

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

Студент гр. 8383

Федоров И.А.

Преподаватель

Ефремов 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.

Ход работы.

Был написан текст исходного **.COM** модуля, который определяет тип PC и версию системы. Для этого программа считывает байт по адресу 0F000:0FFFEh, в котором хранится тип IBM PC, и сравнивает полученный код с таблицей соответствия код-тип (см. табл. 1). Исходный код представлен в приложении А.

Таблица 1 - Соответствие кода и тип PC

Тип IBM PC	Код
PC	FF
PC/XT	FE, FB
AT	FC
PS2 модель 30	FA
PS2 модель 50 или 60	FC
PS2 модель 80	F8
PCjr	FD
PC Convertible	F9

Так как коды для таких типов как AT и PS2 модели 50/60 совпадают, то программа производит дополнительную проверку. Для этого вызывается функция C0h прерывания int 15h, после выполнения которого регистры ES:BX будут указывать на таблицу в области ПЗУ BIOS, где в байте по смещению 03h будет храниться суб-код модели компьютера. В данном случае AT будет соответствовать дополнительный код 00h, а типу PS2 50/60 - 04h. Для определения версии MS DOS используется функция 30h прерывания int 21h.

Из исходного был построен "плохой" **.EXE** модуль, а затем с помощью программы EXE2BIN был получен "хороший" **.COM** модуль (см. рис. 1-2). Во

время линковки было выдано предупреждение об отсутствии объявления стека. выполнения и "хороший" **.COM** (см. рис. 1-2).

```
C:\MASM>masm LR_1COM.ASM
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.
All rights reserved.
Object filename [LR_1COM.OBJ]: Lr_1com.obj
Source listing [NUL.LST]: Lr_1com.lst
Cross-reference [NUL.CRF]: Lr_1com.crf

50424 + 449736 Bytes symbol space free

0 Warning Errors
0 Severe Errors
C:\MASM>_
```

Рисунок 1

```
C:\MASM>link LR_1COM.OBJ
Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987.
All rights reserved.
Run File [LR_1COM.EXE]:
List File [NUL.MAP]: lr_1com.map
Libraries [.LIB]:
LINK : warning L4021: no stack segment
C:\MASM>
```

Рисунок 2

Был написан текст исходного **.EXE** модуля (код представлен в приложении Б), который выполняет те же функции, в результате чего получен корректный модуль **.EXE**. Результаты работы "хорошего" и "плохого" **.EXE** модулей представлены на рис. 3 и 4 соответственно.

```
50424 + 449736 Bytes symbol space free

0 Warning Errors
0 Severe Errors

C:\MASM>link LR_1EXE.OBJ

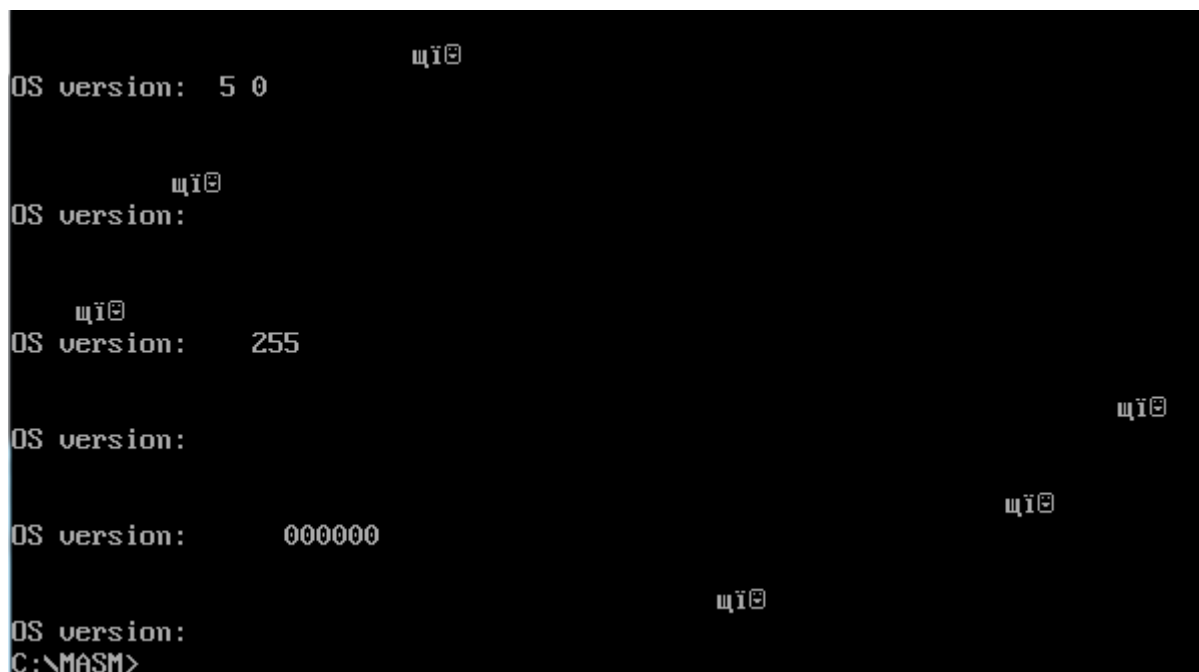
Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.

Run File [LR_1EXE.EXE]:
List File [NUL.MAP]: lr_1exe
Libraries [.LIB]:

C:\MASM>LR_1EXE.EXE

Version PC: AT
OS version: 5.0
Number OEM: 255
Serial number: 000000
C:\MASM>_
```

