

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

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

Студент гр. 8381

Преподаватель

\_\_\_\_\_

\_\_\_\_\_

Почаев Н.А.

Ефремов М.А.

Санкт-Петербург

2020

### **Цель работы.**

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

### **Задание.**

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

Таблица 1 – Соответствие типа и кода 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

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. Если 0, то < 2.0;
- AH – номер модификации;
- BH – серийный номер OEM (Original Equipment Manufacturer);
- BL:CH – 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 символьную 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 xx.yy, где xx - номер основной версии, а yy - 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 необходимо отладить полученный исходный модуль и получить «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.

### **Выполнение работы.**

Выполнение работы производилось на базе операционной системы Windows 7 (32 bit) в редакторе Notepad++. Сборка и отладка модулей производились с помощью компилятора MASM и отладчика AFD. Также в работе был использован консольный файловый менеджер Far Manager и HEX-редактор HxD.

Был написан текст исходного .COM модуля, который определяет тип РС и информацию о системе. Полученный модуль был отлажен, и в результате были

получены «плохой» .EXE модуль и «хороший» .COM модуль (с помощью пакета Borland Tasm).

Пример сборки «хорошего» .COM модуля показан на рисунке 1.

```
E:\lr1>tasm BAD.asm
Turbo Assembler Version 4.0 Copyright (c) 1988, 1993 Borland International

Assembling file:   BAD.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  450k

E:\lr1>tlink com /t BAD.OBJ
Turbo Link Version 4.01 Copyright (c) 1991 Borland International
Fatal: Unable to open file 'com.obj'

E:\lr1>BAD.COM
PC type: AT
MS-DOS:  5.00
OEM:  FF
USER NUMBER:  0000000h
```

Рисунок 1 – Полученный .COM модуль

Во время линковки «плохого» .EXE модуля было выведено предупреждение об отсутствии сегмента стека, представленное на рис. 1.

```
E:\lr1>masm BAD.asm
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1988. All rights reserved.

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

49086 + 436653 Bytes symbol space free

0 Warning Errors
0 Severe Errors

E:\lr1>link BAD.OBJ

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

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

Рисунок 2 – Предупреждение во время линковки

Результат выполнения «плохого» .EXE модуля представлен на рис. 3.

```
E:\1r1>BAD.EXE
```

```
    ZAX=
```

```
    5.00
```

```
    FF
```

```
    0000000h  
    ZAX=
```

```
    ZAX=
```

```
    ZAX=
```

```
    ZAX=
```

Рисунок 3 – Вывод «плохого» .EXE модуля

Результат выполнения «хорошего» .COM модуля представлен на рис. 4.

```
E:\1r1>BAD.COM  
PC type: AT  
MS-DOS: 5.00  
OEM: FF  
USER NUMBER: 0000000h
```

Рисунок 4 – Вывод «хорошего» .COM модуля

Был написан текст исходного .EXE модуля, который выполняет те же функции, что и .COM модуль. В результате постройки и отладки был получен «хороший» .EXE модуль. Результат его выполнения представлен на рис. 5.

