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

Студентка гр. 7383 _____ Иолшина В.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2019

Постановка задачи.

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

Таблица 1 - Описание процедур:

Название процедуры	**Описание процедуры**
TETR_TO_HEX	вспомогательная функция для работы функции BYTE_TO_HEX, которая переводит из двоичной в шестнадцатеричную систему счисления
BYTE_TO_HEX	вспомогательная функция для работы функции BYTE_TO_HEX, которая переводит байтовое число из регистра AX в шестнадцатеричную систему
WRD_TO_HEX	переводит число из регистра AX в строку в 16 системе счисления, записывая получившееся в DI, начиная с младшей цифры
BYTE_TO_DEC	переводит байт из AL в десятичную систему и записывает получившееся число по адресу SI, начиная с младшей цифры
PRINT	вызывает функцию печати строки
OS_TYPE	печатает тип ОС
VERSION_	печатает версию ОС
OEM_	печатает серийный номер OEM
NUMBER_	печатает серийный номер пользователя

Таблица 2 - Описание структур данных:

Название	**Тип**	**Назначение**
PCTYPE	db	Тип ОС
OSVERS	db	Версия ОС

OEM	db	Серийный номер OEM
NUMBER	db	Серийный номер пользователя
PC	db	PC
PC_XT	db	PC/XT
AT_	db	AT
PS2_30	db	PS2 модель 30
PS2_50	db	PS2 модель 50/60
PS2_80	db	PS2 модель 80
PCjr	db	PCjr
PC_Conv	db	PC Convertible

Действия, выполняемые программой:

1) Определение и вывод на экран типа ОС;

2) Определение и вывод на экран версии ОС;

3) Определение и вывод на экран серийного номера OEM;

4) Определение и вывод на экран серийного номера пользователя;

Результат работы программы показан на рис. 1-3.

Рисунок 1 - Результат выполнения goodfile.exe

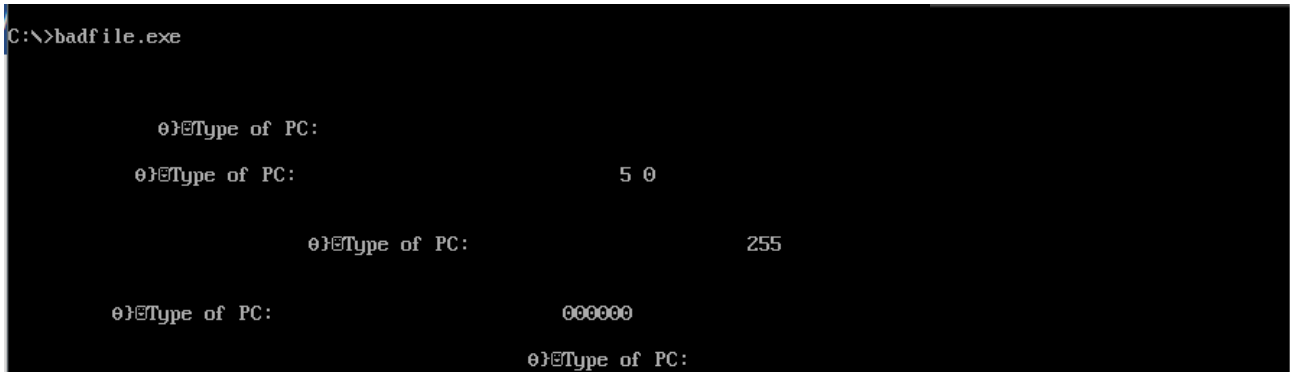


Рисунок 2 - Результат выполнения badfile.exe

Рисунок 3 - Результат выполнения BADFILE.com

Выводы.

В ходе выполнения лабораторной работы были исследованы различия в структурах исходных текстов модулей .COM и .EXE, структур файлов загрузочных модулей и способов их загрузки в основную память. Было замечено, что структура EXE программ сложнее структуры COM программ.

Ответы на контрольные вопросы.

1. Сколько сегментов должна содержать COM-программа?

COM-программа должна содержать 1 сегмент.

2. EXE-программа?

EXE-программа должна содержать как минимум 1 сегмент.

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

В COM-программе обязательным является наличие директивы ORG 100h, создающей смещение в 256 байт, так как в первых 256 байтах находится PSP. А также ASSUME, которая ставит в соответствие сегментам CS и DS начало программы, иначе программа, не обнаружив начало сегмента кода, не скомпилируется.

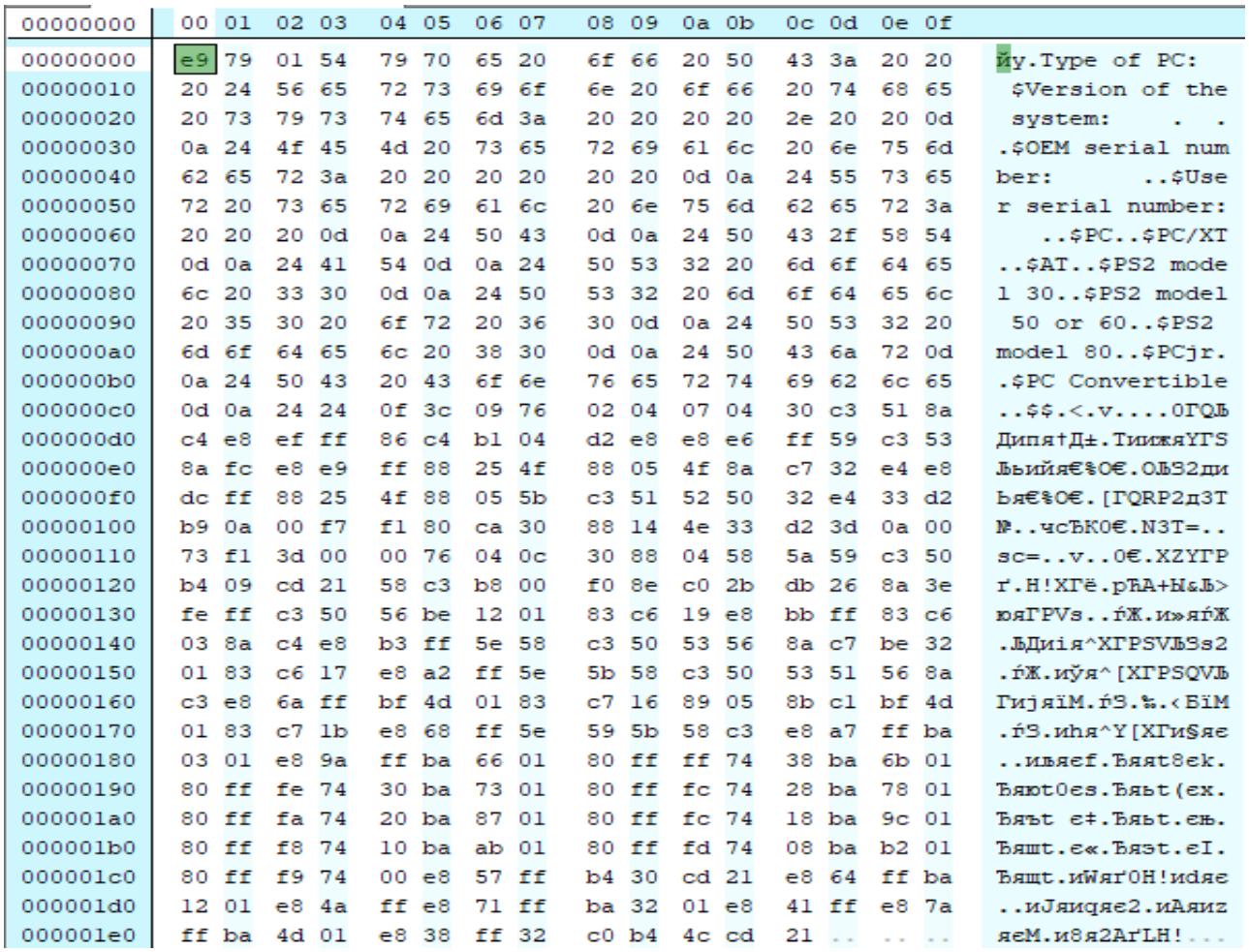
4. Все ли форматы команд можно использовать в COM-программе?

