

**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. 8381

\_\_\_\_\_

Нгуен Ш. Х.

Преподаватель

\_\_\_\_\_

Ефмиров М. А.

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

Таблица 1 - Процедуры, используемые в работе

### **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:\ASM>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: 471k

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

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

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

```
C:\ASM>LAB1_COM.EXE

OS version is 05.00
OS version is 05
OS version is 00
OS version is 255
OS version is 000000
```

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

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

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

### **Отличия исходных текстов COM и EXE программ**

#### **1) Сколько сегментов должна содержать COM-программа?**

COM программа должна содержать ровно один сегмент.

#### **2) EXE-программа?**

EXE программа может содержать сколько угодно сегментов (не меньше одного).

#### **3) Какие директивы должны обязательно быть в тексте COM-программы?**

Обязательная должна быть директива **ORG 100h**. Она нужна по той причине, что при загрузке модуля в ОП в начале COM-программы определяется 256-байтовый (100h) префикс программного сегмента, так что адресация имеет смещение в 256 байт от нулевого адреса. Это смещение мы и задаём директивой **ORG 100h** (в отличие от EXE-программы, где PSP расположен вне кодового сегмента и, следовательно, явно определять смещение там не нужно).

#### **4) Все ли форматы команд можно использовать в COM-программе?**

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.

```
D:\asm\LAB1_EXE.EXE
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 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000210: 0D 0A 4F 53 20 76 65 72 73 69 6F 6E 20 69 73 20
0000000220: 24 2E 24 0D 0A 4F 45 4D 20 69 73 20 24 0D 0A 53
0000000230: 65 72 69 61 6C 20 6E 75 6D 62 65 72 20 69 73 20
0000000240: 24 0D 0A 50 43 20 74 79 70 65 20 69 73 20 24 50
0000000250: 43 24 50 43 2F 58 54 24 41 54 24 50 53 32 20 6D
0000000260: 6F 64 65 6C 20 33 30 24 50 53 32 20 6D 6F 64 65
0000000270: 6C 20 35 30 20 6F 72 20 36 30 24 50 53 32 20 6D
0000000280: 6F 64 65 6C 20 38 30 24 50 43 6A 72 24 50 43 20
0000000290: 43 6F 6E 76 65 72 74 69 62 6C 65 24 75 6E 6B 6E
00000002A0: 6F 77 6E 20 24 00 00 00 00 00 00 00 00 00 00 00
00000002B0: E9 2A 01 52 50 B4 30 CD 21 BA 00 00 E8 92 00 E8
00000002C0: 96 00 BA 13 00 E8 89 00 BA 13 00 8A C7 B4 00 E8
00000002D0: 9D 00 BA 1D 00 E8 79 00 E8 B6 00 58 5A C3 50 06
00000002E0: 52 B8 00 F0 8E C0 26 A0 FE FF 3C FF 74 2C 3C FE
00000002F0: 74 2E 3C FB 74 2A 3C FC 74 2C 3C FA 74 2E 3C FC
0000000300: 74 30 3C F8 74 32 3C FD 74 34 3C F9 74 36 BA 8C
0000000310: 00 E8 3D 00 E8 92 00 EB 34 90 BA 3F 00 EB 2B 90
0000000320: BA 42 00 EB 25 90 BA 48 00 EB 1F 90 BA 4B 00 EB
0000000330: 19 90 BA 58 00 EB 13 90 BA 6B 00 EB 0D 90 BA 78
0000000340: 00 EB 07 90 BA 7D 00 EB 01 90 E8 04 00 5A 07 58
0000000350: C3 50 B4 09 CD 21 58 C3 53 52 50 B3 0A E8 66 00
0000000360: BA 11 00 E8 EB FF 8A C4 E8 5B 00 58 5A 5B C3 50
0000000370: 51 52 53 BB 0A 00 33 C9 33 D2 F7 F3 52 41 85 C0
0000000380: 75 F6 B4 02 5A 80 C2 30 CD 21 E2 F8 5B 5A 59 58
0000000390: C3 50 53 51 52 8A C3 E8 0F 00 8A C5 E8 0A 00 8A
00000003A0: C1 E8 05 00 5A 59 5B 58 C3 50 53 52 B4 00 B3 10
00000003B0: F6 F3 8B D0 B4 02 80 C2 30 CD 21 8A D6 80 C2 30
00000003C0: CD 21 5A 5B 58 C3 50 B4 00 F6 F3 8B D0 B4 02 80
00000003D0: C2 30 CD 21 8A D6 80 C2 30 CD 21 58 C3 1E 2B C0
00000003E0: 50 B8 01 00 8E D8 E8 F5 FE E8 C7 FE 32 C0 B4 4C
00000003F0: CD 21

0
JOS version is
$.JOS OEM is $.JOS
erial number is
$.JPC 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 $
é*0RP`0Í!º è' è
- º!! è¸ º!! ŠÇ' è
¸ ºº èý è¸ XZÄP
R, ðŽÄ& þÿ<ýt,<þ
t.<üt*<üt,<üt.<ü
t0<øt2<ýt4<üt6ºE
è= è' è4ºº? è+º
ºB è%ººH èvººK è
ºººX è!!ººk èJººx
è•ºº} è0ºº Z•X
ÄP`oÍ!XÄSRP³ºèf
ºº èëÿŠÄè[ XZ[ÄP
QRS»º 3É30÷óRA...Ä
uó`ºZ€Ä0Í!âø[ZYX
ÄPSQRSÄºº ŠÄºº Š
Äºº ZY[XÄPSR` ºº
öó<D`º€Ä0Í!ŠÖ€Ä0
Í!Z[XÄP` öó<D`º€
Ä0Í!ŠÖ€Ä0Í!XÄÄ+Ä
P,º Žðèðèçþ2Ä`L
Í!
```