```
E:\1r1>GOOD.EXE  
PC type: AT  
MS-DOS: 5.00  
OEM: FF  
USER NUMBER: 0000000h
```

Рисунок 5 – Вывод «хорошего» .EXE модуля

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

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

.COM - программы содержат только один сегмент. Модель памяти tiny - код, данные и стек объединены в один физический сегмент, максимальный размер которого не мог превышать 64 Кбайта без 256 байтов (последние требуются для создания префикса программного сегмента (PSP)).

#### **2) EXE программа?**

EXE-программа может содержать несколько сегментов. При использовании модели памяти small, в программе должен содержаться один сегмент данных и один сегмент кода в разных физических сегментах и каждый

из них не может превосходить 64 Кбайта. При других моделях памяти (например large) есть возможность использования нескольких сегментов данных и (или) нескольких сегментов кода.

Помимо этого, в программе должен быть описан сегмент стека (до 64 Кбайт, содержит адреса возврата как для программы (для возврата в операционную систему), так и для вызовов подпрограмм (для возврата в главную программу), а также используется для передачи параметров в процедуры). Использует операционная система при обработке прерываний. Регистр сегмента стека (SS) адресует данный сегмент. Адрес текущей вершины стека задается регистрами SS:ESP.

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

При запуске COM-программы первые 100h байт необходимо зарезервировать для префикса программного сегмента (PSP). Для этого используется директива ORG, которая устанавливает относительный адрес для начала выполнения программы: ORG 100h.

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

В .COM файле отсутствует таблица настройки с информацией о типе адресов и их местоположении в коде. Поэтому нельзя использовать команды, связанные с адресом сегмента, так как адрес сегмента неизвестен вплоть до загрузки этого сегмента в память. Загрузчику необходима информация о местоположении в файле загрузочного модуля полей адресов.

**Отличия форматов файлов COM и EXE модулей.**

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

Вид файла COM в шестнадцатеричном формате представлен на рис. 6.

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Текст декодирован
00000000	E9	16	01	50	43	0D	0A	24	50	43	2F	58	54	0D	0A	24	PC..\$PC/XT..\$
00000010	41	54	0D	0A	24	50	53	32	20	6D	6F	64	65	6C	20	33	AT..\$PS2 model 3
00000020	30	0D	0A	24	50	53	32	20	6D	6F	64	65	6C	20	35	30	0..\$PS2 model 50
00000030	2F	36	30	0D	0A	24	50	53	32	20	6D	6F	64	65	6C	20	/60..\$PS2 model
00000040	38	30	0D	0A	24	50	53	6A	72	0D	0A	24	50	43	20	63	80..\$PSjr..\$PC c
00000050	6F	6E	76	65	72	74	69	62	6C	65	0D	0A	24	49	42	4D	convertible..\$IBM
00000060	20	50	43	20	74	79	70	65	20	69	73	3A	20	24	4D	53	PC type is: \$MS
00000070	44	4F	53	20	76	65	72	73	69	6F	6E	20	69	73	3A	20	DOS version is:
00000080	20	2E	20	0D	0A	24	4F	45	4D	20	6E	75	6D	62	65	72	. ..\$OEM number
00000090	20	69	73	3A	20	20	20	20	0D	0A	24	53	65	72	69		is: ..\$Seri
000000A0	61	6C	20	6E	75	6D	62	65	72	20	69	73	3A	20	20	20	al number is:
000000B0	20	20	20	0D	0A	24	50	B4	09	CD	21	58	C3	24	0F	3C	..\$Pr.H!XG\$.<
000000C0	09	76	02	04	07	04	30	C3	51	8A	C4	E8	EF	FF	86	C4	.v....0GQMDипятД
000000D0	B1	04	D2	E8	E8	E6	FF	59	C3	53	8A	FC	E8	E9	FF	88	т.ТижкаYGSльийяе
000000E0	25	4F	88	05	4F	8A	C7	32	E4	E8	DC	FF	88	25	4F	88	%0€.0L32диbя€%0€
000000F0	05	5B	C3	51	52	50	32	E4	33	D2	B9	0A	00	F7	F1	80	. [GQRP2д3TМ...чсЪ
00000100	CA	30	88	14	4E	33	D2	3D	0A	00	73	F1	3D	00	00	76	K0€.N3T=...sc=..v
00000110	04	0C	30	88	04	58	5A	59	C3	52	50	BA	5D	01	E8	95	..0€.XYZYGRPe].и*
00000120	FF	B8	00	F0	8E	C0	26	AO	FE	FF	3C	FF	74	20	3C	FE	яё.pTAc юя<ят <ю
00000130	74	22	3C	FB	74	1E	3C	FC	74	20	3C	FA	74	22	3C	FC	t"<ят.<ьт <ьт"<ь
00000140	74	24	3C	F8	74	26	3C	FD	74	28	3C	F9	74	2A	BA	03	t\$<ят&<эт (<ят*е.
00000150	01	EB	2B	90	BA	08	01	EB	25	90	BA	10	01	EB	1F	90	.л+ђе...л*ђе...л.ђ
00000160	BA	15	01	EB	19	90	BA	24	01	EB	13	90	BA	36	01	EB	е...л.ђе\$.л.ђе6.л