Рисунок 3 – Результат работы "хорошего" **.EXE** модуля



```
OS version: 5 0
OS version:
OS version: 255
OS version:
OS version: 000000
OS version:
C:\MASM>
```

Рисунок 4 – Результат работы некорректного **.EXE** модуля

Ответы на контрольные вопросы.

1) Сколько сегментов должна содержать COM-программа? EXE-программа?

Программа типа COM состоит из одного сегмента, в котором размещаются коды, данные и стек, поэтому размер COM программы ограничен 64К. Программа типа EXE не имеет ограничений, так как в нее может входить любое количество сегментов.

2) Какие директивы должны обязательно быть в тексте COM-программы?

Должна быть директива *org 256*, которая резервирует 256 байт для PSP (код программы начинается со смещения 100H). Так же необходима директива *assume*, которая укажет, что сегментные регистры будут соответствовать единственному сегменту программы. Так как IP будет установлен на 100H (сразу после PSP), то либо нужно чтобы команды располагались сразу, либо необходим оператор *jmp*, если данные расположить раньше кода.

3) Все ли форматы команд можно использовать в COM-программе?

Нет. Например нельзя использовать команды с оператором *seg* (т.к. в программе только один сегмент).

Отличия исходных текстов COM и EXE программ

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

Код располагается с 100h, т.к. это первый байт следующий за PSP. Если вначале программы желательно расположить данные, то необходимо поместить команду *jmp*. Шестнадцатеричный вид модуля **.COM** представлен на рис. 5. Файлы типа COM содержат только скомпилированный код без какой либо дополнительной информации о программе (в отличие от EXE).

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

Содержимое "плохого" EXE файла представлена на рис. 6-7. В "плохом" EXE данные и код хранятся в одном сегменте, как и у COM файла (т.к. оба собраны из одного исходного файла). В начале файл содержит заголовок размером 512 байт, в котором хранится информация, необходимая системе для настройки регистров и программы для загрузки ее в память. Поэтому код начинается с 300h.

