

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

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

Студент гр. 8381

Киреев К.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.

Основные теоретические положения.

Тип IBM PC хранится в байте по адресу 0F000:0FFFEh, в предпоследнем байте ROM BIOS. Соответствие кода и типа компьютера представлено в таблице 1:

Таблица 1 – Идентификация типа компьютера

Код	Тип компьютера
FF	Оригинальный IBM PC
FE	XT, Portable PC
FD	PCjr
FC	AT
FB	XT с памятью 640 К на мат. плате
FA	PS/2 модель 25 или 30
F9	Convertible PC
F8	PS/2 модели 55SX, 70, 80
9A	Compaq XT, Compaq Plus
30	Sperry PC
2D	Compaq PC, Compaq Deskpro

Для определения версии MS DOS следует воспользоваться функцией 30H прерывания 21H. Входным параметром является номер функции в AH:

MOV AH, 30h

INT 21h

Выходными параметрами являются:

AL – номер основной версии;

AH – номер модификации;

ВН – серийный номер OEM (Original Equipment Manufacturer);

BL:СХ – 24-битовый серийный номер пользователя.

Постановка задачи.

Требуется реализовать текст исходного .COM модуля, который определяет тип РС и версию системы. Ассемблерная программа должна читать содержимое предпоследнего байта ROM BIOS, по таблице, сравнивая коды, определять тип РС и выводить строку с названием модели. Если код не совпадает ни с одним значением, то двоичный код переводиться в символьную строку, содержащую запись шестнадцатеричного числа и выводиться на экран в виде соответствующего сообщения.

Затем определяется версия системы. Ассемблерная программа должна по значениям регистров AL и AH формировать текстовую строку в формате хх.уу, где хх - номер основной версии, а уу - номер модификации в десятичной системе счисления, формировать строки с серийным номером OEM (Original Equipment Manufacturer) и серийным номером пользователя. Полученные строки выводятся на экран. Далее производится отладка полученного исходного модуля. Результатом выполнения этого действия будет «хороший» .COM модуль, а также необходимо построить «плохой» .EXE, полученный из исходного текста для .COM модуля. Затем нужно написать текст «хорошего» .EXE модуля, который выполняет те же функции, что и модуль .COM, далее его построить, отладить и сравнить исходные тексты для .COM и .EXE модулей, ответить на контрольные вопросы.

Выполнение работы.

Написан текст исходного .COM модуля, который определяет тип РС и версию системы. Полученный исходный модуль был отлажен. В результате был получен «плохой» .EXE модуль и построен «хороший» .COM модуль с помощью программы EXE2BIN.COM. Во время линковки получено предупреждение об отсутствии сегмента стека, представленное на рис. 1.

```

S:\>masm lab_com.asm
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1988. All rights reserved.

Object filename [lab_com.OBJ]:
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:

    49960 + 455253 Bytes symbol space free

    0 Warning Errors
    0 Severe Errors

S:\>link lab_com.obj

Microsoft (R) Overlay Linker Version 3.64
Copyright (C) Microsoft Corp 1983-1988. All rights reserved.

Run File [LAB_COM.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment

```

Рисунок 1 – Предупреждение об отсутствии стека

Запуск «хорошего» .COM модуля представлен на рис. 2.

```

S:\>exe2bin lab_com.exe lab_com.com

S:\>lab_com.com
PC type: AT or PS2 .50/60
MSDOS version: 5.0
OEM serial number: 0
User serial number: 000000

```

Рисунок 2 – «Хороший» .COM модуль

Запуск «плохого» .EXE модуля представлен на рис. 3.

```

S:\>lab_com.exe
<#t#<"t#|
~±G~0ê#XZY|P|      =!X|PRUWì~0x€ 7  ≡Ä^&a# < tt<#t#<"tt*°t0<²t6<·t<i520x_
δ=Éì_70x7 δ3Éì_?0x« δ)Éì_R0xñ δ▼Éì_\0xÙ δ§Éì_f0xÉ δδÉì_m0xâ δ@É|0=!i6ê0â|*x0 â|♥
â-0G ì_ê0xf è||i6f0â|!!x4 ì_f0xs è|ì6"0x·èD¶â|↓i|i¹x²·ì_"0x6 _^ZX|L=!|PC
Éì_70x7 δ3Éì_?0x« δ0Éì_R0xñ δ▼Éì_\0xÙ δ§Éì_f0xÉ δδÉì_m0xâ δ@É|0=!i6ê0â|*x0 â|♥â-
0G ì_ê0xf è||i6f0â|!!x4 ì_f0xs è|ì6"0x·èD¶â|↓i|i¹x²·ì_"0x6 _^ZX|L=!|PC
Éì_\0xÙ δ§Éì_f0xÉ δδ000000â δ@É|0=!i6ê0â|*x0 â|♥â-0G ì_ê0xf è||i6f0â|!!x4 ì_f0xs è
|ì6"0x·èD¶â|↓i|i¹x²·ì_"0x6 _^ZX|L=!|PC