Нет, в COM-программе нельзя использовать команды вида mov register, segment и команды, содержащие дальнюю адресацию, так как в этих командах используется таблица настройки, в которой содержатся адреса сегментов, однако такая таблица существует только в EXE-файлах.

Отличия форматов файлов COM и EXE модулей

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

HEX-представление .COM файла отображено на рис. 4:



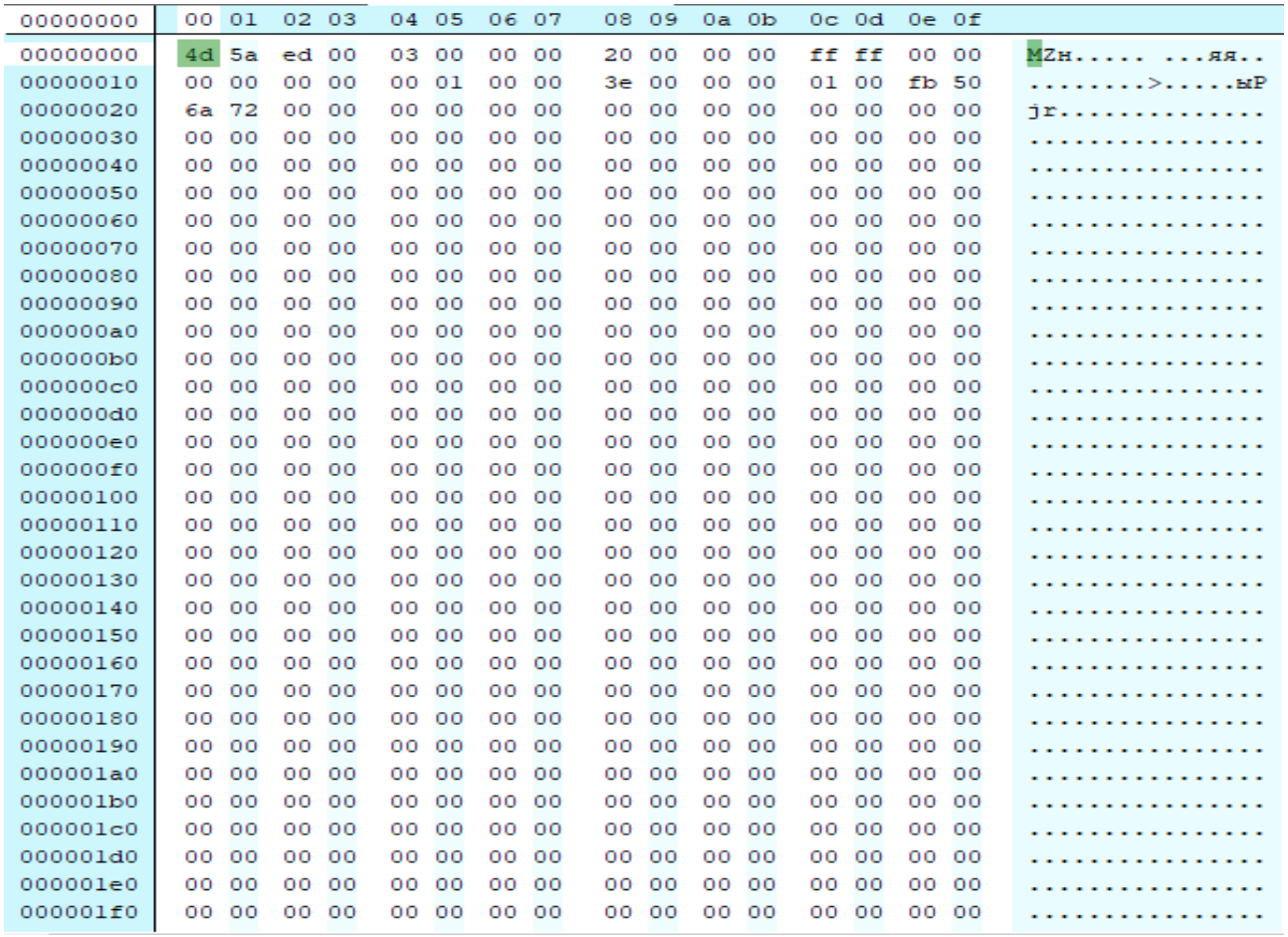
00000000	00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f	
00000000	e9 79 01 54 79 70 65 20 6f 66 20 50 43 3a 20 20	йy.Type of PC:
00000010	20 24 56 65 72 73 69 6f 6e 20 6f 66 20 74 68 65	$Version of the
00000020	20 73 79 73 74 65 6d 3a 20 20 20 20 2e 20 20 0d	system: . .
00000030	0a 24 4f 45 4d 20 73 65 72 69 61 6c 20 6e 75 6d	.$OEM serial num
00000040	62 65 72 3a 20 20 20 20 20 20 0d 0a 24 55 73 65	ber: ..$Use
00000050	72 20 73 65 72 69 61 6c 20 6e 75 6d 62 65 72 3a	r serial number:
00000060	20 20 20 0d 0a 24 50 43 0d 0a 24 50 43 2f 58 54	..$PC..$PC/XT
00000070	0d 0a 24 41 54 0d 0a 24 50 53 32 20 6d 6f 64 65	..$AT..$PS2 mode
00000080	6c 20 33 30 0d 0a 24 50 53 32 20 6d 6f 64 65 6c	l 30..$PS2 model
00000090	20 35 30 20 6f 72 20 36 30 0d 0a 24 50 53 32 20	50 or 60..$PS2
000000a0	6d 6f 64 65 6c 20 38 30 0d 0a 24 50 43 6a 72 0d	model 80..$PCjr.
000000b0	0a 24 50 43 20 43 6f 6e 76 65 72 74 69 62 6c 65	.$PC Convertible
000000c0	0d 0a 24 24 0f 3c 09 76 02 04 07 04 30 c3 51 8a	..$$.<.v....0ГQЉ
000000d0	c4 e8 ef ff 86 c4 b1 04 d2 e8 e8 e6 ff 59 c3 53	Дипя†Д±.ТиижяYГS
000000e0	8a fc e8 e9 ff 88 25 4f 88 05 4f 8a c7 32 e4 e8	Љьийя€%О€.ОЉЗ2ди
000000f0	dc ff 88 25 4f 88 05 5b c3 51 52 50 32 e4 33 d2	Ья€%О€.[ГQRP2д3Т
00000100	b9 0a 00 f7 f1 80 ca 30 88 14 4e 33 d2 3d 0a 00	№..чсЪК0€.N3Т=..
00000110	73 f1 3d 00 00 76 04 0c 30 88 04 58 5a 59 c3 50	sc=..v..0€.XZYГP
00000120	b4 09 cd 21 58 c3 b8 00 f0 8e c0 2b db 26 8a 3e	ґ.Н!ХГё.рЋА+Н&Љ>
00000130	fe ff c3 50 56 be 12 01 83 c6 19 e8 bb ff 83 c6	юяГPVs..ѓЖ.и»яѓЖ
00000140	03 8a c4 e8 b3 ff 5e 58 c3 50 53 56 8a c7 be 32	.ЉДиія^ХГPSVЉЗs2
00000150	01 83 c6 17 e8 a2 ff 5e 5b 58 c3 50 53 51 56 8a	.ѓЖ.иўя^[ХГPSQVЉ
00000160	c3 e8 6a ff bf 4d 01 83 c7 16 89 05 8b c1 bf 4d	ГиjяїМ.ѓЗ.‰.‹БїМ
00000170	01 83 c7 1b e8 68 ff 5e 59 5b 58 c3 e8 a7 ff ba	.ѓЗ.иhя^Y[ХГи§яє
00000180	03 01 e8 9a ff ba 66 01 80 ff ff 74 38 ba 6b 01	..иљяєf.Ъяят8єk.
00000190	80 ff fe 74 30 ba 73 01 80 ff fc 74 28 ba 78 01	Ъяю†0єs.Ъяьt(єх.
000001a0	80 ff fa 74 20 ba 87 01 80 ff fc 74 18 ba 9c 01	Ъяъt є‡.Ъяьt.єњ.
000001b0	80 ff f8 74 10 ba ab 01 80 ff fd 74 08 ba b2 01	Ъящt.є«.Ъяэt.єI.
000001c0	80 ff f9 74 00 e8 57 ff b4 30 cd 21 e8 64 ff ba	Ъящt.иWяґ0Н!идяє
000001d0	12 01 e8 4a ff e8 71 ff ba 32 01 e8 41 ff e8 7a	..иJяицяе2.иАяиz
000001e0	ff ba 4d 01 e8 38 ff 32 c0 b4 4c cd 21	яєМ.и8я2АґLН!...