00000170	0D	90	BA	45	01	EB	07	90	BA	4C	01	EB	01	90	E8	35	.ђеE.л.ђеL.л.ђи5
00000180	FF	B4	30	CD	21	8D	36	6E	01	83	C6	12	E8	64	FF	83	ягOH!K6n.ђЖ.идяђ
00000190	C6	03	8A	C4	E8	5C	FF	BA	6E	01	E8	19	FF	8A	C7	8D	Ж.ЉДи\яеп.и.яЉ3K
000001A0	36	86	01	83	C6	0F	E8	4A	FF	BA	86	01	E8	07	FF	8A	6+.ђЖ.иJяет+.и.яЉ
000001B0	C3	8D	36	9C	01	83	C6	12	E8	0D	FF	89	04	83	C6	06	ГK6њ.ђЖ.и.я%.ђЖ.
000001C0	8B	FE	8B	C1	E8	12	FF	BA	9C	01	E8	E9	FE	58	5A	32	<ю<Би.яењ.ийюXZ2
000001D0	CD	B4	4C	CD	21	C3											AгLN!Г

Рисунок 6 – Вид СОМ файла в шестнадцатеричном виде

СОМ-файл состоит из одного сегмента, а размер файла не превышает 64 КБ. Код располагается с нулевого адреса, что видно на рис. 5.

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

Что располагается с адреса 0?

Вид «плохого» EXE файла в шестнадцатеричном формате представлен на рис. 7.

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	Текст декодирован
00000000	4D	5A	D6	00	03	00	00	00	20	00	00	00	FF	FF	00	00	00	00	23	69	00	01	00	00	1E	00	00	00	01	00	00	00	МЗЦ.....ая...#1.....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	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	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	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	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	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	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	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	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	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	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	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	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	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	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	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000200	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000220	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000260	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000280	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000002A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000002C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000002E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000300	E9	16	01	50	43	0D	0A	24	50	43	2F	58	54	0D	0A	24	41	54	0D	0A	24	50	53	32	20	6D	6F	64	65	6C	20	33	й..PC..\$PC/XT..\$AT..\$PS2 model 3
00000320	30	0D	0A	24	50	53	32	20	6D	6F	64	65	6C	20	35	30	2F	36	30	0D	0A	24	50	53	32	20	6D	6F	64	65	6C	20	0..\$PS2 model 50/60..\$PS2 model
00000340	38	30	0D	0A	24	50	53	6A	72	0D	0A	24	50	43	20	63	6F	6E	76	65	72	74	69	62	6C	65	0D	0A	24	49	42	4D	80..\$PSjr..\$PC convertible..\$IBM
00000360	20	50	43	20	74	79	70	65	20	69	73	3A	20	24	4D	53	44	4F	53	20	76	65	72	73	69	6F	6E	20	69	73	3A	20	PC type is: \$MSDOS version is:
00000380	20	2E	20	0D	0A	24	4F	45	4D	20	6E	75	6D	62	65	72	20	69	73	3A	20	20	20	20	20	0D	0A	24	53	65	72	69	..\$OEM number is: ..\$Serial number is:
000003A0	61	6C	20	6E	75	6D	62	65	72	20	69	73	3A	20	20	20	20	20	0D	0A	24	50	B4	09	CD	21	58	C3	24	0F	3C	<.v...ОГQМДиятД±.ТияяУТЗЪийяе	
000003C0	09	76	02	04	07	04	30	C3	51	8A	C4	E8	EF	FF	86	C4	B1	04	D2	E8	E8	E6	FF	59	C3	53	8A	FC	E8	E9	FF	88	%ОЕ.ОБЗ2дияяе%ОЕ.[QORP2лЗТМ..чсБ
000003E0	25	4F	88	05	4F	8A	C7	32	E4	E8	DC	FF	88	25	4F	88	05	5B	C3	51	52	50	32	E4	33	D2	B9	0A	00	F7	F1	80	КОЕ.НЗТ=...sc=.v..ОЕ.XZYTRPe].и*
00000400	CA	30	88	14	4E	33	D2	3D	0A	00	73	F1	3D	00	00	76	04	0C	30	88	04	58	5A	59	C3	52	50	BA	5D	01	E8	95	яе.рРАе юя<ят <ют<ют.<ят <ят"<ь
00000420	FF	B8	00	F0	8E	C0	26	A0	FE	FF	3C	FF	74	20	3C	FE	74	22	3C	FB	74	1E	3C	FC	74	20	3C	FA	74	22	3C	FC	т\$<шт<эт(<шт*е..л+е..л%е..л.е
00000440	74	24	3C	F8	74	26	3C	FD	74	28	3C	F9	74	2A	BA	03	01	EB	2B	90	BA	08	01	EB	25	90	BA	10	01	EB	1F	90	е..л.е\$е.л.е6.л.еЕ.л.еL.л.и5
00000460	BA	15	01	EB	19	90	BA	24	01	EB	13	90	BA	36	01	EB	0D	90	BA	45	01	EB	07	90	BA	4C	01	EB	01	90	E8	35	яОН!К6н.рЖ.идярЖ.лДи\яен.и.яБЗК