```

Рисунок 3 – «Плохой» .EXE модуль

Написан текст исходного .EXE модуля, который выполняет те же функции, что и предыдущий модуль. Был получен «хороший» .EXE модуль, запуск которого представлен на рис. 4.

```
S:\>lab_exe.exe
PC type: AT or PS2 .50/60
MSDOS version: 5.0
OEM serial number: 0
User serial number: 000000
```

Рисунок 4 – «Хороший» .EXE модуль

Отличия исходных текстов COM и EXE программ.

1. Сколько сегментов должна содержать COM-программа?

Программы типа .COM состоят из единственного сегмента, в котором размещаются программные коды, данные и стек.

2. EXE-программа?

В программах типа .EXE для собственно программы, данных и стека предусматривают отдельные сегменты.

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

Директива ORG 100h, которая резервирует 256 байт для PSP. Заполнять PSP будет по-прежнему система, но место под него в начале сегмента должен отвести программист, код программы располагается только после этого блока.

```
S:\>lab_com.com
<µt<µtµµ
≈±Cµ0êµN3µ=
s±=  µµ0êµXZY|P|      =!X|PRUWµµ~µµε  µ  ≡ÄL&aµ  <  tµ<µtµ<µtt*µ<µt0<µt6<µt<µ520µ
δ=Éµµ7µµµ  δ3Éµµ?µµ«  δ)ÉµµRµµµ  δµÉµµ\µµµ  δ§ÉµµfµµÉ  δδÉµµmµµâ  δµÉµµ|0=µ!µ6êµâ  µµ0  â  µ
â-µG  µµêµµf  èµµµ6fµâ  µ!µ4  µµfµµS  èµµµ6µµµ.µèDµâ  µµµµµµµ²µµµµµ6  ^ZX|L=µ!µPC
Éµµ7µµµ  δ3Éµµ?µµ«  δ0ÉµµRµµµ  δµÉµµ\µµµ  δ§ÉµµfµµÉ  δδÉµµmµµâ  δµÉµµ|0=µ!µ6êµâ  µµ0  â  µµâ-
µG  µµêµµf  èµµµ6fµâ  µ!µ4  µµfµµS  èµµµ6µµµ.µèDµâ  µµµµµµµ²µµµµµ6  ^ZX|L=µ!µPC
Éµµ\µµµ  δ§ÉµµfµµÉ  δδ000000â  δµÉµµ|0=µ!µ6êµâ  µµ0  â  µµâ-µG  µµêµµf  èµµµ6fµâ  µ!µ4  µµfµµS  è
µµµ6µµµ.µèDµâ  µµµµµµµ²µµµµµ6  ^ZX|L=µ!µPC
```

Рисунок 5 – .COM модуль без директивы 100h

Директива ASSUME, ставящая в соответствие адрес сегмента программы сегментам кода и данных.

```
lab_com.asm(11): error A2062: Missing or unreachable CS
lab_com.asm(15): error A2062: Missing or unreachable CS
lab_com.asm(28): error A2062: Missing or unreachable CS
lab_com.asm(48): error A2062: Missing or unreachable CS
lab_com.asm(56): error A2062: Missing or unreachable CS
lab_com.asm(68): error A2062: Missing or unreachable CS
lab_com.asm(75): error A2062: Missing or unreachable CS
lab_com.asm(83): error A2062: Missing or unreachable CS
lab_com.asm(105): error A2062: Missing or unreachable CS
lab_com.asm(105): error A2062: Missing or unreachable CS
lab_com.asm(105): error A2062: Missing or unreachable CS
lab_com.asm(105): error A2062: Missing or unreachable CS
lab_com.asm(105): error A2062: Missing or unreachable CS
lab_com.asm(105): error A2062: Missing or unreachable CS
lab_com.asm(105): error A2062: Missing or unreachable CS
lab_com.asm(107): error A2062: Missing or unreachable CS
lab_com.asm(120): error A2062: Missing or unreachable CS
lab_com.asm(128): error A2062: Missing or unreachable CS

49960 + 455253 Bytes symbol space free

0 Warning Errors
20 Severe Errors
```