Рисунок 4 - HEX представление .COM файла

COM-файл содержит только машинный код и данные программы. Код располагается с нулевого адреса.

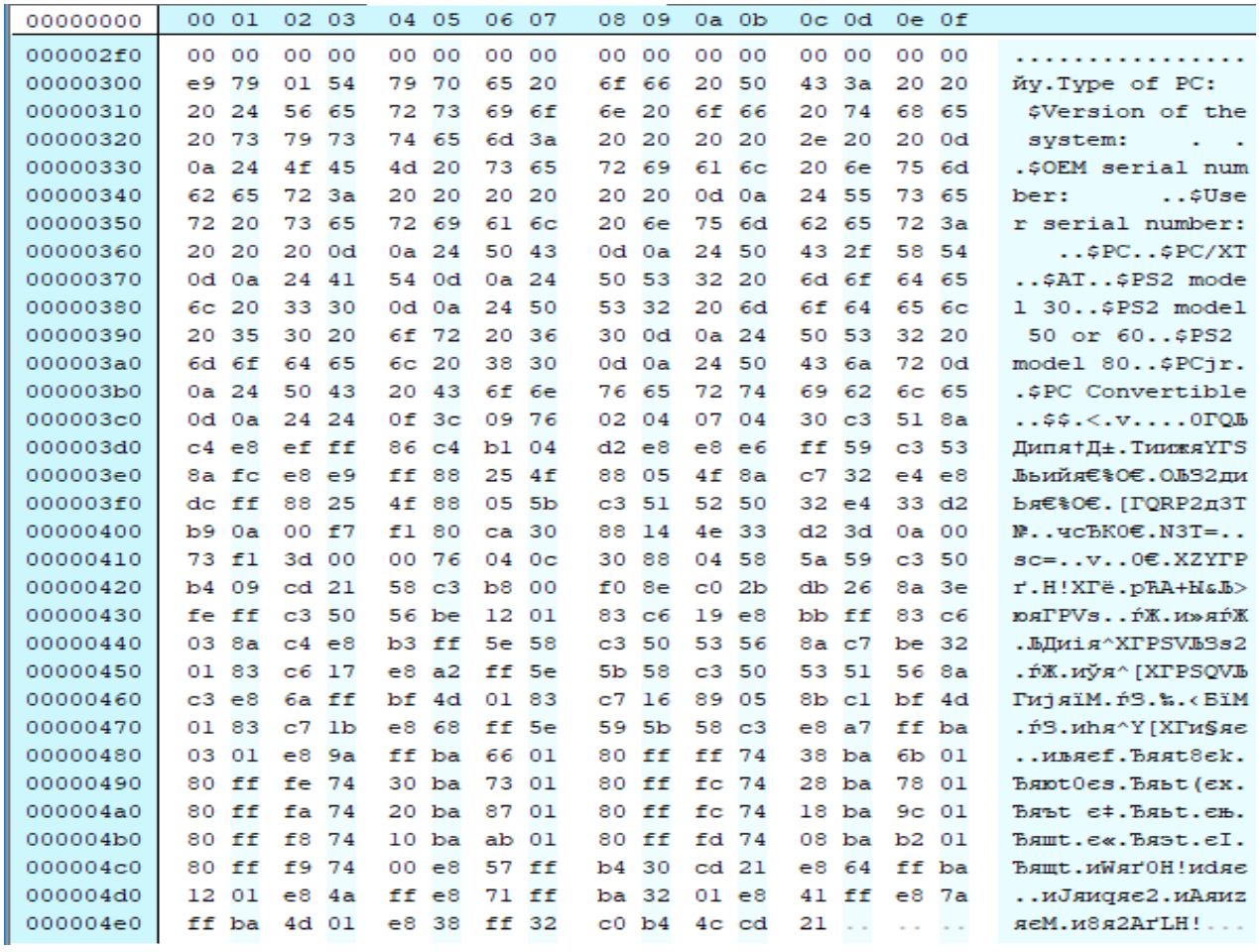
2. Какова структура файла «плохого» EXE? С какого адреса располагается код? Что располагается с адреса 0?