00000480	FF	B4	30	CD	21	8D	36	6E	01	B3	C6	12	E8	64	FF	83	C6	03	8A	C4	E8	5C	FF	BA	6E	01	E8	19	FF	8A	C7	8D	6т.рЖ.иJяет+.и.яЛК6ъ.рЖ.и.яъ.рЖ.
000004A0	36	86	01	83	C6	0F	E8	4A	FF	BA	86	01	E8	07	FF	8A	C3	8D	36	9C	01	83	C6	12	E8	0D	FF	89	04	83	C6	06	<ю<Би.яъ.июXZ2ArLH!Г
000004C0	8B	FE	8B	C1	E8	12	FF	BA	9C	01	E8	E9	FE	58	5A	32	CD	B4	4C	CD	21	C3											

Рисунок 7 – Вид «плохого» EXE файла

В «плохом» EXE-файле код располагается с адреса 300h. С нулевого адреса располагается управляющая информация для загрузчика, образующая заголовок.

Также 100h резервируются командой ORG 100h (что вызовет отличие в структуре «плохого» и «хорошего» EXE).

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

Вид файла «хорошего» EXE в шестнадцатеричном виде представлен на рис.

8.



Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	Текст декодирован
00000000	4D	5A	E5	01	03	00	01	00	20	00	00	00	FF	FF	00	00	00	02	5C	22	63	00	2C	00	МZe.....\с,,
00000018	1E	00	00	00	01	00	66	00	2C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....f,,
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000048	00	00	00	00	00	00	00	00	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	00	00	00	00	00	00	00	00	.....
00000078	00	00	00	00	00	00	00	00	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	00	00	00	00	00	00	00	00	.....
000000A8	00	00	00	00	00	00	00	00	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	00	00	00	00	00	00	00	00	.....
000000D8	00	00	00	00	00	00	00	00	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	00	00	00	00	00	00	00	00	.....
00000108	00	00	00	00	00	00	00	00	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	00	00	00	00	00	00	00	00	.....
00000138	00	00	00	00	00	00	00	00	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	00	00	00	00	00	00	00	00	.....
00000168	00	00	00	00	00	00	00	00	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	00	00	00	00	00	00	00	00	.....
00000198	00	00	00	00	00	00	00	00	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	00	00	00	00	00	00	00	00	.....
000001C8	00	00	00	00	00	00	00	00	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	00	00	00	00	00	00	00	00	.....
000001F8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000210	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000228	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000258	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000270	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000288	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000002A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000002B8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000002D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000002E8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000300	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000318	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000330	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000348	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000360	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000378	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000390	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000003A8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000003C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000003D8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000003F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	50	43	0D	0A	24	50	43	2F	.....PC..\$PC/
00000408	58	54	0D	0A	24	41	54	0D	0A	24	50	53	32	20	6D	6F	64	65	6C	20	33	30	0D	0A	XT..\$AT..\$PS2 model 30..
00000420	24	50	53	32	20	6D	6F	64	65	6C	20	35	30	2F	36	30	0D	0A	24	50	53	32	20	6D	\$PS2 model 50/60..\$PS2 m
00000438	6F	64	65	6C	20	38	30	0D	0A	24	50	53	6A	72	0D	0A	24	50	43	20	63	6F	6E	76	odel 80..\$PSjr..\$PC conv
00000450	65	72	74	69	62	6C	65	0D	0A	24	49	42	4D	20	50	43	20	74	79	70	65	20	69	73	ertible..\$IBM PC type is
00000468	3A	20	24	4D	53	44	4F	53	20	76	65	72	73	69	6F	6E	20	69	73	3A	20	20	2E	20	: \$MSDOS version is:
00000480	0D	0A	24	4F	45	4D	20	6E	75	6D	62	65	72	20	69	73	3A	20	20	20	20	0D	0A	..	..\$OEM number is:
00000498	24	53	65	72	69	61	6C	20	6E	75	6D	62	65	72	20	69	73	3A	20	20	20	20	20	20	\$Serial number is:
000004B0	0D	0A	24	00	00	00	00	00	00	00	00	00	00	00	00	50	B4	09	CD	21	58	C3	24	..	..\$Pr.N!XГ\$
000004C8	0F	3C	09	76	02	04	07	04	30	C3	51	8A	C4	E8	EF	FF	86	C4	B1	04	D2	E8	E8	E6	.<.v...OГQМДипятД±.Тиж
000004E0	FF	59	C3	53	8A	FC	E8	E9	FF	88	25	4F	88	05	4F	8A	C7	32	E4	E8	DC	FF	88	25	яУГСЪийяёёОё.ОМ32дйбёё
000004F8	4F	88	05	5B	C3	51	52	50	32	E4	33	D2	B9	0A	00	F7	F1	80	CA	30	88	14	4E	33	Оё. [ГQRP2д3ТМ.чсЪКОё.N3
00000510	D2	3D	0A	00	73	F1	3D	00	00	76	04	0C	30	88	04	58	5A	59	C3	52	50	B8	20	00	T=..sc=..v..Oё.XZYГRPё .
00000528	8E	D8	BA	5A	00	E8	90	FF	B8	00	FO	8E	CD	26	AO	FE	FF	3C	FF	74	20	3C	FE	74	ТШеZ.иђяё.pђАё юя<ят <от
00000540	22	3C	FB	74	1E	3C	FC	74	20	3C	FA	74	22	3C	FC	74	24	3C	F8	74	26	3C	FD	74	"<шт.<шт <шт"<шт\$<шт\$<шт
00000558	28	3C	F9	74	2A	BA	00	00	EB	2B	90	BA	05	00	EB	25	90	BA	0D	00	EB	1F	90	BA	(<шт*ё...л+ёе...л+ёе...л.ёе
00000570	12	00	EB	19	90	BA	21	00	EB	13	90	BA	33	00	EB	0D	90	BA	42	00	EB	07	90	BA	..л.ёе!..л.ёе3..л.ёеВ.л.ёе
00000588	49	00	EB	01	90	E8	30	FF	B4	30	CD	21	8D	36	6B	00	83	C6	12	E8	5F	FF	83	C6	I..л.ёиОягОН!Кёк.ђЖ.и яђЖ
000005A0	03	8A	C4	E8	57	FF	BA	6B	00	E8	14	FF	8A	C7	8D	36	83	00	83	C6	0F	E8	45	FF	МдиWяек.и.ялъКёђ.ђЖ.иЕя
000005B8	BA	83	00	E8	02	FF	8A	C3	8D	36	99	00	83	C6	12	E8	08	FF	89	04	83	C6	06	8B	еђ.и.яМГКё™.ђЖ.и.яъ.ђЖ.<
000005D0	FE	8B	C1	E8	0D	FF	BA	99																	

адреса 400h. Если из исходного текста .EXE-программы убрать сегмент стека, то код будет располагаться с адреса 200h. Отличие от «плохого» EXE в том, что в «хорошем» не резервируется дополнительно 100h, которые в COM файле требовались для PSP, поэтому адреса начала кода отличаются на  $100h + S$ , где  $S$  – размер стека.

### Загрузка COM модуля в основную память.

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

Запуск файла .COM в отладчике AFD.EXE представлен на рис. 9.

AX 0000	SI 0000	CS 159B	IP 0100	Stack +0 0000	FLAGS 0200							
BX 0000	DI 0000	DS 158B		+2 0000								
CX 02D6	BP 0000	ES 158B	HS 158B	+4 0000	OF	DF	IF	SF	ZF	AF	PF	CF
DX 0000	SP 0000	SS 159B	FS 158B	+6 0000	0	0	1	0	0	0	0	0
CMD >					0	1	2	3	4	5	6	7
				DS:0000	CD	20	FF	9F	00	9A	EE	FE
				DS:0008	1D	F0	ED	04	50	05	4B	01
0100 E91601 JMP 0219				DS:0010	15	04	56	01	15	04	50	05
0103 50 PUSH AX				DS:0018	01	01	01	00	02	FF	FF	FF
0104 43 INC BX				DS:0020	FF	FF	FF	FF	FF	FF	FF	FF
0105 0D0A24 OR AX, 240A				DS:0028	FF	FF	FF	FF	54	15	C4	FF
0108 50 PUSH AX				DS:0030	50	05	14	00	18	00	8B	15
0109 43 INC BX				DS:0038	FF	FF	FF	FF	00	00	00	00
010A 2F DAS				DS:0040	05	00	00	00	00	00	00	00
010B 58 POP AX				DS:0048	00	00	00	00	00	00	00	00
2												
DS:0000				CD	20	FF	9F	00	9A	EE	FE	
DS:0010				15	04	56	01	15	04	50	05	
DS:0020				FF	FF	FF	FF	FF	FF	FF	FF	
DS:0030				50	05	14	00	18	00	8B	15	
DS:0040				05	00	00	00	00	00	00	00	

Рисунок 9 – Отладка файла .COM

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?

С нулевого адреса (0h) располагается сегмент PSP.

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

Сегментные регистры имеют значения 48DDh и указывают на PSP.

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

В COM модуле нельзя объявить стек, он создается автоматически. На рис. 8 видно, что SP указывает на FFFEh. Стек занимает оставшуюся память (из 64 Кб), а его адреса изменяются от больших к меньшим, то есть от FFFEh к 0000h.

### Загрузка «хорошего» EXE. модуля в основную память.

Запуск «хорошего» EXE модуля в отладчике AFD.EXE представлен на рис.

10.

AX 0000	SI 0000	CS 15CD	IP 0063	Stack +0 4350	FLAGS 0200												
BX 0000	DI 0000	DS 1591		+2 0A0D	OF	DF	IF	SF	ZF	AF	PF	CF					
CX 03E5	BP 0000	ES 1591	HS 1591	+4 5024	0	0	1	0	0	0	0	0					
DX 0000	SP 0200	SS 15A1	FS 1591	+6 2F43	0	0	1	0	0	0	0	0					
CMD >				0	1	2	3	4	5	6	7						
				DS:0000	CD	20	FF	9F	00	9A	EE	FE					
				DS:0008	1D	F0	ED	04	53	05	4B	01					
