BEGIN

;DATA
OS_VERSION_NUMBER DB 13, 10, "OS VERSION IS $"
DOT DB ".$"
OEM DB 13, 10, "OEM IS $"
SERIAL DB 13, 10, "SERIAL NUMBER IS $"
TYPE_PC_STR DB 13, 10, "PC TYPE IS $"
TYPE_PC DB "PC$"
TYPE_PC_XT DB "PC/XT$"
TYPE_AT DB "AT$"
TYPE_PS2_M30 DB "PS2 MODEL 30$"
TYPE_PS2_M5060 DB "PS2 MODEL 50 OR 60$"
TYPE_PS2M80 DB "PS2 MODEL 80$"
TYPE_PC_JR DB "PCJR$"
TYPE_PC_CONV DB "PC CONVERTIBLE$"
TYPE_UNKNOWN DB "UNKNOWN $"

;PROCEDURES
;_____
OS_VER_PROC PROC
        PUSH DX
        PUSH AX
        MOV AH, 30H ;GETTING INFO
        INT 21H

        MOV DX, OFFSET OS_VERSION_NUMBER
        CALL WRITE
        CALL WRITE_OS_VERSION
        MOV DX, OFFSET OEM
        CALL WRITE
        MOV DX, OFFSET OEM
        MOV AL, BH
        MOV AH, 0
        CALL WRITE_DEC
        MOV DX, OFFSET SERIAL
        CALL WRITE
```

```

        CALL WRITE_SERIAL

        POP AX
        POP DX
        RET
OS_VER_PROC ENDP
;_____
PC_TYPE_PROC PROC
        PUSH AX
        PUSH ES
        PUSH DX

        MOV AX, 0F000H
        MOV ES, AX
        MOV AL, ES:[0FFFEH]
        CMP AL, 0FFH
        JE PC
        CMP AL, 0FEH
        JE XT
        CMP AL, 0FBH
        JE XT
        CMP AL, 0FCH
        JE PCAT
        CMP AL, 0FAH
        JE PS30
        CMP AL, 0FCH;ОПЕЧАТКА В МЕТОДЕ
        JE PS50
        CMP AL, 0F8H
        JE PS80
        CMP AL, 0FDH
        JE JR
        CMP AL, 0F9H
        JE CONV
        ;DEFAULT
        MOV DX, OFFSET TYPE_UNKNOWN
        CALL WRITE
        CALL WRITE_HEX_BYTE
        JMP FINISH
PC:
        MOV DX, OFFSET TYPE_PC
        JMP RES

```

```

XT:
    MOV DX, OFFSET TYPE_PC_XT
    JMP RES

PCAT:
    MOV DX, OFFSET TYPE_AT
    JMP RES

PS30:
    MOV DX, OFFSET TYPE_PS2_M30
    JMP RES

PS50:
    MOV DX, OFFSET TYPE_PS2_M5060
    JMP RES

PS80:
    MOV DX, OFFSET TYPE_PS2M80
    JMP RES

JR:
    MOV DX, OFFSET TYPE_PC_JR
    JMP RES

CONV:
    MOV DX, OFFSET TYPE_PC_CONV
    JMP RES

RES:
    CALL WRITE

FINISH:
    POP DX
    POP ES
    POP AX
    RET

PC_TYPE_PROC ENDP

; _____

WRITE PROC
    PUSH AX
    MOV AH, 9H
    INT 21H
    POP AX
    RET

WRITE ENDP

; _____

WRITE_OS_VERSION PROC
    PUSH BX

```

```

        PUSH DX
        PUSH AX

        MOV BL, 10
        ;VER
        CALL WRITE_DEC_BYTE
        ;DOT
        MOV DX, OFFSET DOT
        CALL WRITE
        ;MOD
        MOV AL, AH
        CALL WRITE_DEC_BYTE

        POP AX
        POP DX
        POP BX
        RET
WRITE_OS_VERSION ENDP
; _____
WRITE_DEC PROC
        PUSH AX
        PUSH CX
        PUSH DX
        PUSH BX

        MOV BX, 10
        XOR CX, CX
GETTING_NUMS:
        XOR DX, DX
        DIV BX
        PUSH DX
        INC CX
        TEST AX, AX
        JNZ GETTING_NUMS
        MOV AH, 02H
WRITING:
        POP DX
        ADD DL, '0'
        INT 21H
        LOOP WRITING

```