Рисунок 6 – .COM модуль без директивы ASSUME

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

Так как в COM-программе все сегментные регистры определяются в момент запуска программы, а не в момент компиляции (ассемблирования), то невозможно использование, например, таких конструкций:

mov ax, DATA или mov ax, CODE

Нельзя использовать команды вида mov <регистр>, seg <имя сегмента>

Например mov ax, seg CODE

Отличия форматов файлов COM и EXE модулей.

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

C:\Users\Adm\Desktop\Apps\Projects\Asm\LAB COM.COM																h	ANSI
0000000000:	EB	64	90	24	0F	3C	09	76	02	04	07	04	30	C3	51	8A	лдж\$<ov♦♦♦0ГQБ
0000000010:	E0	E8	EF	FF	86	C4	B1	04	D2	E8	E8	E6	FF	59	C3	53	аипятД±ТилижяYTS
0000000020:	8A	FC	E8	E9	FF	88	25	4F	88	05	4F	8A	C7	32	E4	E8	Лвийя€%O€OБЗ2ди
0000000030:	DC	FF	88	25	4F	88	05	5B	C3	51	52	50	32	E4	33	D2	Ъя€%O€[ГQRP2дЗТ
0000000040:	B9	0A	00	F7	F1	80	CA	30	88	14	4E	33	D2	3D	0A	00	М чсЪK0€NЗТ=
0000000050:	73	F1	3D	00	00	76	04	0C	30	88	04	58	5A	59	C3	50	sc= v♦♦0€XZYTP
0000000060:	B4	09	CD	21	58	C3	50	52	56	57	8D	16	7E	02	E8	EE	гOH!XГPRVWК~ио
0000000070:	FF	B8	00	F0	8E	C0	26	A0	FE	FF	3C	FF	74	18	3C	FE	яё рНА& ня<ят†<ю
0000000080:	74	1E	3C	FC	74	24	3C	FA	74	2A	3C	F8	74	30	3C	FD	т▲<ьт\$<ьт*<шт0<э
0000000090:	74	36	3C	F9	74	3C	8D	16	32	02	E8	C2	FF	EB	3D	90	т6<шт<К-2иВял=ж
00000000A0:	8D	16	37	02	E8	B8	FF	EB	33	90	8D	16	3F	02	E8	AE	К-7иёялЗжК-?ио
00000000B0:	FF	EB	29	90	8D	16	52	02	E8	A4	FF	EB	1F	90	8D	16	ял)жК-Rииял▼жК-
00000000C0:	5C	02	E8	9A	FF	EB	15	90	8D	16	66	02	E8	90	FF	EB	\иивял\$жК-fииьял
00000000D0:	0B	90	8D	16	6D	02	E8	86	FF	EB	01	90	B4	30	CD	21	жК-миитял@жгOH!
00000000E0:	8D	36	88	02	83	C6	0F	E8	4F	FF	83	C6	03	86	C4	E8	К6€иЖиОяиЖ♥†Ди
00000000F0:	47	FF	8D	16	88	02	E8	66	FF	8A	C7	8D	36	9F	02	83	ГяК-€иифялЗжК€и
0000000100:	C6	13	E8	34	FF	8D	16	9F	02	E8	53	FF	8A	C3	8D	36	Ж!!и4яК-ииСялГжК6
0000000110:	BD	02	E8	F9	FE	89	44	14	83	C6	19	8B	FE	8B	C1	E8	Сишю%DиX!<ю«Ви
0000000120:	FD	FE	8D	16	BD	02	E8	36	FF	5F	5E	5A	58	B4	4C	CD	эюК-Sии6я ^ZXrLH
0000000130:	21	C3	50	43	0D	0A	24	50	43	2F	58	54	0D	0A	24	41	!ГРСи\$PC/XTи\$A
0000000140:	54	20	6F	72	20	50	53	32	20	2E	35	30	2F	36	30	0D	T or PS2 .50/60и
0000000150:	0A	24	50	53	32	20	2E	33	30	0D	0A	24	50	53	32	20	и\$PS2 .30и\$PS2
0000000160:	2E	38	30	0D	0A	24	50	43	6A	72	0D	0A	24	50	43	20	.80и\$PCjии\$PC
0000000170:	43	6F	6E	76	65	72	74	69	62	6C	65	0D	0A	24	50	43	Convertibleи\$PC
0000000180:	20	74	79	70	65	3A	20	24	4D	53	44	4F	53	20	76	65	type: \$MSDOS ve
0000000190:	72	73	69	6F	6E	3A	20	20	2E	20	20	20	0D	0A	24	4F	rsion: . ии\$O
00000001A0:	45	4D	20	73	65	72	69	61	6C	20	6E	75	6D	62	65	72	EM serial number
00000001B0:	3A	20	20	20	20	20	20	20	20	20	0D	0A	24	55	73	65	:
00000001C0:	72	20	73	65	72	69	61	6C	20	6E	75	6D	62	65	72	3A	ии\$Use
00000001D0:	20	20	20	20	20	20	20	20	20	6E	75	6D	62	65	72	3A	r serial number:
									0D	0A	24						ии\$