0063 52	PUSH		DX	DS:0010	15	04	56	01	15	04	53	05					
0064 50	PUSH		AX	DS:0018	01	01	01	00	02	FF	FF	FF					
0065 B8C115	MOV		AX,15C1	DS:0020	FF	FF	FF	FF	FF	FF	FF	FF					
0068 8ED8	MOV		DS,AX	DS:0028	FF	FF	FF	FF	57	15	C4	FF					
006A BA5A00	MOV		DX,005A	DS:0030	53	05	14	00	18	00	91	15					
006D E890FF	CALL		0000	DS:0038	FF	FF	FF	FF	00	00	00	00					
0070 B800F0	MOV		AX,F000	DS:0040	05	00	00	00	00	00	00	00					
0073 8EC0	MOV		ES,AX	DS:0048	00	00	00	00	00	00	00	00					
2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
DS:0000	CD	20	FF	9F	00	9A	EE	FE	1D	F0	ED	04	53	05	4B	01	u
DS:0010	15	04	56	01	15	04	53	05	01	01	01	00	02	FF	FF	FF	u
DS:0020	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	57	15	C4	FF	FF	u
DS:0030	53	05	14	00	18	00	91	15	FF	FF	FF	FF	00	00	00	00	u
DS:0040	05	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	u

Рисунок 10 – Отладка «хорошего» EXE модуля

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

Определяется сегментный адрес свободного участка ОП, в который можно загрузить программу. Создается блок памяти для PSP и программы. 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.

В PSP заносятся соответствующие значения. В рабочую область загрузчика считывается форматированная часть заголовка файла. Определяется смещение



начала загрузочного модуля в EXE файле. Вычисляется сегментный адрес (START\_SEG) для загрузки. В память считывается загрузочный модуль. Таблица настройки порциями считывается в рабочую память. Для каждого элемента таблицы настройки к полю сегмента прибавляется сегментный адрес начального сегмента (в результате элемент таблицы указывает на нужное слово в памяти). Управление передается загруженной задаче по адресу из заголовка.

## 2) На что указывают регистры DS и ES?

Изначально регистры DS и ES указывают на начало сегмента PSP. Именно поэтому в начале программы для корректной работы с данными необходимо загрузить в DS адрес сегмента данных.

## 3) Как определяется стек?

Стек может быть объявлен при помощи директивы ASSUME, которая устанавливает сегментный регистр SS на начало сегмента стека, а также задает значение SP, указанное в заголовке. Также стек может быть объявлен с помощью директивы STACK. Если стек не объявлять, то он будет создан автоматически таким же образом, как в COM-модуле. Вид программы EXE модуля без объявленного стека после команды push в отладчике представлен на рис. 11.

AX 0000	SI 0000	CS 15AD	IP 0063	Stack +0 4350	FLAGS 0200												
BX 0000	DI 0000	DS 1591		+2 0A0D													
CX 01E5	BP 0000	ES 1591	HS 1591	+4 5024	OF	DF	IF	SF	ZF	AF	PF	CF					
DX 0000	SP 0000	SS 15A1	FS 1591	+6 2F43	0	0	1	0	0	0	0	0					
[CMD >]						0	1	2	3	4	5	6	7				
					DS:0000	CD	20	FF	9F	00	9A	EE	FE				
					DS:0008	1D	F0	ED	04	53	05	4B	01				
0063 52	PUSH DX			DS:0010	15	04	56	01	15	04	53	05					
0064 50	PUSH AX			DS:0018	01	01	01	00	02	FF	FF	FF					
0065 B8A115	MOV AX,15A1			DS:0020	FF	FF	FF	FF	FF	FF	FF	FF					
0068 8ED8	MOV DS,AX			DS:0028	FF	FF	FF	FF	57	15	C4	FF					
006A BA5A00	MOV DX,005A			DS:0030	53	05	14	00	18	00	91	15					
006D E890FF	CALL 0000			DS:0038	FF	FF	FF	FF	00	00	00	00					
0070 B800F0	MOV AX,F000			DS:0040	05	00	00	00	00	00	00	00					
0073 8EC0	MOV ES,AX			DS:0048	00	00	00	00	00	00	00	00					
2						0	1	2	3	4	5	6	7				
DS:0000	CD	20	FF	9F	00	9A	EE	FE	1D	F0	ED	04	53	05	4B	01	
DS:0010	15	04	56	01	15	04	53	05	01	01	01	00	02	FF	FF	FF	.....S.K.
DS:0020	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	57	15	C4	FF	.....W.
DS:0030	53	05	14	00	18	00	91	15	FF	FF	FF	FF	00	00	00	00	.....S
DS:0040	05	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
1 Step					2StepProc	3Retrieve	4 Help	5Set BRK	6	7 up	8 dn	9 le	0 ri				

Рисунок 11 – Отладка EXE модуля без объявленного стека

#### 4) Как определяется точка входа?

Смещение точки входа в программу загружается в указатель команд IP и определяется операндом директивы END <метка для входа>, 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

### ИСХОДНЫЙ КОД ПРОГРАММЫ. BAD.ASM