3) Какова структура файла "хорошего" EXE? Чем он отличается от файла "плохо" EXE? Корректная версия в шестнадцатеричном виде представлена на рис. 8. У "хорошего" EXE команды программы, данные и стек находятся в отдельных сегментах, в отличие от "плохого", у которого все находится в одном сегменте, т.к. он собран из файла с программой, предназначенной для COM. Код "хорошего" EXE начинается 240h (512), т.к. в начале у него находится только заголовок размером 200h и выделенный в программе стек, код "плохого" начинается с 300h, т.к. кроме заголовка в программе выделена память под PSP.

```

view LR_1COM.COM - Far 3.0.5555 x64
C:\masm\LR_1COM.COM
00000000: E9 F5 01 0D 0A 4F 53 20 76 65 72 73 69 6F 6E 3A éô0JOS version:
00000001: 20 24 20 20 2E 20 20 24 0D 0A 4E 75 6D 62 65 72 $ . $JNumber
00000002: 20 4F 45 4D 3A 20 24 20 20 20 20 20 24 0D 0A OEM: $ $J
00000003: 53 65 72 69 61 6C 20 6E 75 6D 62 65 72 3A 20 24 Serial number: $
00000004: 20 20 20 20 20 20 20 20 20 20 20 24 0D 0A 56 $JAV
00000005: 65 72 73 69 6F 6E 20 50 43 3A 20 24 50 43 24 50 ersion PC: $PC$P
00000006: 43 2F 58 54 24 41 54 24 50 53 32 20 6D 6F 64 65 C/XT$AT$PS2 mode
00000007: 6C 20 33 30 24 50 53 32 20 6D 6F 64 65 6C 20 38 1 30$PS2 model 8
00000008: 30 24 50 43 6A 72 24 50 53 32 20 6D 6F 64 65 6C 0$PCjr$PS2 model
00000009: 20 35 30 2F 36 30 24 50 43 20 43 6F 6E 76 65 72 50/60$PC Conver
0000000A: 74 69 62 6C 65 24 4E 6F 74 20 66 6F 75 6E 64 20 tible$Not found
0000000B: 20 20 20 20 20 20 20 20 20 24 50 B4 09 CD 21 58 $P'oi!X
0000000C: C3 24 0F 3C 09 76 02 04 07 04 30 C3 51 8A E0 E8 Å$<ov0♦♦0AQŠaè
0000000D: EF FF 86 C4 B1 04 D2 E8 E8 E6 FF 59 C3 53 8A FC iÿ†Ä±♦ÖèèæÿYÄSŠü
0000000E: E8 E9 FF 88 25 4F 88 05 4F 8A C7 E8 DE FF 88 25 èéÿ~%0^+0ŠÇèpÿ~%
0000000F: 4F 88 05 5B C3 51 52 32 E4 33 D2 B9 0A 00 F7 F1 0^+ [ÄQR2ä3D¹ ÷ñ
00000010: 80 CA 30 88 14 4E 33 D2 3D 0A 00 73 F1 3C 00 74 €Ê0^9N3D= sñ< t
00000011: 04 0C 30 88 04 5A 59 C3 50 53 B8 00 F0 8E C0 26 ♦90^♦ZYÄPS. ðZÄ&
00000012: A0 FE FF BA 4D 01 E8 91 FF 3C FF 74 2B 3C FE 74 by°M0è'ÿ<ÿt+<þt
00000013: 2D 3C FB 74 29 3C FC 74 4F 3C FA 74 2D 3C F8 74 -<ût)<üt0<ût-<øt
00000014: 35 3C FD 74 37 3C F9 74 39 BE A6 01 83 C6 10 E8 5<ÿt7<üt9%!øfAè
00000015: 7A FF BA A6 01 EB 3F 90 BA 5C 01 EB 39 90 BA 5F zÿ°!øè?0°\øè90°_
00000016: 01 EB 33 90 BA 65 01 EB 2D 90 BA 68 01 EB 27 90 0è30°e0è-0°h0è'0è
00000017: BA 87 01 EB 21 90 BA 75 01 EB 1B 90 BA 82 01 EB °†0è!0°u0è<0°,0è
00000018: 15 90 BA 97 01 EB 0F 90 B4 C0 CD 15 26 8A 47 03 $0°-0è00'ÄÍ$&ŠG♥
00000019: 3C 00 74 D0 EB DA E8 21 FF 5B 58 C3 52 B4 30 CD < tDèÜè!ÿ[XÄR'0Í
0000001A: 21 50 BA 03 01 E8 12 FF BE 12 01 46 E8 46 FF 58 !P°♥0è†ÿ%†0FèÿX
0000001B: 8A C4 83 C6 03 E8 3D FF BA 12 01 E8 FC FE BA 18 ŠÄfA♥è=ÿ°†0èüþ°†
0000001C: 01 E8 F6 FE BE 27 01 83 C6 05 8A C7 E8 26 FF BA 0èöþ% 'øfA†ŠÇè&ÿ°
0000001D: 27 01 E8 E5 FE BA 2E 01 E8 DF FE BF 40 01 83 C7 '0èâþ°.0èßþ;0øfÇ
0000001E: 0A 8B C1 E8 F7 FE 8A C3 E8 E1 FE 83 EF 02 89 05 <Äè÷þŠÄèápfi0%†
0000001F: BA 40 01 E8 C4 FE 5A C3 E8 1D FF E8 9E FF 32 C0 °00èÄþZÄè+ÿèžÿ2Ä
00000020: B4 4C CD 21 1Í!