Рисунок 5 – Содержимое файла «хорошего» EXE модуля

```

D:\asm\LAB1_COM.COM
00000000: E9 BF 01 0D 0A 4F 53 20 76 65 72 73 69 6F 6E 20 é;@.OS version
00000001: 69 73 20 24 2E 24 0D 0A 4F 45 4D 20 69 73 20 24 is $.$.OEM is $
00000002: 0D 0A 53 65 72 69 61 6C 20 6E 75 6D 62 65 72 20 .Serial number
00000003: 69 73 20 24 0D 0A 50 43 20 74 79 70 65 20 69 73 is $.PC type is
00000004: 20 24 50 43 24 50 43 2F 58 54 24 41 54 24 50 53 $PC$PC/XT$AT$PS
00000005: 32 20 6D 6F 64 65 6C 20 33 30 24 50 53 32 20 6D 2 model 30$PS2 m
00000006: 6F 64 65 6C 20 35 30 20 6F 72 20 36 30 24 50 53 odel 50 or 60$PS
00000007: 32 20 6D 6F 64 65 6C 20 38 30 24 50 43 6A 72 24 2 model 80$PCjr$
00000008: 50 43 20 43 6F 6E 76 65 72 74 69 62 6C 65 24 75 PC Convertible$u
00000009: 6E 6B 6E 6F 77 6E 20 24 52 50 B4 30 CD 21 BA 03 nknown $RP'0Í!♥
0000000A: 01 E8 92 00 E8 96 00 BA 16 01 E8 89 00 BA 16 01 @è' è- ¨-@è% ¨-@
0000000B: 8A C7 B4 00 E8 9D 00 BA 20 01 E8 79 00 E8 B6 00 ŠÇ' è  ¨ 0èy èŸ
0000000C: 58 5A C3 50 06 52 B8 00 F0 8E C0 26 A0 FE FF 3C XZÄP¶R, ðŽÂ& þÿ<
0000000D: FF 74 2C 3C FE 74 2E 3C FB 74 2A 3C FC 74 2C 3C ŷt,<þt.<üt* <üt,<
0000000E: FA 74 2E 3C FC 74 30 3C F8 74 32 3C FD 74 34 3C út.<üt0<øt2<ýt4<
0000000F: F9 74 36 BA 8F 01 E8 3D 00 E8 92 00 EB 34 90 BA ùt6  ¨  ¨è= è' ë4 ¨ ¨
00000010: 42 01 EB 2B 90 BA 45 01 EB 25 90 BA 4B 01 EB 1F B0ë+ ¨ ¨E ¨ ¨% ¨ ¨K ¨ ¨ë▼
00000011: 90 BA 4E 01 EB 19 90 BA 5B 01 EB 13 90 BA 6E 01 ¨ ¨N ¨ ¨ë ¨ ¨ ¨ ¨[ ¨ ¨ë!! ¨ ¨n ¨
00000012: EB 0D 90 BA 7B 01 EB 07 90 BA 80 01 EB 01 90 E8 ë ¨ ¨ ¨{ ¨ ¨ë+ ¨ ¨ ¨ ¨ë ¨ ¨ ¨è
00000013: 04 00 5A 07 58 C3 50 B4 09 CD 21 58 C3 53 52 50 ♦ Z•XÄP´oÍ!XÄSRP
00000014: B3 0A E8 66 00 BA 14 01 E8 EB FF 8A C4 E8 5B 00 ³ ¨èf ¨Ÿ ¨èëÿŠÄè[
00000015: 58 5A 5B C3 50 51 52 53 BB 0A 00 33 C9 33 D2 F7 XZ[ÄPQRS» ¨ 3É3Ð÷
00000016: F3 52 41 85 C0 75 F6 B4 02 5A 80 C2 30 CD 21 E2 óRA...Äuö´0Z€Ä0Í!â
00000017: F8 5B 5A 59 58 C3 50 53 51 52 8A C3 E8 0F 00 8A ø[ZYXÄPSQRŠÄèø Š
00000018: C5 E8 0A 00 8A C1 E8 05 00 5A 59 5B 58 C3 50 53 Äè ¨ ŠÄè+ ZY[XÄPS
00000019: 52 B4 00 B3 10 F6 F3 8B D0 B4 02 80 C2 30 CD 21 R´ ³»öó<Ð´0€Ä0Í!
0000001A: 8A D6 80 C2 30 CD 21 5A 5B 58 C3 50 B4 00 F6 F3 ŠÖ€Ä0Í!Z[XÄP´ öó
0000001B: 8B D0 B4 02 80 C2 30 CD 21 21 8A D6 80 C2 30 CD 21 <Ð´0€Ä0Í!ŠÖ€Ä0Í!
0000001C: 58 C3 E8 FE FE E8 D0 FE 32 C0 B4 4C CD 21 XÄèþpèÐþ2Ä´LÍ!