HEX-представление «плохого» .EXE файла отображено на рис. 5-6:



00000000	00 01 02 03	04 05 06 07	08 09 0a 0b	0c 0d 0e 0f	
00000000	4d 5a ed 00	03 00 00 00	20 00 00 00	ff ff 00 00	MZн..... ...яя..
00000010	00 00 00 00	00 01 00 00	3e 00 00 00	01 00 fb 50	........>.....ыP
00000020	6a 72 00 00	00 00 00 00	00 00 00 00	00 00 00 00	jr..............
00000030	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000040	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000050	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000060	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000070	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000080	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000090	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
000000a0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
000000b0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
000000c0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
000000d0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
000000e0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
000000f0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000100	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000110	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000120	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000130	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000140	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000150	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000160	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000170	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000180	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000190	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
000001a0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
000001b0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
000001c0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
000001d0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
000001e0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
000001f0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....

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



00000000	00 01 02 03	04 05 06 07	08 09 0a 0b	0c 0d 0e 0f	
000002f0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000300	e9 79 01 54	79 70 65 20	6f 66 20 50	43 3a 20 20	йy.Type of PC:
00000310	20 24 56 65	72 73 69 6f	6e 20 6f 66	20 74 68 65	$Version of the
00000320	20 73 79 73	74 65 6d 3a	20 20 20 20	2e 20 20 0d	system: . .
00000330	0a 24 4f 45	4d 20 73 65	72 69 61 6c	20 6e 75 6d	.$OEM serial num
00000340	62 65 72 3a	20 20 20 20	20 20 0d 0a	24 55 73 65	ber: ..$Use
00000350	72 20 73 65	72 69 61 6c	20 6e 75 6d	62 65 72 3a	r serial number:
00000360	20 20 20 0d	0a 24 50 43	0d 0a 24 50	43 2f 58 54	..$PC..$PC/XT
00000370	0d 0a 24 41	54 0d 0a 24	50 53 32 20	6d 6f 64 65	..$AT..$PS2 mode
00000380	6c 20 33 30	0d 0a 24 50	53 32 20 6d	6f 64 65 6c	l 30..$PS2 model
00000390	20 35 30 20	6f 72 20 36	30 0d 0a 24	50 53 32 20	50 or 60..$PS2
000003a0	6d 6f 64 65	6c 20 38 30	0d 0a 24 50	43 6a 72 0d	model 80..$PCjr.
000003b0	0a 24 50 43	20 43 6f 6e	76 65 72 74	69 62 6c 65	.$PC Convertible
000003c0	0d 0a 24 24	0f 3c 09 76	02 04 07 04	30 c3 51 8a	..$$.<.v....0ГQЉ
000003d0	c4 e8 ef ff	86 c4 b1 04	d2 e8 e8 e6	ff 59 c3 53	ДипятД±.ТиижяYГS
000003e0	8a fc e8 e9	ff 88 25 4f	88 05 4f 8a	c7 32 e4 e8	Љьийя€%О€.ОЉ2ди
000003f0	dc ff 88 25	4f 88 05 5b	c3 51 52 50	32 e4 33 d2	Ья€%О€.[ГQRP2д3Т
00000400	b9 0a 00 f7	f1 80 ca 30	88 14 4e 33	d2 3d 0a 00	№..чсЪК0€.N3Т=..
00000410	73 f1 3d 00	00 76 04 0c	30 88 04 58	5a 59 c3 50	sc=..v..0€.XZYГP
00000420	b4 09 cd 21	58 c3 b8 00	f0 8e c0 2b	db 26 8a 3e	ґ.Н!ХГё.рЋА+Н&Љ>
00000430	fe ff c3 50	56 be 12 01	83 c6 19 e8	bb ff 83 c6	юяГPVs..ѓЖ.и»яѓЖ
00000440	03 8a c4 e8	b3 ff 5e 58	c3 50 53 56	8a c7 be 32	.ЉДиія^XГPSVЉЗs2
00000450	01 83 c6 17	e8 a2 ff 5e	5b 58 c3 50	53 51 56 8a	.ѓЖ.иўя^[XГPSQVЉ
00000460	c3 e8 6a ff	bf 4d 01 83	c7 16 89 05	8b c1 bf 4d	ГиjяїМ.ѓЗ.‰.‹БїМ
00000470	01 83 c7 1b	e8 68 ff 5e	59 5b 58 c3	e8 a7 ff ba	.ѓЗ.иhя^Y[XГи§яє
00000480	03 01 e8 9a	ff ba 66 01	80 ff ff 74	38 ba 6b 01	..иљяєf.Ъяят8єk.
00000490	80 ff fe 74	30 ba 73 01	80 ff fc 74	28 ba 78 01	Ъяют0єs.Ъяьт(єх.
000004a0	80 ff fa 74	20 ba 87 01	80 ff fc 74	18 ba 9c 01	Ъяът є‡.Ъяьт.єњ.
000004b0	80 ff f8 74	10 ba ab 01	80 ff fd 74	08 ba b2 01	Ъяшт.є«.Ъяэт.єІ.
000004c0	80 ff f9 74	00 e8 57 ff	b4 30 cd 21	e8 64 ff ba	Ъящт.иWяґ0Н!идяє
000004d0	12 01 e8 4a	ff e8 71 ff	ba 32 01 e8	41 ff e8 7a	..иJяиqяє2.иАяиz
000004e0	ff ba 4d 01	e8 38 ff 32	c0 b4 4c cd	21	яєМ.и8я2АґLН!...