```

Рисунок 5 - Содержимое COM файла

```

view LR_1COM.EXE - Far 3.0.5555 x64
C:\masm\LR_1COM.EXE
00000000: 4D 5A 04 01 03 00 00 00 20 00 00 00 FF FF 00 00 MZ♦0♥ üÿ
00000001: 00 00 36 A8 00 01 00 00 1E 00 00 00 01 00 00 00 6" 0 ▲ 0
00000002: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000003: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000004: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000005: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000006: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000007: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000008: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000009: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000B: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000D: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000011: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

Рисунок 6 – Содержимое "плохого" EXE файла

00000002E0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000002F0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000300:	E9 F5 01 0D 0A 4F 53 20	76 65 72 73 69 6F 6E 3A	éô0)OS version:
0000000310:	20 24 20 20 2E 20 20 24	0D 0A 4E 75 6D 62 65 72	\$. \$)Number
0000000320:	20 4F 45 4D 3A 20 24 20	20 20 20 20 20 24 0D 0A	OEM: \$ \$)S
0000000330:	53 65 72 69 61 6C 20 6E	75 6D 62 65 72 3A 20 24	Serial number: \$
0000000340:	20 20 20 20 20 20 20 20	20 20 20 20 24 0D 0A 56	\$)S/V
0000000350:	65 72 73 69 6F 6E 20 50	43 3A 20 24 50 43 24 50	ersion PC: \$PC\$P
0000000360:	43 2F 58 54 24 41 54 24	50 53 32 20 6D 6F 64 65	C/XT\$AT\$PS2 mode
0000000370:	6C 20 33 30 24 50 53 32	20 6D 6F 64 65 6C 20 38	l 30\$PS2 model 8
0000000380:	30 24 50 43 6A 72 24 50	53 32 20 6D 6F 64 65 6C	0\$PCjr\$PS2 model
0000000390:	20 35 30 2F 36 30 24 50	43 20 43 6F 6E 76 65 72	50/60\$PC Conver
00000003A0:	74 69 62 6C 65 24 4E 6F	74 20 66 6F 75 6E 64 20	tible\$Not found
00000003B0:	20 20 20 20 20 20 20 20	20 24 50 B4 09 CD 21 58	\$P'oí!X
00000003C0:	C3 24 0F 3C 09 76 02 04	07 04 30 C3 51 8A E0 E8	Ã\$ø<ov0♦♦0ÃQŠæ
00000003D0:	EF FF 86 C4 B1 04 D2 E8	E8 E6 FF 59 C3 53 8A FC	ÿÿtÃ+0èèæÿVÃSŠü
00000003E0:	E8 E9 FF 88 25 4F 88 05	4F 8A C7 E8 DE FF 88 25	èéÿ`%0`+0ŠÇèbÿ`%
00000003F0:	4F 88 05 5B C3 51 52 32	E4 33 D2 B9 0A 00 F7 F1	O`+ [ÃQR2ã30` ÷ñ
0000000400:	80 CA 30 88 14 4E 33 D2	3D 0A 00 73 F1 3C 00 74	€E0`¶N30= sñ< t
0000000410:	04 0C 30 88 04 5A 59 C3	50 53 B8 00 F0 8E C0 26	♦00`ZYÃPS. ðŽÃ&
0000000420:	A0 FE FF BA 4D 01 E8 91	FF 3C FF 74 2B 3C FE 74	bÿM0è 'ÿ<ÿt+<øt
0000000430:	2D 3C FB 74 29 3C FC 74	4F 3C FA 74 2D 3C F8 74	-<üt)<üt0<üt-<øt
0000000440:	35 3C FD 74 37 3C F9 74	39 BE A6 01 83 C6 10 E8	5<ÿt7<üt9% øfA>è
0000000450:	7A FF BA A6 01 EB 3F 90	BA 5C 01 EB 39 90 BA 5F	zyø!øè?0ø\0è90ø_
0000000460:	01 EB 33 90 BA 65 01 EB	2D 90 BA 68 01 EB 27 90	0è30øøè-0øh0è'0
0000000470:	BA 87 01 EB 21 90 BA 75	01 EB 1B 90 BA 82 01 EB	øtøè!0øu0è<0ø,0è
0000000480:	15 90 BA 97 01 EB 0F 90	B4 C0 CD 15 26 8A 47 03	\$0ø-0èø0`ÃÍ\$8ŠG♥
0000000490:	3C 00 74 D0 EB DA E8 21	FF 5B 58 C3 52 B4 30 CD	< tðèÜè!ÿ[XÃR`0Í
00000004A0:	21 50 BA 03 01 E8 12 FF	BE 12 01 46 E8 46 FF 58	!Pø♥0ètÿ%tøFèFÿX
00000004B0:	8A C4 83 C6 03 E8 3D FF	BA 12 01 E8 FC FE BA 18	ŠÃfA♥è=ÿøtøèüþø†
00000004C0:	01 E8 F6 FE BE 27 01 83	C6 05 8A C7 E8 26 FF BA	0èøþ%`øfA+ŠÇè&ÿø
00000004D0:	27 01 E8 E5 FE BA 2E 01	E8 DF FE BF 40 01 83 C7	'0èâþø.0èßþøøfC
00000004E0:	0A 8B C1 E8 F7 FE 8A C3	E8 E1 FE 83 EF 02 89 05	ø<Ãè:þŠÃèâþfï0%+
00000004F0:	BA 40 01 E8 C4 FE 5A C3	E8 1D FF E8 9E FF 32 C0	ø0øèÃþZÃè+ÿèžÿ2Ã
0000000500:	B4 4C CD 21		`LÍ!
1Help 2Text 3Quit 4Dump 5 6Edit 7Search 8AN			

Рисунок 7