```

Рисунок 6 – Содержисое файла COM модуля

D:\asm\LAB1_COM.EXE															
00000001D0:	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
00000001F0:	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
0000000210:	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
0000000230:	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
0000000250:	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
0000000270:	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
0000000290:	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
00000002B0:	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
00000002D0:	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
00000002F0:	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 é:0)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 OS 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`0Í!e♥
00000003A0:	01	E8	92	00	E8	96	00	BA	16	01	E8	89	00	BA	16 01 0è' è- e-0è% e-0
00000003B0:	8A	C7	B4	00	E8	9D	00	BA	20	01	E8	79	00	E8	B6 00 ŠÇ' è 0 èy è 9
00000003C0:	58	5A	C3	50	06	52	B8	00	F0	8E	C0	26	A0	FE	FF 3C XZÄP✦R. ðŽ& þÿ<
00000003D0:	FF	74	2C	3C	FE	74	2E	3C	FB	74	2A	3C	FC	74	2C 3C ŷt,<þt.<üt* <üt,<
00000003E0:	FA	74	2E	3C	FC	74	30	3C	F8	74	32	3C	FD	74	34 3C út.<üt0<øt2<ýt4<
00000003F0:	F9	74	36	BA	8F	01	E8	3D	00	E8	92	00	EB	34	90 BA üt6º00è= è' è4º
0000000400:	42	01	EB	2B	90	BA	45	01	EB	25	90	BA	4B	01	EB 1F B0ë+ºE0ë%ºK0ë▼
0000000410:	90	BA	4E	01	EB	19	90	BA	5B	01	EB	13	90	BA	6E 01 ººN0ë↓ºº[0ë!!ººnº
0000000420:	EB	0D	90	BA	7B	01	EB	07	90	BA	80	01	EB	01	90 E8 ë)ºº{0ë•ºº€0ëººè
0000000430:	04	00	5A	07	58	C3	50	B4	09	CD	21	58	C3	53	52 50 ♦ Z•XÄP´oÍ!XÄSRP
0000000440:	B3	0A	E8	66	00	BA	14	01	E8	EB	FF	8A	C4	E8	5B 00 ³ëf º90ëëÿŠÄè[
0000000450:	58	5A	5B	C3	50	51	52	53	BB	0A	00	33	C9	33	D2 F7 XZ[ÄPQRS» 3É3D÷
0000000460:	F3	52	41	85	C0	75	F6	B4	02	5A	80	C2	30	CD	21 E2 óRA...Äuö´0Z€Ä0Í!â
0000000470:	F8	5B	5A	59	58	C3	50	53	51	52	8A	C3	E8	0F	00 8A ø[ZYXÄPSQRSŠÄèº Š
0000000480:	C5	E8	0A	00	8A	C1	E8	05	00	5A	59	5B	58	C3	50 53 Äè ŠÄè+ ZY[XÄPS
0000000490:	52	B4	00	B3	10	F6	F3	8B	D0	B4	02	80	C2	30	CD 21 R´ ³»öó<ð´0€Ä0Í!
00000004A0:	8A	D6	80	C2	30	CD	21	5A	5B	58	C3	50	B4	00	F6 F3 ŠÖ€Ä0Í!Z[XÄP´ öó
00000004B0:	8B	D0	B4	02	80	C2	30	CD	21	8A	D6	80	C2	30	CD 21 <ð´0€Ä0Í!ŠÖ€Ä0Í!
00000004C0:	58	C3	E8	FE	FE	E8	D0	FE	32	C0	B4	4C	CD	21	32 XÄèþþèðþ2Ä´LÍ!