Рисунок 7 - .COM модуль в шестнадцатеричном виде

COM файл состоит из одного сегмента и, независимо от фактического размера программы, ей выделяется 64 Кбайт адресного пространства, Код начинается с адреса 0h, но при загрузке модуля устанавливается смещение в 100h.

Сначала идут байты, отвечающие за код. Например, коды команд, представленные на рис. 8. Данные начинаются на 130h, второй байт.

0111 E8EFF	CALL	0103	0100 EB64	JMP	0166
0103 240F	AND	AL,0F	0166 50	PUSH	AX
0105 3C09	CMP	AL,09	0167 52	PUSH	DX
0107 7602	JNA	010B	0168 56	PUSH	SI
0109 0407	ADD	AL,07	0169 57	PUSH	DI
010B 0430	ADD	AL,30	016A 8D167E02	LEA	DX,[027E]
010D C3	RET		016E E8EEFF	CALL	015F
010E 51	PUSH	CX	0171 B800F0	MOV	AX,F000
010F 8AE0	MOV	AH,AL	0174 8EC0	MOV	ES,AX

Рисунок 8 – команды .COM модуль

C:\Users\Adm\Desktop\Apps\Projects\Asm\LAB_COM.EXE																		
0000000000:	4D	5A	DB	00	03	00	00	00	20	00	00	00	FF	FF	00	00	MZH ▼	яя
0000000010:	00	00	3F	8E	00	01	00	00	1E	00	00	00	01	00	00	00	?R @ ▲ @	
0000000020:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
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		
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	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	00		
0000000220:	00	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	00		
0000000240:	00	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	00		
0000000260:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000270:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000280:	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:	EB	64	90	24	0F	3C	09	76	02	04	07	04	30	C3	51	8A	лdћ\$0<ov	♦♦♦0ГQБ
0000000310:	E0	E8	EF	FF	86	C4	B1	04	D2	E8	E8	E6	FF	59	C3	53	аипя†Д±♦ТииияYTS	
0000000320:	8A	FC	E8	E9	FF	88	25	4F	88	05	4F	8A	C7	32	E4	E8	Ьыййя€%0€#0Б32ди	
0000000330:	DC	FF	88	25	4F	88	05	5B	C3	51	52	50	32	E4	33	D2	бя€%0€# [ГQRP2д3Т	
0000000340:	B9	0A	00	F7	F1	80	CA	30	88	14	4E	33	D2	3D	0A	00	№ чсЪK0€TN3T=	
0000000350:	73	F1	3D	00	00	76	04	0C	30	88	04	58	5A	59	C3	50	sc= v♦+0€♦XZYTP	
0000000360:	B4	09	CD	21	58	C3	50	52	56	57	8D	16	7E	02	E8	EE	r°H!XTPRVWК~ио	
0000000370:	FF	B8	00	F0	8E	C0	26	A0	FE	FF	3C	FF	74	18	3C	FE	яё рѢа юя<ят†<ю	
0000000380:	74	1E	3C	FC	74	24	3C	FA	74	2A	3C	F8	74	30	3C	FD	t▲<ьт\$<ьт* <шт0<э	
0000000390:	74	36	3C	F9	74	3C	8D	16	32	02	E8	C2	FF	EB	3D	90	t6<шт<К-2иВял=ћ	
00000003A0:	8D	16	37	02	E8	B8	FF	EB	33	90	8D	16	3F	02	E8	AE	К-7иёял3ћК-?ио	
00000003B0:	FF	EB	29	90	8D	16	52	02	E8	A4	FF	EB	1F	90	8D	16	ял)ћК-Рииял▼ћК-	
00000003C0:	5C	02	E8	9A	FF	EB	15	90	8D	16	66	02	E8	90	FF	EB	\иивял\$ћК-fиИћял	