```

C:\masm\LR_1\EXE.EXE
00000000: 4D 5A 55 00 03 00 01 00 20 00 00 00 FF FF 00 00 MZU ♥ @
00000001: 40 00 03 95 40 01 10 00 1E 00 00 00 01 00 44 01 @ ♥X@@▶ ▲ @ D@
00000002: 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ▶
00000003: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000004: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000005: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000006: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000007: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000008: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000009: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000B: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000D: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000011: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000012: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000013: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000014: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000015: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000016: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000017: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000018: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000019: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001B: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001D: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000021: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000022: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000023: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000024: 0D 0A 4F 53 20 76 65 72 73 69 6F 6E 3A 20 24 20 OS version: $
00000025: 20 2E 20 20 24 0D 0A 4E 75 6D 62 65 72 20 4F 45 . $OSNumber OE
00000026: 4D 3A 20 24 20 20 20 20 20 20 24 0D 0A 53 65 72 M: $ OSSer
00000027: 69 61 6C 20 6E 75 6D 62 65 72 3A 20 24 20 20 20 ial number: $
00000028: 20 20 20 20 20 20 20 20 20 20 24 0D 0A 56 65 72 73 OSVers
00000029: 69 6F 6E 20 50 43 3A 20 24 50 43 24 50 43 2F 58 ion PC: $PC$PC/X
0000002A: 54 24 41 54 24 50 53 32 20 6D 6F 64 65 6C 20 33 T$AT$PS2 model 3

```

Рисунок 8 – Содержимое "хорошего" EXE

Загрузка COM модуля в основную память

1) Какой формат загрузки модуля COM? С какого адреса располагается код? Вид в памяти модуля COM представлен на рис. 9. PSP, команды, данные, а также стек находятся в одном сегменте и идут подряд в памяти. После загрузки в память регистр IP инициализируется числом 256 (100h), т.к. в начале сегмента находится PSP, поэтому код программы будет иметь адрес 100h.



Рисунок 9 – Вид модуля COM в памяти

2) Что располагается с адреса 0?

С адреса 0 располагается PSP, который заполняется системой (но память под него выделена в программе командой `org 100h`).

