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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Операционные системы»

Тема: Исследование структур загрузочных модулей

Студентка гр. 8383	 Максимова А.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.

Выполнение работы.

Был написан текст исходного .СОМ модуля, код которого представлен в приложении А, который определяет тип РС и версию системы. Ассемблерная программа читает содержимое предпоследнего байта ROM BIOS. Полученный код сравнивается с известными значениями, выводится тип РС. В случае, когда код не совпадает ни с одним из известных значений, переводится в символьную строку, содержащую запись шестнадцатеричного числа и выводится на экран. После определяется версия системы: используя функцию 30H прерывания 21H, получаем выходные параметры. Формируются соответствующие текстовые строки, выводятся на экран. Строятся "плохой" и "хороший".ЕХЕ, код которого представлен в приложении Б. Результаты работы модулей представлены на рис. 1, 2 и 3 соответственно.

Процедуры:

Название:	Предназначение:					
TETD TO HEY	Процедура перевода тетрады в					
TETR_TO_HEX	шестнадцатеричную цифру (символ)					
DVTE TO HEY	Процедура перевода байта AL в два					
BYTE_TO_HEX	символа шестнадцатеричного числа					
	Процедура перевода слова в					
WRD_TO_HEX	шестнадцатеричную систему					
	счисления					
DVTE TO DEC	Процедура перевода байта в					
BYTE_TO_DEC	десятичную систему счисления					
PRINTF	Процедура печати					
TWDE DC	Процедура определения и вывода типа					
TYPE_PC	PC					
CORRECT	Процедура для вывода версии системы					

	в заданном формате хх.уу
	Процедура определения и вывода
VERSION_SYSTEM	версии системы, серийного номера
	ОЕМ и номера пользователя

```
C:\>lr1.com
Register value AX = AT
System version: 05.00
Serial number OEM: 00
Serial number user: 000000
```

Рисунок 1 - результат работы .com



Рисунок 2 - результат работы "плохого" .exe

```
C:\>lr12.exe
Register value AX = AT
System version: 05.00
Serial number OEM: 00
Serial number user: 000000
```

Рисунок 3 - результат работы "хорошего" .exe

Ответы на контрольные вопросы.

Отличия исходных текстов СОМ и ЕХЕ программ

1. Сколько сегментов должна содержать СОМ-программа?

СОМ программа должна содержать один сегмент - данные, код и стек объединены в одном сегменте, поэтому СОМ-файл ограничен размером одного сегмента и не превышает 64К.

- 2. Сколько сегментов должна содержать EXE-программа? EXE-программы могут иметь любое число сегментов, минимум 1.
- 3. Какие директивы должны обязательно быть в тексте COM-программы? Директива ORG 100h, устанавливающая значение программного счетчика в 100h (при загрузке COM файла в память DOS занимает первые 256 байт блоком данных PSP и располагает код программы только после этого блока).
- 4. Все ли форматы команд можно использовать в СОМ-программе? Невозможно использовать команды, содержащие seg <имя сегмента>, так как адрес сегмента до запуска программы неизвестен.

Был запущен FAR, в котором были открыты загрузочные модули файлов .COM и .EXE ("хорошего" и "плохого"). Соответствующие скриншоты представлены на рис. 4, 5 и 6.