Рисунок 9 - «Плохой» .EXE модуль в шестнадцатеричном виде

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

В «плохом» EXE файле данные и код содержатся в одном сегменте. Код располагается с адреса 300h. С адреса 0h располагается таблица разметки. Также 100h резервируются командой ORG 100h.

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

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 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
0000000210:	24 0F 3C 09 76 02 04 07	04 30 C3 51 8A E0 E8 EF	00<ov0000000220:
FF 86 C4 B1 04 D2 E8 E8	E6 FF 59 C3 53 8A FC E8	0000000230:	0000000230:
E9 FF 88 25 4F 88 05 4F	8A C7 32 E4 E8 DC FF 88	0000000240:	0000000240:
25 4F 88 05 5B C3 51 52	50 32 E4 33 D2 B9 0A 00	0000000250:	0000000250:
F7 F1 80 CA 30 88 14 4E	33 D2 3D 0A 00 73 F1 3D	0000000260:	0000000260:
00 00 76 04 0C 30 88 04	58 5A 59 C3 50 B4 09 CD	0000000270:	0000000270:
21 58 C3 1E 33 C0 50 B8	14 00 8E D8 8D 16 4C 00	0000000280:	0000000280:
E8 E9 FF B8 00 F0 8E C0	26 A0 FE FF 3C FF 74 18	0000000290:	0000000290:
3C FE 74 1E 3C FC 74 24	3C FA 74 2A 3C F8 74 30	00000002A0:	00000002A0:
3C FD 74 36 3C F9 74 3C	8D 16 00 00 E8 BD FF EB	00000002B0:	00000002B0:
3D 90 8D 16 05 00 E8 B3	FF EB 33 90 8D 16 0D 00	00000002C0:	00000002C0:
E8 A9 FF EB 29 90 8D 16	20 00 E8 9F FF EB 1F 90	00000002D0:	00000002D0:
8D 16 2A 00 E8 95 FF EB	15 90 8D 16 34 00 E8 8B	00000002E0:	00000002E0:
FF EB 0B 90 8D 16 3B 00	E8 81 FF EB 01 90 B4 30	00000002F0:	00000002F0:
CD 21 8D 36 56 00 83 C6	0F E8 4A FF 83 C6 03 86	0000000300:	0000000300:
C4 E8 42 FF 8D 16 56 00	E8 61 FF 8A C7 8D 36 6D	0000000310:	0000000310:
00 83 C6 13 E8 2F FF 8D	16 6D 00 E8 4E FF 8A C3	0000000320:	0000000320:
8D 36 8B 00 E8 F4 FE 89	44 14 83 C6 19 8B FE 8B	0000000330:	0000000330:
C1 E8 F8 FE 8D 16 8B 00	E8 31 FF B4 4C CD 21 CB	0000000340:	0000000340:
50 43 0D 0A 24 50 43 2F	58 54 0D 0A 24 41 54 20	0000000350:	0000000350:
6F 72 20 50 53 32 20 2E	35 30 2F 36 30 0D 0A 24	0000000360:	0000000360:
50 53 32 20 2E 33 30 0D	0A 24 50 53 32 20 2E 38	0000000370:	0000000370:
30 0D 0A 24 50 43 6A 72	0D 0A 24 50 43 20 43 6F	0000000380:	0000000380:
6E 76 65 72 74 69 62 6C	65 0D 0A 24 50 43 20 74	0000000390:	0000000390:
79 70 65 3A 20 24 4D 53	44 4F 53 20 76 65 72 73	00000003A0:	00000003A0:
69 6F 6E 3A 20 20 2E 20	20 20 0D 0A 24 4F 45 4D	00000003B0:	00000003B0:
20 73 65 72 69 61 6C 20	6E 75 6D 62 65 72 3A 20	00000003C0:	00000003C0:
20 20 20 20 20 20 20 20	0D 0A 24 55 73 65 72 20		

Рисунок 10 - «Хороший» .EXE модуль в шестнадцатеричном виде

В «хорошем» файле EXE содержится информация для загрузчика, сегмент стека, сегмент данных и сегмент кода. Код располагается с адреса 210h в отличие от 300h в «плохом» .EXE файле.