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

Результат работы отладчика TD.EXE для COM продемонстрирован на рис. 10. После загрузки программы все сегментные регистры указывают на начало сегмента (единственного), т.е. фактически на начало PSP. Все они имеют одинаковое значение 489D.

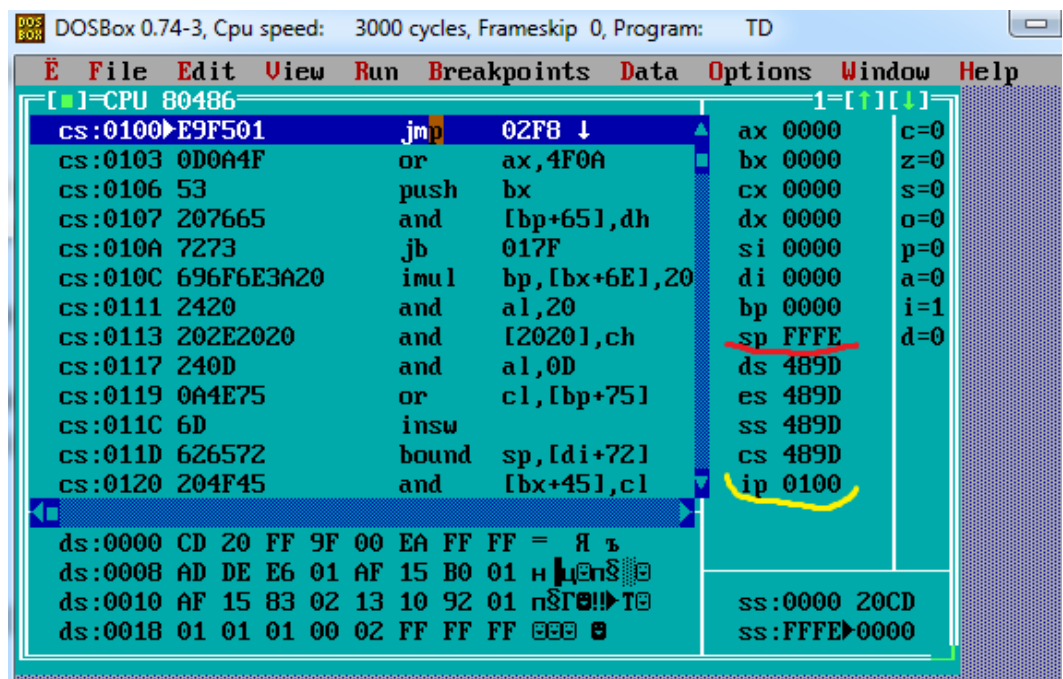


Рисунок 10 – Результат работы отладчик TD для .COM

3) Как определяется стек?

После загрузки в память регистр *SS* указывает на начало сегмента стека, а в указатель стека *SP* кладется смещение конца сегмента стека. Стек может определяться либо "вручную" с помощью директивы *stack*, либо при отсутствии явного объявления система сама создает стек по умолчанию по адресу *FFFFh* (относительно начала сегмента команд).

4) Как определяется точка входа?

Смещение точки входа берется из операнда директивы *end* и загружается в регистр *IP*.

Выводы.

В ходе работы были исследованы различия в исходных текстах модулей типов *.COM* и *.EXE*. Были разобраны структуры файлов загрузочных моделей и способы их загрузки в основную память.

ПРИЛОЖЕНИЕ А

КОД ДЛЯ СОМ ФАЙЛА

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

; ДАННЫЕ
SYSTEM_VER db 13,10,"OS version: $"
OUT_VER db " . $"
OEM_NUMBER db 13, 10, "Number OEM: $"
OUT_OEM db "    $"
USER_NUMBER db 13, 10, "Serial number: $"
OUT_UNUM db "    $"
PC_VER db 13, 10, "Version PC: $"
MODEL_PC db "PC$"
MODEL_PCXT db "PC/XT$"
MODEL_AT db "AT$"
MODEL_30 db "PS2 model 30$"
MODEL_80 db "PS2 model 80$"
MODEL_JR db "PCjr$"
MODEL_5060 db "PS2 model 50/60$"
MODEL_CONVERT db "PC Convertible$"
WARNING_VER db "Not found"      $"

; ПРОЦЕДУРЫ
;-----

WRITE_STR PROC near
    push ax
    mov ah, 09h
    int 21h
    pop ax
    ret
WRITE_STR 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
;-----

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

WRD_TO_HEX PROC near
;перевод в 16 с/с 16-ти разрядного числа
```

```

; в AX - число, DI - адрес последнего символа
    push    BX
    mov     BH, AH
    call    BYTE_TO_HEX
    mov     [DI], AH
    dec     DI
    mov     [DI], AL
    dec     DI
    mov     AL, BH
    call    BYTE_TO_HEX
    mov     [DI], AH
    dec     DI
    mov     [DI], AL
    pop     BX
    ret

WRD_TO_HEX ENDP
;-----
BYTE_TO_DEC PROC near
; перевод в 10с/с, SI - адрес поля младшей цифры
    push    CX
    push    DX
    xor     AH, AH
    xor     DX, DX
    mov     CX, 10
loop_bd:  div     CX
    or      DL, 30h
    mov     [SI], DL
            dec     si
    xor     DX, DX
    cmp     AX, 10
    jae     loop_bd
    cmp     AL, 00h
    je      end_1
    or      AL, 30h
    mov     [SI], AL

end_1:    pop     DX
    pop     CX
    ret

BYTE_TO_DEC ENDP
;-----