```

TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100H ; резервирование места для PSP
START: JMP BEGIN

; DATA SEGMENT
STRING                db 'AX= ',0DH,0AH,'$'

PC_TYPEM              db 'IBM PC type is : ', '$'
PC_TYPE               db 'PC', 13, 10, '$'
PC_XT_TYPE            db 'PC/XT', 13, 10, '$'
AT_TYPE               db 'AT', 13, 10, '$'
PS230                 db 'PS2 model 30',13, 10, '$'
PS250                 db 'PS2 model 50 or 60', 13, 10, '$'
PS280                 db 'PS2 model 80', 13, 10, '$'
PCjr_TYPE             db 'PCjr_TYPE', 13, 10, '$'
PC_CONVERTIBLE        db 'PC Convertible', 13, 10, '$'

MS_DOS_VERSION        db 'MS-DOS:          ', 13, 10, '$'
OEM                   db 'OEM:              ', 13, 10, '$'
USER_NUM              db 'USER NUMBER:      ', 13, 10, '$'

; Procedures
;-----
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 10 с/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
;-----

```

```

TYPE_PC PROC near
; Вывод текущей системы
    push AX
    push BX
    push CX
    push ES
    push DI

    mov DX, offset PC_TYPEM
    mov AH, 09h
    int 21h
    xor AH,AH

    mov CX, 0F000h
    mov ES,CX
    mov AL, ES:[0FFFEh]

    cmp AX, 0FFh
    jz PCM
    cmp AX, 0FEh
    jz PC_XT_TYPEM
    cmp AX, 0FBh
    jz PC_XT_TYPEM
    cmp AX, 0FCh
    jz ATM
    cmp AX, 0FAh
    jz PS230M
    cmp AX, 0F6h
    jz PS250M
    cmp AX, 0F8h
    jz PS280M
    cmp AX, 0FDh
    jz PCjr_TYPEM
    cmp AX, 0F9h
    jz PC_CONVERTIBLEM

PCM:
    mov DX,offset PC_TYPE
    mov AH,09h
    int 21h

    jmp ENDPC

PC_XT_TYPEM:
    mov DX,offset PC_XT_TYPE
    mov AH,09h
    int 21h

```



```

        jmp ENDPC

ATM:
    mov DX,offset AT_TYPE
    mov AH,09h
    int 21h

    jmp ENDPC

PS230M:
    mov DX,offset PS230
    mov AH,09h
    int 21h

    jmp ENDPC

PS250M:
    mov DX,offset PS250
    mov AH,09h
    int 21h

    jmp ENDPC

PS280M:
    mov DX,offset PS280
    mov AH,09h
    int 21h

    jmp ENDPC

PCjr_TYPEM:
    mov DX,offset PCjr_TYPE
    mov AH,09h
    int 21h

    jmp ENDPC
P
C_CONVERTIBLEM:
    mov DX,offset PC_CONVERTIBLE
    mov AH,09h
    int 21h

    jmp ENDPC

ENDPC:
    pop AX

```

```

        pop BX
        pop CX
        pop ES
        ret
TYPE_PC ENDP
;-----
MS_DOS_VER PROC near
    ; Вывод информации о версии DOS
    push AX
    push BX
    push CX
    push DX

    xor AX,AX
    mov ah, 30h
    int 21h

    push BX
    push CX

    ; Определяем и выводим версию
    mov BX, offset MS_DOS_VERSION
    mov DH, AH
    xor AH, AH
    call BYTE_TO_HEX
    cmp AL, '0'
    jz al_null
    mov [BX+9],AX
    add BX,2
    jmp continue_ms

al_null:

    mov [BX+9],AH
    inc BX

continue_ms:
    xor AX,AX
    mov AL,DH
    xor DX,DX
    mov CH, '.'
    mov [BX+9],CH
    inc BX
    call BYTE_TO_HEX
    mov [BX+9],AX

```

```

    mov DX, offset MS_DOS_VERSION
    mov AH,09h
    int 21h

;Определяем OEM

    pop CX
    pop BX

    xor DX,DX
    mov DX,BX
    mov BX,offset OEM
    xor AX,AX
    mov AL,DH
    call BYTE_TO_HEX
    cmp al,'0'
    jz null_oem
    mov [BX+6],AX
    jmp continue_oem
null_oem:
    mov AH,'0'
    mov [BX+6],AH

continue_oem:
    mov BX,DX
    mov DX, offset OEM
    mov AH,09h
    int 21h

;Определяем номер пользователя
    mov AL,BL
    mov BX, offset USER_NUM
    call BYTE_TO_HEX
    mov [BX+14],AX
    add BX,2

    xor AX,AX
    mov AL,CH
    call BYTE_TO_HEX
    mov [BX+14],AX
    add BX,2

    xor AX,AX
    mov AL,CL
    call BYTE_TO_HEX
    mov [BX+14],AX

```

```

    add BX,2

    xor AX,AX
    mov AH,'h'
    mov [BX+14],AH

    mov DX,offset USER_NUM
    mov AH,09h
    int 21h

    pop DI
    pop AX
    pop BX
    pop CX
    pop DX
    ret
MS_DOS_VER ENDP
;-----

; CODE SEGMENT
BEGIN:

    ; PS TYPE
    call TYPE_PC

    ; MS-DOS VERSION
    call MS_DOS_VER

    xor AL,AL
    mov AH,4Ch
    int 21H
TESTPC ENDS
    END START

```

## ПРИЛОЖЕНИЕ Б

### ИСХОДНЫЙ КОД ПРОГРАММЫ. GOOD.ASM

```

EOFLine EQU '$'

AStack   SEGMENT STACK
          DW 12 DUP(?)
AStack   ENDS

DATA SEGMENT

PC_TYPEM      db 'PC type: ', '$'
PC_TYPE       db 'PC', 13, 10, '$'
PCXT          db 'PC/XT', 13, 10, '$'
AT_TYPE       db 'AT', 13, 10, '$'
PS230         db 'PS2 model 30', 13, 10, '$'
PS250         db 'PS2 model 50 or 60', 13, 10, '$'
PS280         db 'PS2 model 80', 13, 10, '$'
PCjr_TYPE     db 'PCjr_TYPE', 13, 10, '$'
PC_CONVERT    db 'PC Convertible', 13, 10, '$'