Загрузка COM модуля в основную память.



Рисунок 11 – Образ памяти программы типа .COM

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

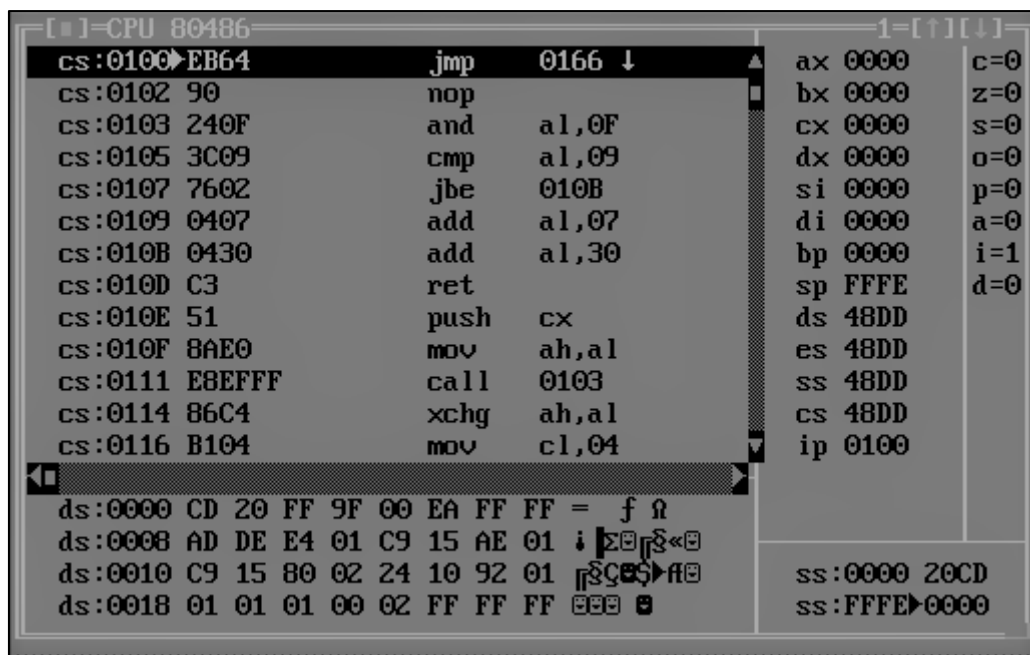


Рисунок 12 – Загрузка .COM модуля в память

После загрузки COM-программы в память сегментные регистры указывают на начало PSP. Код располагается с адреса 100h, IP = 0100h.

2. Что располагается с 0 адреса?

С нулевого адреса располагается адрес начала PSP.

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

Сегментные регистры имеют значение 48DDh. Они указывают на начало PSP, что продемонстрировано на рис. 11.

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

Стек определяется автоматически, указатель стека устанавливается на конец сегмента. Если для программы размер сегмента в 64КБ является достаточным, то DOS устанавливает в регистре SP адрес конца сегмента – FFFEh, что также можно увидеть на рис. 11. Адреса расположены в диапазоне от FFFEh до 0000h.

Загрузка «хорошего» EXE модуля в основную память.