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

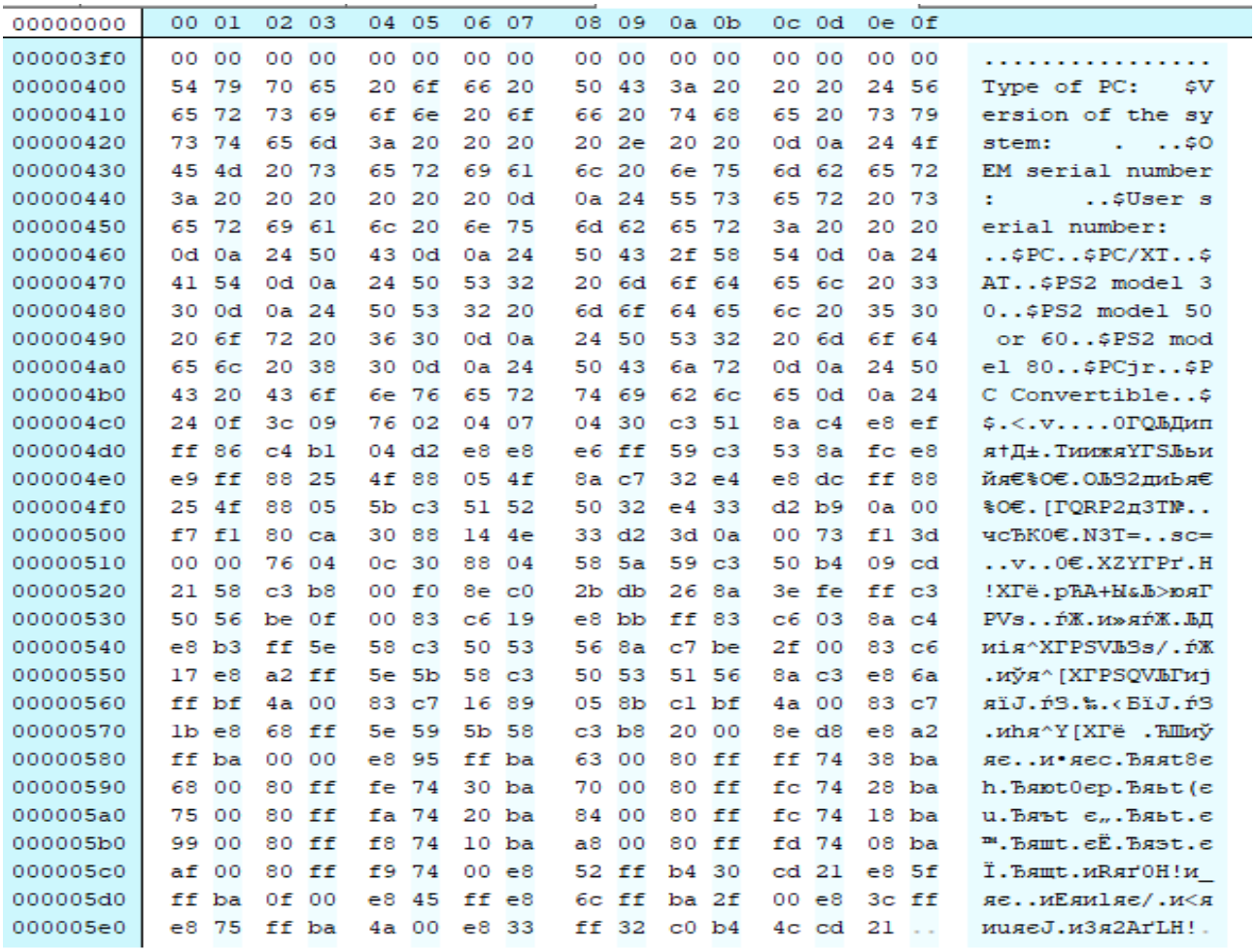
В «плохом» EXE код и данные не разделены по сегментам, а перемешаны. Код располагается с адреса 300h, т.к. заголовок занимает 200h байт и команда ORG 100h «сдвигает» код на дополнительные 100h. Значит, с нулевого адреса располагается заголовок. В первых двух байтах можно увидеть символы MZ, означающие, что формат файла – 16-битный и его следует запускать в соответствии со структурой EXE-файлов. За заголовком следует таблица настройки, при помощи которых строится данный EXEфайл и зарезервированные директивой ORG 100h байт.

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

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

00000000	00 01 02 03	04 05 06 07	08 09 0a 0b	0c 0d 0e 0f	
00000000	4d 5a ef 01	03 00 01 00	20 00 00 00	ff ff 00 00	MZп..... ...яя..
00000010	00 02 00 00	b9 00 2c 00	3e 00 00 00	01 00 fb 50	....№.,.>.....ыP
00000020	6a 72 00 00	00 00 00 00	00 00 00 00	00 00 00 00	jr..............
00000030	00 00 00 00	00 00 00 00	00 00 00 00	00 00 ba 00	..............є.
00000040	2c 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	,...............
00000050	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000060	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000070	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000080	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000090	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
000000a0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
000000b0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
000000c0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
000000d0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
000000e0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
000000f0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000100	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000110	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000120	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000130	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000140	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000150	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000160	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000170	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000180	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
00000190	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
000001a0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
000001b0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
000001c0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
000001d0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
000001e0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....
000001f0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	....