MS_DOS_VERSION db 'MS-DOS: ', 13, 10, '$'
OEM            db 'OEM: ', 13, 10, '$'
USER_NUM       db 'USER NUMBER: ', 13, 10, '$'
DATA          ENDS

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

; Procedures
;-----
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 10 с/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
```

```
;-----
```

```
TYPE_PC PROC near
; Выводит сообщение, какая это система
```

```
push AX
push BX
push CX
push ES
```

```
mov DX, offset PC_TYPEM
mov AH, 09h
int 21h
xor AH,AH
```

```
mov CX, 0F000h
mov ES,CX
mov AL, ES:[0FFFEh]
```

```
cmp AX, 0FFh
jz PCM
cmp AX, 0FEh
jz PCXTM
cmp AX, 0FBh
jz PCXTM
cmp AX, 0FCh
jz ATM
cmp AX, 0FAh
jz PS230M
cmp AX, 0F6h
jz PS250M
cmp AX, 0F8h
jz PS280M
cmp AX, 0FDh
jz PCjr_TYPEM
cmp AX, 0F9h
jz PC_CONVERTM
```

```
PCM:
```

```
mov DX,offset PC_TYPE
mov AH,09h
int 21h

jmp ENDP
```

PCXTM:

```
mov DX,offset PCXT
mov AH,09h
int 21h
```

jmp ENDPC

ATM:

```
mov DX,offset AT_TYPE
mov AH,09h
int 21h
```

jmp ENDPC

PS230M:

```
mov DX,offset PS230
mov AH,09h
int 21h
```

jmp ENDPC

PS250M:

```
mov DX,offset PS250
mov AH,09h
int 21h
```

jmp ENDPC

PS280M:

```
mov DX,offset PS280
mov AH,09h
int 21h
```

jmp ENDPC

PCjr\_TYPEM:

```
mov DX,offset PCjr_TYPE
mov AH,09h
int 21h
```

jmp ENDPC

PC\_CONVERTM:

```
mov DX,offset PC_CONVERT
mov AH,09h
int 21h
```



```

        jmp ENDPc

ENDPC:
        pop AX
        pop BX
        pop CX
        pop ES
        ret
TYPE_PC ENDP
;-----
MS_DOS_VER PROC near
; Выводит информацию о версии dos
        push AX
        push BX
        push CX
        push DX

        xor AX,AX
        mov ah, 30h
        int 21h

        push BX
        push CX
; Определяем и выводим версию

        mov BX, offset MS_DOS_VERSION
        mov DH, AH
        xor AH, AH
        call BYTE_TO_HEX
        cmp AL, '0'
        jz al_null
        mov [BX+9],AX
        add BX,2
        jmp continue_ms

al_null:

        mov [BX+9],AH
        inc BX

continue_ms:
        xor AX,AX
        mov AL,DH
        xor DX,DX
        mov CH, '.'
        mov [BX+9],CH
        inc BX

```

```

    call BYTE_TO_HEX
    mov [BX+9],AX

    mov DX, offset MS_DOS_VERSION
    mov AH,09h
    int 21h

; Определяем OEM

    pop CX
    pop BX

    xor DX,DX
    mov DX,BX
    mov BX,offset OEM
    xor AX,AX
    mov AL,DH
    call BYTE_TO_HEX
    cmp al,'0'
    jz null_oem
    mov [BX+6],AX
    jmp continue_oem
null_oem:
    mov AH,'0'
    mov [BX+6],AH

continue_oem:
    mov BX,DX
    mov DX, offset OEM
    mov AH,09h
    int 21h

; Определяем номер пользователя
    mov AL,BL
    mov BX, offset USER_NUM
    call BYTE_TO_HEX
    mov [BX+14],AX
    add BX,2

    xor AX,AX
    mov AL,CH
    call BYTE_TO_HEX
    mov [BX+14],AX
    add BX,2

    xor AX,AX

```

```

    mov AL,CL
    call BYTE_TO_HEX
    mov [BX+14],AX
    add BX,2

    xor AX,AX
    mov AH,'h'
    mov [BX+14],AH

    mov DX,offset USER_NUM
    mov AH,09h
    int 21h

    pop AX
    pop BX
    pop CX
    pop DX
    ret
MS_DOS_VER ENDP
;-----

WriteMsg PROC NEAR
    mov AH,9
    int 21h
    ret
WriteMsg ENDP

Main PROC FAR
    push DS
    sub AX,AX
    push AX
    mov AX,DATA
    mov DS,AX

; Тип ПК
        call TYPE_PC

; Версия Ms-dos
        call MS_DOS_VER

; Выход в DOS
    xor AL,AL
    mov AH,4Ch
    int 21H
    ret

```

Main	ENDP
CODE	ENDS
	END Main