Рисунок 7 – Содержимое файла «плохого» EXE модуля

## Отличие форматов файлов COM и EXE модулей

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

COM файл содержит только машинный код и данные программы. Код начинается с адреса 0h, но при загрузке происходит смещение на 100h.

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

В «плохом» EXE модуле машинный код и данные содержатся в одном сегменте. С адреса 0h идёт таблица настроек (Relocation table). Код располагается с адреса 300h.

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

В «хорошем» EXE модуле данные, стек и машинный код находятся в разных сегментах. От «плохого» EXE он так же отличается наличием стека.

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

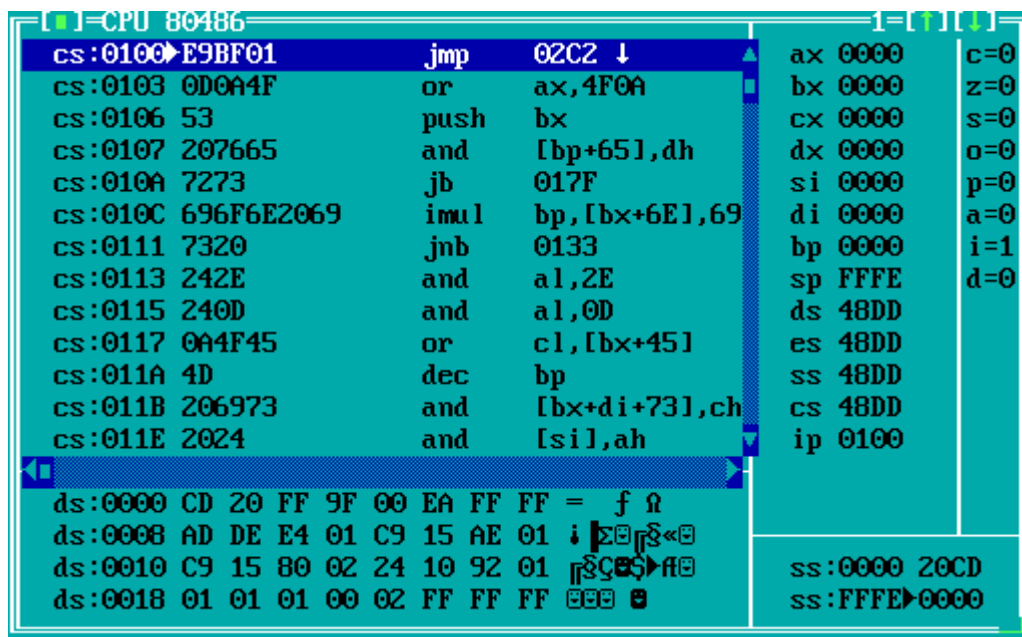


Рисунок 8 – Результат загрузки COM файла в память



## **Загрузка COM модуля в основную память**

### **1) Какой формат загрузки модуля COM? С какого адреса располагается код?**

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) Что располагается с адреса 0?**

Со смещения 0h в программном сегменте начинается PSP.