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



00000000	00 01 02 03	04 05 06 07	08 09 0a 0b	0c 0d 0e 0f	
000003f0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	.....
00000400	54 79 70 65	20 6f 66 20	50 43 3a 20	20 20 24 56	Type of PC: $V
00000410	65 72 73 69	6f 6e 20 6f	66 20 74 68	65 20 73 79	ersion of the sy
00000420	73 74 65 6d	3a 20 20 20	20 2e 20 20	0d 0a 24 4f	stem: . ..$O
00000430	45 4d 20 73	65 72 69 61	6c 20 6e 75	6d 62 65 72	EM serial number
00000440	3a 20 20 20	20 20 20 0d	0a 24 55 73	65 72 20 73	: ..$User s
00000450	65 72 69 61	6c 20 6e 75	6d 62 65 72	3a 20 20 20	erial number:
00000460	0d 0a 24 50	43 0d 0a 24	50 43 2f 58	54 0d 0a 24	..$PC..$PC/XT..$
00000470	41 54 0d 0a	24 50 53 32	20 6d 6f 64	65 6c 20 33	AT..$PS2 model 3
00000480	30 0d 0a 24	50 53 32 20	6d 6f 64 65	6c 20 35 30	0..$PS2 model 50
00000490	20 6f 72 20	36 30 0d 0a	24 50 53 32	20 6d 6f 64	or 60..$PS2 mod
000004a0	65 6c 20 38	30 0d 0a 24	50 43 6a 72	0d 0a 24 50	el 80..$PCjr..$P
000004b0	43 20 43 6f	6e 76 65 72	74 69 62 6c	65 0d 0a 24	C Convertible..$
000004c0	24 0f 3c 09	76 02 04 07	04 30 c3 51	8a c4 e8 ef	$.<.v....0ГQЊДип
000004d0	ff 86 c4 b1	04 d2 e8 e8	e6 ff 59 c3	53 8a fc e8	яtД±.ТиижяYГSЉьи
000004e0	e9 ff 88 25	4f 88 05 4f	8a c7 32 e4	e8 dc ff 88	йяЂ%О€.ОЊ2диЬяЂ
000004f0	25 4f 88 05	5b c3 51 52	50 32 e4 33	d2 b9 0a 00	%О€.[ГQRP2д3Т№..
00000500	f7 f1 80 ca	30 88 14 4e	33 d2 3d 0a	00 73 f1 3d	чсЂК0€.N3Т=..sc=
00000510	00 00 76 04	0c 30 88 04	58 5a 59 c3	50 b4 09 cd	..v..0€.XZYГPґ.Н
00000520	21 58 c3 b8	00 f0 8e c0	2b db 26 8a	3e fe ff c3	!XГё.рЋА+Н&Љ>юяГ
00000530	50 56 be 0f	00 83 c6 19	e8 bb ff 83	c6 03 8a c4	PVs..ѓЖ.и»яѓЖ.ЊД
00000540	e8 b3 ff 5e	58 c3 50 53	56 8a c7 be	2f 00 83 c6	иія^XГPSVЉЗs/.ѓЖ
00000550	17 e8 a2 ff	5e 5b 58 c3	50 53 51 56	8a c3 e8 6a	.иўя^[XГPSQVЉГиj
00000560	ff bf 4a 00	83 c7 16 89	05 8b c1 bf	4a 00 83 c7	яїJ.ѓЗ.‰.‹БїJ.ѓЗ
00000570	1b e8 68 ff	5e 59 5b 58	c3 b8 20 00	8e d8 e8 a2	.иhя^Y[XГё .ЋШиў
00000580	ff ba 00 00	e8 95 ff ba	63 00 80 ff	ff 74 38 ba	яє..и•яєс.Ъяят8є
00000590	68 00 80 ff	fe 74 30 ba	70 00 80 ff	fc 74 28 ba	h.Ъяют0ер.Ъяьт(є
000005a0	75 00 80 ff	fa 74 20 ba	84 00 80 ff	fc 74 18 ba	u.Ъяът є„.Ъяьт.є
000005b0	99 00 80 ff	f8 74 10 ba	a8 00 80 ff	fd 74 08 ba	™.Ъящт.єЁ.Ъяэт.є
000005c0	af 00 80 ff	f9 74 00 e8	52 ff b4 30	cd 21 e8 5f	Ї.Ъящт.иRяґ0Н!и_
000005d0	ff ba 0f 00	e8 45 ff e8	6c ff ba 2f	00 e8 3c ff	яє..иЕяиляє/.и<я
000005e0	e8 75 ff ba	4a 00 e8 33	ff 32 c0 b4	4c cd 21 ..	иuяєJ.и3я2АґLН!.

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

В отличие от «плохого» EXE, в «хорошем» код, стек и данные выделены в отдельные сегменты. С нулевого адреса все так же находится заголовок с таблицей настроек. С адреса 200h в «хорошем» EXE-файле идет сегмент стека, 8 под который дополнительно выделено 20 байт, а с адреса 220h располагается код программы. Для «хорошего» EXE в директиве org 100h нет необходимости, т.к. загрузчик автоматически расположит программу после PSP.

Загрузка COM модуля в основную память

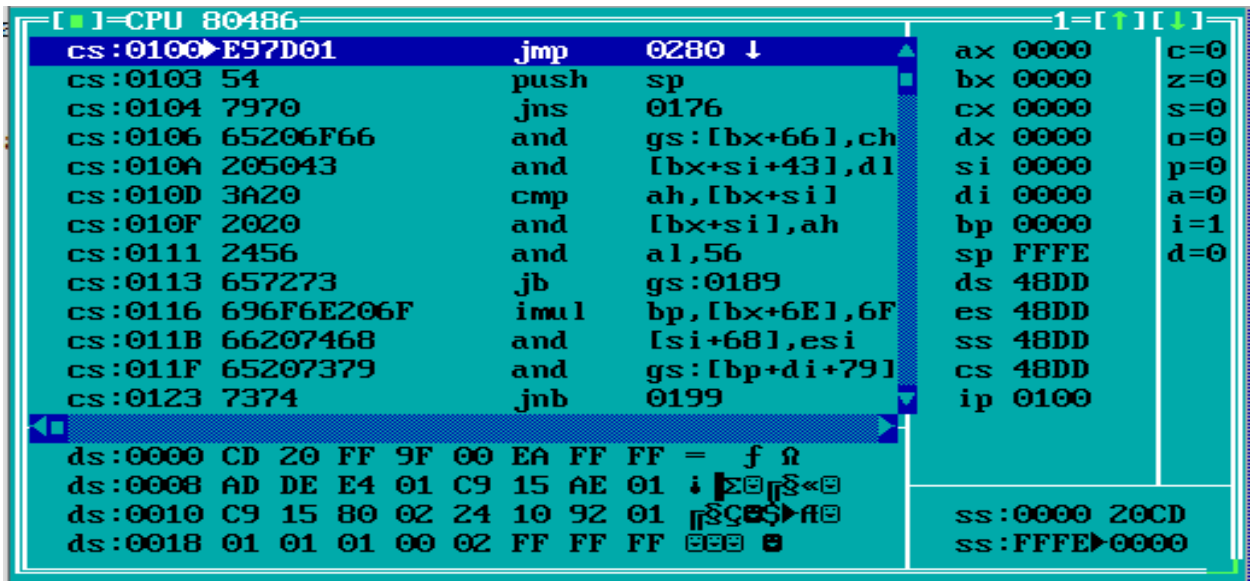


Рисунок 9 – Результат загрузки .COM в основную память

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

Формат загрузки модуля COM:

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

2. Что располагается с адреса 0?