Рисунок 13 – Образ памяти программы типа .EXE

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

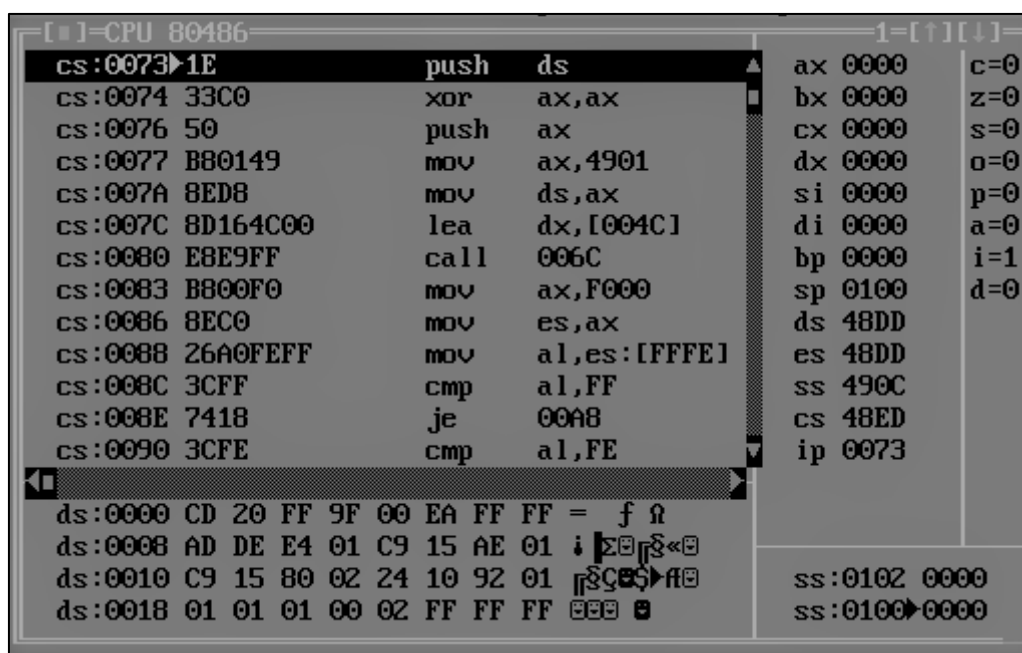


Рисунок 14 – Загрузка «хорошего» .EXE модуля в память

Для PSP и программы выделяется блок памяти. Система, загрузив программу в память, инициализирует сегментные регистры, так что регистры DS и ES указывают на начало PSP(48DDh), CS - на начало сегмента команд(48EDh), а SS - на начало сегмента стека(490Ch). В указатель стека SP кладется смещение конца сегмента стека, а в указатель команд IP загружается смещение точки входа в программу. Таким образом, после загрузки программы в память адресуемыми оказываются все сегменты, кроме сегмента (или сегментов) данных.

2. На что указывают регистры DS и ES?

DS и ES указывают на начало PSP. Инициализация регистра DS в первых строках программы позволяет сделать адресуемым и сегмент данных.

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 указатель стека SP - смещение конца сегмента стека.

4. Как определяется точка входа?

При загрузке программы в указатель команд IP загружается смещение точки входа в программу (которое берется из операнда директивы END).

Вывод.

В ходе работы было проведено исследование различий в структурах исходных текстов модулей .COM и .EXE, структур файлов загрузочных модулей и способов их загрузки в основную память.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ. LAB_COM.COM

```
lab segment
assume cs:lab, ds:lab, es:nothing, ss:nothing
org 100h
main: jmp processing

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
ax               ;байт в al переводится в два символа 16 числа в
                 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 цифра
                 ret
byte_to_hex      endp

word_to_hex      proc near
                 ;перевод в 16 ss 16 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
                 xor ah, ah
                 call byte_to_hex
                 mov [di], ah
                 dec di
                 mov [di], al
                 pop bx
                 ret
word_to_hex      endp
```

```

byte_to_dec      proc near
                  ;перевод в 10 сс, si - адрес поля младшей цифры
                  push cx
                  push dx
                  push ax
                  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 ax, 00h
                  jbe end_l
                  or al, 30h
                  mov [si], al

end_l:
                  pop ax
                  pop dx
                  pop cx
                  ret

byte_to_dec      endp

print proc near
                  push ax
                  mov ah, 09h
                  int 21h
                  pop ax
                  ret

print endp

processing:
                  push ax
                  push dx
                  push si
                  push di
                  lea dx, pc_arg
                  call print

                  mov ax, 0F000h ;указывает ES на ПЗУ
                  mov es, ax
                  mov al, es:[0FFFEh]

                  irpc case, FECA8D9
                  cmp al, 0F&case&h
                  je type_&case&

```

endm

```
irpc met, FECA8D9
type_&met&:
    lea dx, pc_&met&
    call print
    jmp OS
endm
```

OS:

```
    mov ah, 30h
    int 21h ;al - основная версия, ah - модификация,
bh - OEM, bl:cx - номер пользователя
```

```
    lea si, os_arg
    add si, 15
    call byte_to_dec ;пишется основная версия
    add si, 3
    xchg al, ah
    call byte_to_dec ;пишется модификация
    lea dx, os_arg
    call print
```