```

        POP BX
        POP DX
        POP CX
        POP AX
        RET
WRITE_DEC ENDP
; _____
WRITE_SERIAL PROC
        PUSH AX
        PUSH BX
        PUSH CX
        PUSH DX

        ;FIRST BYTE
        MOV AL, BL
        CALL WRITE_HEX_BYTE
        ;SECOND BYTE
        MOV AL, CH
        CALL WRITE_HEX_BYTE
        ;THIRD BYTE
        MOV AL, CL
        CALL WRITE_HEX_BYTE

        POP DX
        POP CX
        POP BX
        POP AX
        RET
WRITE_SERIAL ENDP
; _____
WRITE_HEX_BYTE PROC
        PUSH AX
        PUSH BX
        PUSH DX

        MOV AH, 0
        MOV BL, 16
        DIV BL
        MOV DX, AX
        MOV AH, 02H
        ADD DL, '0'

```

```

        INT 21H;
        MOV DL, DH
        ADD DL, '0'
        INT 21H;

        POP DX
        POP BX
        POP AX
        RET
WRITE_HEX_BYTE ENDP
; _____

WRITE_DEC_BYTE PROC
        PUSH AX
        MOV AH, 0
        DIV BL
        MOV DX, AX
        MOV AH, 02H
        ADD DL, '0'
        INT 21H
        MOV DL, DH
        ADD DL, '0'
        INT 21H
        POP AX
        RET
WRITE_DEC_BYTE ENDP
; _____

BEGIN:
        CALL PC_TYPE_PROC
        CALL OS_VER_PROC

        ;TO DOS
        XOR AL, AL
        MOV AH, 4CH
        INT 21H
LAB1 ENDS
END START

```

ПРИЛОЖЕНИЕ Б

КОД ПРОГРАММЫ ДЛЯ EXE ФАЙЛА

MYSTACK SEGMENT STACK

DW 100H

MYSTACK ENDS

DATA SEGMENT

;DATA

OS_VERSION_NUMBER DB 13, 10, "OS VERSION IS \$"

DOT DB ".\$"

OEM DB 13, 10, "OEM IS \$"

SERIAL DB 13, 10, "SERIAL NUMBER IS \$"

TYPE_PC_STR DB 13, 10, "PC TYPE IS \$"

TYPE_PC DB "PC\$"

TYPE_PC_XT DB "PC/XT\$"

TYPE_AT DB "AT\$"

TYPE_PS2_M30 DB "PS2 MODEL 30\$"

TYPE_PS2_M5060 DB "PS2 MODEL 50 OR 60\$"

TYPE_PS2M80 DB "PS2 MODEL 80\$"

TYPE_PC_JR DB "PCJR\$"

TYPE_PC_CONV DB "PC CONVERTIBLE\$"

TYPE_UNKNOWN DB "UNKNOWN \$"

DATA ENDS

LAB1 SEGMENT

ASSUME CS:LAB1, DS:DATA, ES:NOTHING, SS:MYSTACK

START: JMP BEGIN

;PROCEDURES

; _____

OS_VER_PROC PROC

PUSH DX

PUSH AX

MOV AH, 30H ;GETTING INFO

INT 21H

MOV DX, OFFSET OS_VERSION_NUMBER

CALL WRITE

CALL WRITE_OS_VERSION

MOV DX, OFFSET OEM


```

CALL WRITE
MOV DX, OFFSET OEM
MOV AL, BH
MOV AH, 0
CALL WRITE_DEC
MOV DX, OFFSET SERIAL
CALL WRITE
CALL WRITE_SERIAL

POP AX
POP DX
RET
OS_VER_PROC ENDP
; _____
PC_TYPE_PROC PROC
    PUSH AX
    PUSH ES
    PUSH DX

    MOV AX, 0F000H
    MOV ES, AX
    MOV AL, ES:[0FFFEH]
    CMP AL, 0FFH
    JE PC
    CMP AL, 0FEH
    JE XT
    CMP AL, 0FBH
    JE XT
    CMP AL, 0FCH
    JE PCAT
    CMP AL, 0FAH
    JE PS30
    CMP AL, 0FCH;ОПЕЧАТКА В МЕТОДЕ
    JE PS50
    CMP AL, 0F8H
    JE PS80
    CMP AL, 0FDH
    JE JR
    CMP AL, 0F9H
    JE CONV
    ;DEFAULT

```