```

```

PRINT_PCTYPE PROC near

    push    ax
    push    bx
    MOV     AX, 0F000H ; указывает ES на ПЗУ
    MOV     ES, AX ;
    MOV     AL, ES:[0FFFEH] ;получаем байт
    mov     dx, offset PC_VER
    call    WRITE_STR
    cmp     al, 0FFH ; PC
    JE      mod_pc
    cmp     al, 0FEH ; PC/XT
    JE      mod_xt
    cmp     al, 0FBH ; PC/XT
    JE      mod_xt
    cmp     al, 0FCH
    JE      is_at_5060
    cmp     al, 0FAH ; PC2 30
    JE      mod_30
    cmp     al, 0F8H ; PC2 80

```

```

JE mod_80
cmp al, 0FDH ; PCjr
JE mod_jr
cmp al, 0F9H ; PC Convertible
JE mod_conv

error_type:
    mov si, offset WARNING_VER
    add si, 16
    call BYTE_TO_HEX
    mov dx, offset WARNING_VER
    jmp end_

mod_pc:
    mov dx, offset MODEL_PC
    jmp end_

mod_xt:
    mov dx, offset MODEL_PCXT
    jmp end_

mod_at:
    mov dx, offset MODEL_AT
    jmp end_

mod_30:
    mov dx, offset MODEL_30
    jmp end_

mod_5060:
    mov dx, offset MODEL_5060
    jmp end_

mod_80:
    mov dx, offset MODEL_80
    jmp end_

mod_jr:
    mov dx, offset MODEL_JR
    jmp end_

mod_conv:
    mov dx, offset MODEL_CONVERT
    jmp end_

is_at_5060:
    mov ah, 192 ; At - 00 ; 50/60 - 04
    int 15h
    mov al, ES:[BX+3]
    cmp al, 00h
    je mod_at ; =
    jmp mod_5060

end_:
    call WRITE_STR
    pop bx
    pop ax
    ret
PRINT_PCTYPE ENDP

PRINT_OS PROC near
    push dx
    mov ah, 30h
    int 21h
    push ax
    mov dx, offset SYSTEM_VER
    call WRITE_STR
    mov si, offset OUT_VER
    inc si
    call BYTE_TO_DEC
    pop ax

```

```

mov al, ah ;делает над ah
add si, 3
call BYTE_TO_DEC
mov dx, offset OUT_VER
call WRITE_STR
mov dx, offset OEM_NUMBER
call WRITE_STR
mov si, offset OUT_OEM
add si, 5
mov al, bh ;хранится оем
call BYTE_TO_DEC
mov dx, offset OUT_OEM
call WRITE_STR
mov dx, offset USER_NUMBER
call WRITE_STR
mov di, offset OUT_UNUM
add di, 10
mov ax, cx
call WRD_TO_HEX
mov al, bl
call BYTE_TO_HEX
sub di, 2
mov [di], ax
mov dx, offset OUT_UNUM
call WRITE_STR
pop dx
ret
PRINT_OS ENDP
;-----

```

```

; КОД"
BEGIN:
    call PRINT_PCTYPE
    call PRINT_OS
    xor AL,AL
    mov AH,4Ch
    int 21H ; в com cs уже на psp
; int 20h

TESTPC ENDS
END START

```

ПРИЛОЖЕНИЕ Б

КОД ДЛЯ EXE ФАЙЛА

```

AStack    SEGMENT    STACK
           DW 20h DUP(?)

AStack    ENDS

DATA      SEGMENT
           SYSTEM_VER db 13,10,"OS version: $"
           OUT_VER db " . $"
           OEM_NUMBER db 13, 10, "Number OEM: $"
           OUT_OEM db "    $"
           USER_NUMBER db 13, 10, "Serial number: $"
           OUT_UNUM db "    $"
           PC_VER db 13, 10, "Version PC: $"
           MODEL_PC db "PC$"
           MODEL_PCXT db "PC/XT$"
           MODEL_AT db "AT$"
           MODEL_30 db "PS2 model 30$"
           MODEL_80 db "PS2 model 80$"
           MODEL_JR db "PCjr$"
           MODEL_5060 db "PS2 model 50/60$"
           MODEL_CONVERT db "PC Convertible$"
           WARNING_VER db "Not found    $"
DATA      ENDS

CODE      SEGMENT
           ASSUME CS:CODE,DS:DATA,SS:AStack

; ПРОЦЕДУРЫ
;-----

WRITE_STR PROC near
           push ax
           mov ah, 09h
           int 21h
           pop ax
           ret
WRITE_STR 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
;-----

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