OEM:

```
    mov al, bh
    lea si, oem_arg
    add si, 19
    call byte_to_dec
    lea dx, oem_arg
    call print
```

serial_number:

```
    mov al, bl
    lea si, user_arg
    call byte_to_hex
    mov [si+20], ax
    add si, 25
    mov di, si ;ax - число, di - адрес последнего
символа для word_to_hex
    mov ax, cx
    call word_to_hex
    lea dx, user_arg
    call print

    pop di
    pop si
    pop dx
    pop ax
    mov ah, 4ch
    int 21h
    ret
```

```

pc_F db          'PC', 0dh, 0ah, '$'
pc_E db 'PC/XT', 0dh, 0ah, '$'
pc_C db          'AT or PS2 .50/60', 0dh, 0ah, '$'
pc_A db 'PS2 .30', 0dh, 0ah, '$'
pc_8 db 'PS2 .80', 0dh, 0ah, '$'
pc_D db 'PCjr', 0dh, 0ah, '$'
pc_9 db 'PC Convertible', 0dh, 0ah, '$'
pc_arg db 'PC type: ', '$'
os_arg db 'MSDOS version:  .  ', 0dh, 0ah, '$'
oem_arg          db 'OEM serial number:          ', 0dh, 0ah, '$'
user_arg db      'User serial number:          ', 0dh, 0ah, '$'

lab ends
end main

```


ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД ПРОГРАММЫ. LAB_EXE.EXE

```
dosseg
.model small
.stack 100h
.data
pc_F db 'PC', 0dh, 0ah, '$'
pc_E db 'PC/XT', 0dh, 0ah, '$'
pc_C db 'AT or PS2 .50/60', 0dh, 0ah, '$'
pc_A db 'PS2 .30', 0dh, 0ah, '$'
pc_8 db 'PS2 .80', 0dh, 0ah, '$'
pc_D db 'PCjr', 0dh, 0ah, '$'
pc_9 db 'PC Convertible', 0dh, 0ah, '$'
pc_arg db 'PC type: ', '$'
os_arg db 'MSDOS version: . ', 0dh, 0ah, '$'
oem_arg db 'OEM serial number: ', 0dh, 0ah, '$'
user_arg db 'User serial number: ', 0dh, 0ah, '$'
.code
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 16 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 цифра
    ret
byte_to_hex endp

word_to_hex proc near
    ;перевод в 16 сс 16 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
xor ah, ah
call byte_to_hex
mov [di], ah
dec di
mov [di], al
pop bx
ret
word_to_hex    endp

byte_to_dec    proc near
;перевод в 10 сс, si - адрес поля младшей цифры
push cx
push dx
push ax
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 ax, 00h
jbe end_1
or al, 30h
mov [si], al

end_1:
pop ax
pop dx
pop cx
ret
byte_to_dec    endp

print proc near
push ax
mov ah, 09h
int 21h
pop ax
ret

print endp

processing proc far
push ds
xor ax, ax
push ax

```

```

mov ax, @data
mov ds, ax
lea dx, pc_arg
call print

mov ax, 0F000h ;указывает ES на ПЗУ
mov es, ax
mov al, es:[0FFFEh]

irpc case, FECA8D9
cmp al, 0F&case&h
je type_&case&
endm

irpc met, FECA8D9
type_&met&:
    lea dx, pc_&met&
    call print
    jmp OS
endm

```

OS:

```

mov ah, 30h
int 21h ;al - основная версия, ah - модификация,
bh - OEM, bl:cx - номер пользователя

```

```

lea si, os_arg
add si, 15
call byte_to_dec ;пишется основная версия
add si, 3
xchg al, ah
call byte_to_dec ;пишется модификация
lea dx, os_arg
call print

```

OEM:

```

mov al, bh
lea si, oem_arg
add si, 19
call byte_to_dec
lea dx, oem_arg
call print

```

serial_number:

```

mov al, bl
lea si, user_arg
call byte_to_hex
mov [si+20], ax
add si, 25
mov di, si ;ax - число, di - адрес последнего
символа для word_to_hex

```

```
mov ax, cx  
call word_to_hex  
lea dx, user_arg  
call print
```

```
mov ah, 4ch  
int 21h  
ret
```

```
processing endp  
end processing
```