```

        MOV DX, OFFSET TYPE_UNKNOWN
        CALL WRITE
        CALL WRITE_HEX_BYTE
        JMP FINISH

PC:
        MOV DX, OFFSET TYPE_PC
        JMP RES

XT:
        MOV DX, OFFSET TYPE_PC_XT
        JMP RES

PCAT:
        MOV DX, OFFSET TYPE_AT
        JMP RES

PS30:
        MOV DX, OFFSET TYPE_PS2_M30
        JMP RES

PS50:
        MOV DX, OFFSET TYPE_PS2_M5060
        JMP RES

PS80:
        MOV DX, OFFSET TYPE_PS2M80
        JMP RES

JR:
        MOV DX, OFFSET TYPE_PC_JR
        JMP RES

CONV:
        MOV DX, OFFSET TYPE_PC_CONV
        JMP RES

RES:
        CALL WRITE

FINISH:
        POP DX
        POP ES
        POP AX
        RET

PC_TYPE_PROC ENDP
; _____

WRITE PROC
        PUSH AX
        MOV AH, 9H

```

```

        INT 21H
        POP AX
        RET
WRITE ENDP
; _____
WRITE_OS_VERSION PROC
        PUSH BX
        PUSH DX
        PUSH AX

        MOV BL, 10
        ;VER
        CALL WRITE_DEC_BYTE
        ;DOT
        MOV DX, OFFSET DOT
        CALL WRITE
        ;MOD
        MOV AL, AH
        CALL WRITE_DEC_BYTE

        POP AX
        POP DX
        POP BX
        RET
WRITE_OS_VERSION ENDP
; _____
WRITE_DEC PROC
        PUSH AX
        PUSH CX
        PUSH DX
        PUSH BX

        MOV BX, 10
        XOR CX, CX
GETTING_NUMS:
        XOR DX, DX
        DIV BX
        PUSH DX
        INC CX
        TEST AX, AX
        JNZ GETTING_NUMS

```

```

        MOV AH, 02H
WRITING:
        POP DX
        ADD DL, '0'
        INT 21H
        LOOP WRITING

```

```

        POP BX
        POP DX
        POP CX
        POP AX
        RET

```

```

WRITE_DEC ENDP

```

```

; _____

```

```

WRITE_SERIAL PROC

```

```

        PUSH AX
        PUSH BX
        PUSH CX
        PUSH DX

```

```

        ;FIRST BYTE
        MOV AL, BL
        CALL WRITE_HEX_BYTE
        ;SECOND BYTE
        MOV AL, CH
        CALL WRITE_HEX_BYTE
        ;THIRD BYTE
        MOV AL, CL
        CALL WRITE_HEX_BYTE

```

```

        POP DX
        POP CX
        POP BX
        POP AX
        RET

```

```

WRITE_SERIAL ENDP

```

```

; _____

```

```

WRITE_HEX_BYTE PROC

```

```

        PUSH AX
        PUSH BX
        PUSH DX

```

```

        MOV AH, 0
        MOV BL, 16
        DIV BL
        MOV DX, AX
        MOV AH, 02H
        ADD DL, '0'
        INT 21H;
        MOV DL, DH
        ADD DL, '0'
        INT 21H;

        POP DX
        POP BX
        POP AX
        RET

```

```
WRITE_HEX_BYTE ENDP
```

```
; _____
```

```
WRITE_DEC_BYTE PROC
```

```

        PUSH AX
        MOV AH, 0
        DIV BL
        MOV DX, AX
        MOV AH, 02H
        ADD DL, '0'
        INT 21H
        MOV DL, DH
        ADD DL, '0'
        INT 21H
        POP AX
        RET

```

```
WRITE_DEC_BYTE ENDP
```

```
; _____
```

```
BEGIN:
```

```

        PUSH DS
        SUB  AX,AX
        PUSH AX
        MOV  AX,DATA
        MOV  DS,AX

```

```
CALL PC_TYPE_PROC
CALL OS_VER_PROC

;TO DOS
XOR AL, AL
MOV AH, 4CH
INT 21H
LAB1 ENDS
END START
```