### **3) Какие значения имеют сегментные регистры? На какие области памяти они указывают?**

Сегментные регистры указывают на начало адрес сегмента PSP.

### **4) Как определяется стек? Какую область памяти он занимает? Какие адреса?**

Регистр SP указывает на конец стека (FFFFh), SS – на начало (0h). Адреса расположены в диапазоне 0h – FFFEh (FFFEh, т.к. это последний адрес, кратный двум).

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

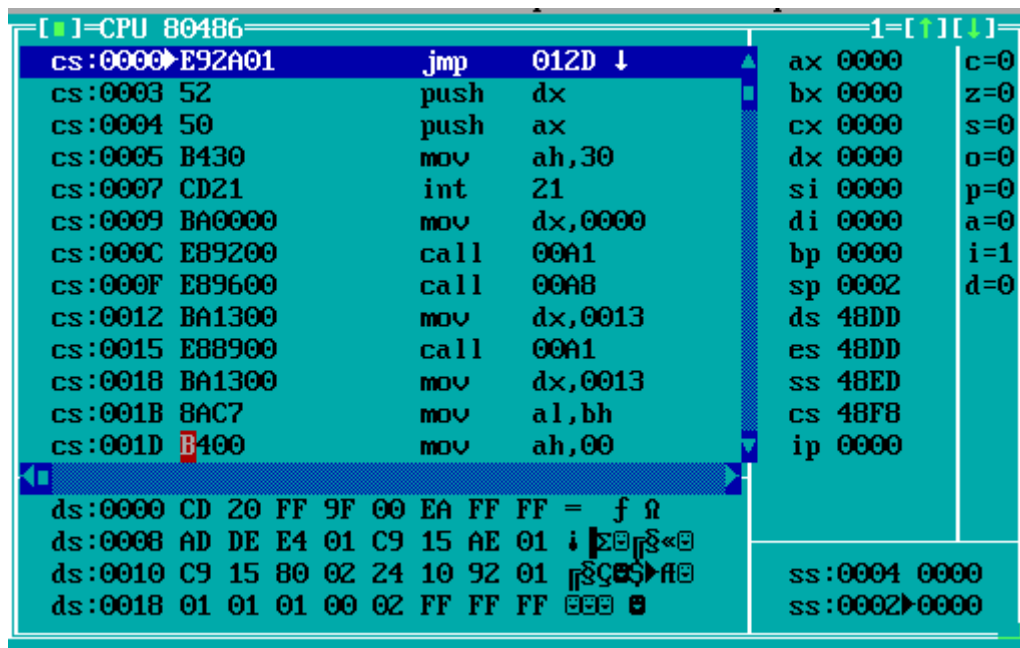


Рисунок 9 – Результат загрузки EXE файла в память

## Загрузка «хорошего» EXE модуля в основную память

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

Для PSP и программы выделяется блок памяти. После запуска программы DS и ES указывают на начало PSP (48DDh), CS – на начало сегмента команд (4919h), а SS – на начало сегмента стека (48EDh). IP имеет ненулевое значение, так как в программе есть дополнительные процедуры, расположенные до основной.

2) На что указывают регистры DS и ES?

Регистры ES и DS указывают на начало PSP.

3) Как определяется стек?

Стек определяется при помощи директивы ASSUME CS:MYSTACK, где MYSTACK – сегмент, отведенный под стек:

```
MYSTACK SEGMENT STACK
```

```
DW 100H
```

```
MYSTACK ENDS
```

#### **4) Как определяется точка входа?**

Смещение точки входа в программу загружается в указатель команд IP и определяется операндом директивы END, который называется точкой входа. Операндом является функция или метка, с которой необходимо начать программу.

#### **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 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;
        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 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
```