Сегмент PSP.

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

Все сегментные регистры (CS, DS, ES, SS) в данном случае равны 48DD и указывают на начало PSP.

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

DOS автоматически определяет стек. SS указывает на начало сегмента, SP=FFFEh – на его конец. Стек занимает весь сегмент COM-программы. При добавлении данных в стек, адрес на который указывает SP уменьшается и данные записываются в конце выделенной области памяти, то есть заполняется снизу вверх. Стек может дойти до кода или данных программы при достаточном количестве элементов. В COM-модулях стек растёт от старших адресов к младшим. Адреса могут быть расположены в диапазоне 0000h - FFFEh.

Загрузка «хорошего» EXE модуля в основную память

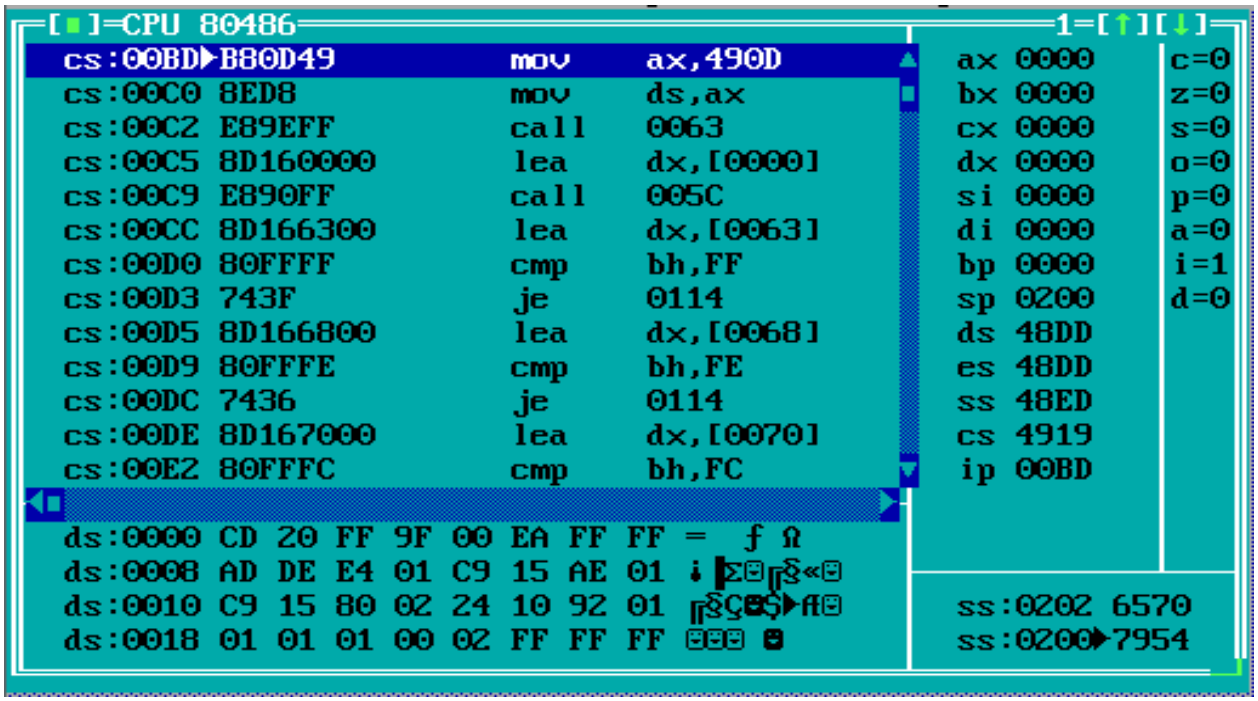


Рисунок 10 – Результат загрузки «хорошего» .EXE в основную память

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

Сначала помещается PSP, а дальше устанавливаются сегментные регистры. DS и ES устанавливаются на начало сегмента PSP, SS=48ED – начало сегмента стека, CS=4919 – начало сегмента команд. В IP загружается смещение точки входа в программу, которая берётся из метки после директивы END.

2. На что указывают регистры DS и ES?

DS и ES указывают на начало PSP.

3. Как определяется стек?

Стек определяется с помощью директивы STACK, после которой задаётся размер стека. При запуске программы в SS заносится адрес сегмента стека, его начало, а в SP – адрес его вершины.

4. Как определяется точка входа?

Точка входа в программу определяется с помощью директивы END. После этой директивы указывается метка (адрес), куда переходит программа при запуске.