```



```

;-----
WRD_TO_HEX PROC near
;перевод в 16 с/с 16-ти разрядного числа
; в AX - число, DI - адрес последнего символа
    push    BX
    mov     BH,AH
    call    BYTE_TO_HEX
    mov     [DI],AH
    dec     DI
    mov     [DI],AL
    dec     DI
    mov     AL,BH
    call    BYTE_TO_HEX
    mov     [DI],AH
    dec     DI
    mov     [DI],AL
    pop     BX
    ret
WRD_TO_HEX ENDP
;-----
BYTE_TO_DEC PROC near
; перевод в 10с/с, SI - адрес поля младшей цифры
    push    CX
    push    DX
    xor     AH,AH
    xor     DX,DX
    mov     CX,10
loop_bd:  div     CX
    or      DL,30h
    mov     [SI],DL
    dec     si
    xor     DX,DX
    cmp     AX,10
    jae     loop_bd
    cmp     AL,00h
    je      end_1
    or      AL,30h
    mov     [SI],AL

end_1:    pop     DX
    pop     CX
    ret
BYTE_TO_DEC ENDP
;-----

```

```

PRINT_PCTYPE PROC near
    push    ax
    push    bx
    MOV     AX,0F000H ; указывает ES на ПЗУ
    MOV     ES,AX ;
    MOV     AL,ES:[0FFFEH] ;получаем байт
    mov     dx, offset PC_VER
    call    WRITE_STR
    cmp     al, 0FFH ; PC
    JE      mod_pc
    cmp     al, 0FEH ; PC/XT
    JE      mod_xt
    cmp     al, 0FBH ; PC/XT
    JE      mod_xt
    cmp     al, 0FCH
    JE      is_at_5060
    cmp     al, 0FAH ; PC2 30

```

```

JE mod_30
cmp al, 0F8H ; PC2 80
JE mod_80
cmp al, 0FDH ; PCjr
JE mod_jr
cmp al, 0F9H ; PC Convertible
JE mod_conv

error_type:
    mov si, offset WARNING_VER
    add si, 16
    call BYTE_TO_HEX
    mov dx, offset WARNING_VER
    jmp end_

mod_pc:
    mov dx, offset MODEL_PC
    jmp end_

mod_xt:
    mov dx, offset MODEL_PCXT
    jmp end_

mod_at:
    mov dx, offset MODEL_AT
    jmp end_

mod_30:
    mov dx, offset MODEL_30
    jmp end_

mod_5060:
    mov dx, offset MODEL_5060
    jmp end_

mod_80:
    mov dx, offset MODEL_80
    jmp end_

mod_jr:
    mov dx, offset MODEL_JR
    jmp end_

mod_conv:
    mov dx, offset MODEL_CONVERT
    jmp end_

is_at_5060:
    mov ah, 192 ; At - 00 ; 50/60 - 04
    int 15H
    mov al, ES:[BX+3]
    cmp al, 00h
    je mod_at ; =
    jmp mod_5060

end_:
    call WRITE_STR
    pop bx
    pop ax
    ret
PRINT_PCTYPE ENDP

PRINT_OS PROC near
    push dx
    mov ah, 30h
    int 21h
    push bx
    push ax
    mov dx, offset SYSTEM_VER
    call WRITE_STR

```

```

        mov si, offset OUT_VER
        inc si
        call BYTE_TO_DEC
        pop ax
mov al, ah ;делает над ah
add si, 3
        call BYTE_TO_DEC
        mov dx, offset OUT_VER
        call WRITE_STR
        mov dx, offset OEM_NUMBER
        call WRITE_STR
        mov si, offset OUT_OEM
        add si, 5
        pop bx
        mov al, bh ;хранится оем
        call BYTE_TO_DEC
        mov dx, offset OUT_OEM
        call WRITE_STR
        mov dx, offset USER_NUMBER
        call WRITE_STR
mov di, offset OUT_UNUM
        add di, 10
        mov ax, cx
        call WRD_TO_HEX
        mov al, bl
        call BYTE_TO_HEX
        sub di, 2
        mov [di], ax
        mov dx, offset OUT_UNUM
        call WRITE_STR

        pop dx
        ret
PRINT_OS ENDP
;-----

Main PROC FAR
        push ax
        sub AX,AX
        mov AX,DATA
        mov DS,AX
        pop ax
        call PRINT_PCTYPE
        call PRINT_OS
        xor AL,AL
        mov AH,4Ch
        int 21H
        ;ret
Main ENDP
CODE ENDS
        END Main

```