ПРИЛОЖЕНИЕ А

GOODFILE.ASM

```
STACK SEGMENT STACK
       DW 0100h DUP(?)

STACK ENDS

DATA SEGMENT

;Данные

PCTYPE       db 'Type of PC:   ','$'
OSVERS       db 'Version of the system:    .  ',0Dh,0Ah,'$'
OEM                db 'OEM serial number:      ',0Dh,0Ah,'$'
NUMBER       db 'User serial number:   ',0Dh,0Ah,'$'

PC                 db 'PC',0Dh,0Ah,'$'
PC_XT        db 'PC/XT',0Dh,0Ah,'$'
AT_                db 'AT',0Dh,0Ah,'$'
PS2_30             db 'PS2 model 30',0Dh,0Ah,'$'
PS2_50             db 'PS2 model 50 or 60',0Dh,0Ah,'$'
PS2_80             db 'PS2 model 80',0Dh,0Ah,'$'
PCjr         db 'PCjr',0Dh,0Ah,'$'
PC_Conv      db 'PC Convertible',0Dh,0Ah,'$'
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA, ES:NOTHING, SS:STACK

;Процедуры
;-----------------------------
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
             push   cx
             mov           al,ah
             call   TETR_TO_HEX
             xchg   al,ah
             mov           cl,4
             shr           al,cl
             call   TETR_TO_HEX
             pop           cx
             ret
BYTE_TO_HEX        ENDP
;--------------------------
WRD_TO_HEX         PROC   near ;перевод в шестнадцатеричную сс шестнадцатибитового
числа
             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
WRD_TO_HEX        ENDP
;----------------------------
BYTE_TO_DEC       PROC  near ;перевод байтового числа в дестеричную сс
            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
;----------------------------
OS_TYPE PROC near ; нахождение типа PC
             mov ax, 0F000h
             mov es, ax
           sub bx, bx
             mov bh, es:[0FFFEh]
             ret
OS_TYPE           ENDP
;----------------------------
VERSION_          PROC  near ; нахождение версии системы
            push  ax
            push  si
            lea         si,OSVERS
            add         si,19h
            call  BYTE_TO_DEC
            add         si,3h
            mov   al,ah
            call  BYTE_TO_DEC
```

```
            pop    si
            pop    ax
            ret
VERSION_          ENDP
;-----------------------------
OEM_        PROC   near ; нахождение серийного номера OEM
            push   ax
            push   bx
            push   si
            mov    al,bh
            lea          si,OEM
            add          si,17h
            call   BYTE_TO_DEC
            pop          si
            pop          bx
            pop          ax
            ret
OEM_        ENDP
;-----------------------------
NUMBER_     PROC   near ; нахождение серийного номера пользователя
            push   ax
            push   bx
            push   cx
            push   si
            mov          al,bl
            call   BYTE_TO_HEX
            lea          di,NUMBER
            add          di,22
            mov          [di],AX
            mov          ax,cx
            lea          di,NUMBER
            add          di,27
            call   WRD_TO_HEX
            pop          si
            pop          cx
            pop          bx
            pop          ax
            ret
NUMBER_     ENDP
;-----------------------------

BEGIN:
            mov ax, DATA
            mov ds, ax
            ;mov bx, ds
            call   OS_TYPE
            lea          dx,PCTYPE
            call   PRINT
            lea    dx, PC
            cmp bh, 0FFh
            je     output

            lea    dx, PC_XT
            cmp bh, 0FEh
            je     output

            lea    dx, AT_
            cmp bh, 0FCh
            je     output

            lea    dx, PS2_30
```

```
                cmp bh, 0FAh
                je    output

                lea   dx, PS2_50
                cmp bh, 0FCh
                je    output

                lea   dx, PS2_80
                cmp bh, 0F8h
                je    output

                lea   dx, PCjr
                cmp bh, 0FDh
                je    output

                lea   dx, PC_Conv
                cmp bh, 0F9h
                je    output
output:
                call  PRINT

                mov           ah,30h
                int           21h
                call  VERSION_ ; нахождение версии системы
                lea           dx,OSVERS
                call  PRINT

                call  OEM_ ; нахождение серийного номера OEM
                lea           dx,OEM
                call  PRINT

                call  NUMBER_ ; нахождение серийного номера пользователя
                lea           dx,NUMBER
                call  PRINT

                xor al, al
                mov ah, 4ch
                int 21h
CODE ENDS
END BEGIN
```

ПРИЛОЖЕНИЕ Б
BADFILE.ASM

```
TESTPC SEGMENT
             ASSUME CS:TESTPC,   DS:TESTPC,   ES:NOTHING,  SS:NOTHING
             ORG          100H
START: JMP           BEGIN
;Данные
PCTYPE       db 'Type of PC:   ','$'
OSVERS       db 'Version of the system:    .  ',0Dh,0Ah,'$'
OEM                  db 'OEM serial number:      ',0Dh,0Ah,'$'
NUMBER       db 'User serial number:   ',0Dh,0Ah,'$'

PC                   db 'PC',0Dh,0Ah,'$'
PC_XT        db 'PC/XT',0Dh,0Ah,'$'
AT_                  db 'AT',0Dh,0Ah,'$'
PS2_30               db 'PS2 model 30',0Dh,0Ah,'$'
PS2_50               db 'PS2 model 50 or 60',0Dh,0Ah,'$'
PS2_80               db 'PS2 model 80',0Dh,0Ah,'$'
PCjr         db 'PCjr',0Dh,0Ah,'$'
PC_Conv      db 'PC Convertible',0Dh,0Ah,'$'

;Процедуры
;----------------------------
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 шестнадцатеричной сс
             push    cx
             mov             al,ah
             call    TETR_TO_HEX
             xchg    al,ah
             mov             cl,4
             shr             al,cl
             call    TETR_TO_HEX
             pop             cx
             ret
BYTE_TO_HEX          ENDP
;----------------------------
WRD_TO_HEX           PROC   near ; шестнадцатибитовое число в шестнадцатеричную сс
             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
```

```
WRD_TO_HEX          ENDP
;----------------------------
BYTE_TO_DEC         PROC   near ; перевод в десятичную сс
              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
;----------------------------
OS_TYPE PROC near ; нахождение типа PC
               mov ax, 0F000h
               mov es, ax
            sub bx, bx
               mov bh, es:[0FFFEh]
               ret
OS_TYPE             ENDP
;----------------------------
VERSION_            PROC   near ; нахождение версии системы
              push   ax
              push   si
              lea           si,OSVERS
              add           si,19h
              call   BYTE_TO_DEC
              add           si,3h
              mov    al,ah
              call   BYTE_TO_DEC
              pop    si
              pop    ax
              ret
VERSION_            ENDP
;----------------------------
OEM_          PROC   near ; нахождение серийного номера OEM
              push   ax
              push   bx
              push   si
              mov    al,bh
```

```
            lea         si,OEM
            add         si,17h
            call  BYTE_TO_DEC
            pop         si
            pop         bx
            pop         ax
            ret
OEM_        ENDP
;-----------------------------
NUMBER_     PROC  near ; нахождение серийного номера пользователя
            push  ax
            push  bx
            push  cx
            push  si
            mov         al,bl
            call  BYTE_TO_HEX
            lea         di,NUMBER
            add         di,22
            mov         [di],AX
            mov         ax,cx
            lea         di,NUMBER
            add         di,27
            call  WRD_TO_HEX
            pop         si
            pop         cx
            pop         bx
            pop         ax
            ret
NUMBER_     ENDP
;-----------------------------

BEGIN:
            call  OS_TYPE
            lea         dx,PCTYPE
            call  PRINT
            lea   dx, PC
            cmp bh, 0FFh
            je    output

            lea   dx, PC_XT
            cmp bh, 0FEh
            je    output

            lea   dx, AT_
            cmp bh, 0FCh
            je    output

            lea   dx, PS2_30
            cmp bh, 0FAh
            je    output

            lea   dx, PS2_50
            cmp bh, 0FCh
            je    output

            lea   dx, PS2_80
            cmp bh, 0F8h
            je    output

            lea   dx, PCjr
            cmp bh, 0FDh
```

```
            je    output

            lea   dx, PC_Conv
            cmp bh, 0F9h
            je    output
output:
            call  PRINT
            mov         ah,30h
            int         21h
            call  VERSION_ ; нахождение версии системы
            lea         dx,OSVERS
            call  PRINT

            call  OEM_ ; нахождение серийного номера OEM
            lea         dx,OEM
            call  PRINT

            call  NUMBER_ ; нахождение серийного номера пользователя
            lea         dx,NUMBER
            call  PRINT

;Выход в DOS
            xor al, al
            mov ah, 4ch
            int 21h

TESTPC      ENDS
            END START
```