00000000000:	EO	3 E	02	FA	43	OD.	ØA.	24	50	43	2F	E 0	E /	OD.	QΛ	24	é> @ PC♪ ≋ \$PC/ X T♪ ≈ \$
000000000000000000000000000000000000000									20	6D					20		ATDmsPS2 model 3
0000000010:			0A		50		32		6D	6F	64	65			38		0 1 ■ \$PS2 model 80
00000000020:									ØA.	24					6F		
00000000030:			72					0D		9A		43			69	73	J⊠\$PCjrJ⊠\$PC Con
						62			ØD		24	52		67			vertible⊅⊠\$Regis
0000000050:	74		72			61		75	65	20		58			20		ter value AX = \$
0000000060:						69	6E	65	64		0D	ØA.			65		indefined.♪≊\$Pen
0000000070:						61	74		20	62		74		ЗА		09	ultimate byte:00
0000000080:					79		74		6D	20		65			69		>≡\$System versio
0000000090:			20	20	20	2E	20	20	ØD	ØA.	24	53	65	72	69	61	n: . ♪⊠\$Seria
00000000A0:	6C		6E			62		72	20	4F	45		за	20	20		1 number OEM:
000000000В0:			ØD			53		72	69	61		20			6D		⊅≊ \$Serial numb
00000000C0:		72		75	73	65	72	ЗА	20	20	20	20	20	20	20		er user:
00000000D0:			ØD.			24		3C	09		02		07		30		N =\$\$¢<0∨ 0 ♦•♦0Ã
00000000E0:			E0			FF	86	C4	B1	04		E8	E8		FF		QŠàèïÿ†Ä±♦ÒèèæÿY
00000000F0:			8A		E8	E9	FF	88	25	4F	88	05			C7		ÃSŠüèéÿ^‰0^ + 0ŠÇè
0000000100:						88		5B	C3	51					D2		Þÿ^%0^♣[ÃQR2ä3Ò¹
0000000110:					80	CA		88	14	4E					00		≡ ÷ñ€Ê0^¶N3Ò=≡ s
0000000120:						0C	30	88	04	5A	59	C3	В4		CD		ñ< t◆♀0^◆ZYôoÍ!
0000000130:	C3	B8	00	FØ	8E	CØ	26	A0	FE	FF	EB	01	90	3C	FF	75	Ã, ðŽÀ& þÿë⊕⊡<ÿu
0000000140:	09	BA	03	01	E8	E5	FF	EB	71	90	3C	FE	75	09	BA	08	oº♥⊕èåÿëq⊡<þuoº•
0000000150:	01	E8	D8	FF	EB	64	90	3C	FB	75	09	BA	08	01	E8	CB	@èØÿëd⊡<ûuoº <mark>•</mark> @èË
0000000160:	FF	EB	57	90	3C	FC	75	09	BA	10	01	E8	BE	FF	EB	4A	ÿëW⊡<üuoº►⊕è¾ÿëJ
0000000170:	90	3C	FA	75	09	BA	15	01	E8	B1	FF	EB	3D	90	3C	F8	2<úuoº§@è±ÿë=2<ø
0000000180:	75	09	BA	24	01	E8	A4	FF	EB	30	90	3C	FD	75	09	BA	uoº\$@è¤ÿë0⊡<ýuoº
0000000190:	33	01	E8	97	FF	EB	23	90	3C	F9	75	09	BA	3A	01	E8	3@è–ÿë#⊡<ùuoº:@è
00000001A0:	8A	FF	EB	16	90	BA	60	01	E8	81	FF	BF	6D	01	83	C7	Šÿë − ⊡º`@è⊡ÿ¿m@ƒÇ
00000001B0:	15	E8	3D	FF	BA	6D	01	E8	72	FF	C3	53	вз	10	F6	F3	§è=ÿºm⊕èrÿÃS³►öó
00000001C0:	80	C4	30	04	30	5B	СЗ	B4	30	CD	21	51	8B	C8	E8	EA	€Ä0•0[à 0Í!Q<Èèê
00000001D0:	FF	BE	83	01	83	C6	11	88	24	4E	88	04	33	CØ	88	E5	ÿ%f⊕fƬ^\$N^◆3ÀŠå
00000001E0:	E8	D8	FF	83	С6	04	88	24	4E	88	04	BA	83	01	E8	3B	èØÿfÆ♦^\$N^♦ºf@è;
00000001F0:	FF	33	CØ	8A	C7	E8	СЗ	FF	BE	9B	01	83	C6	14	88	24	ÿ3ÀŠÇèÃÿ¾>@fƶ^\$
0000000200:	4E	88	04	ВА	9B	01	E8	23	FF	59	33	CØ	8A	С3	E8	AA	N^♦º>@è#ÿY3ÀŠÃèª

Рисунок 4 - загрузочный модуль .СОМ

```
000000250: 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
000000270: 00 00 00 00 00 00 00 00
                                                    00 00 00 00 00 00 00
0000000280: 00 00 00 00 00 00 00 00
                                                    00 00 00 00 00 00 00
0000000290: 00 00 00 00 00 00 00 00
                                                    00 00 00 00 00 00 00
00000002A0: 00 00 00 00 00 00 00 00
                                                    00 00 00 00 00 00 00
                                                   00000002B0: 00 00 00 00 00 00 00 00
00000002C0: 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 00 00 00 00 00 00
00000002E0: 00 00 00 00 00 00 00 00
                                                    00 00 00 00 00 00 00
00000002F0: 00 00 00 00 00 00 00 00
0000000300: E9 3E 02 50 43 0D 0A 24
                                                                                      é>@PC♪æ$PC/XT♪æ$
0000000310: 41 54 0D 0A 24 50 53 32
0000000320: 30 0D 0A 24 50 53 32 20
                                                    20 6D 6F
                                                                    65 6C 20 33
                                                                                       AT♪≊$PS2 model
                                                    6D 6F 64 65
                                                                    6C 20 38 30
                                                                                      0⊅≡$PS2 model 80
0000000330: 0D 0A 24 50 43 6A 72 0D
0000000340: 76 65 72 74 69 62 6C 65
0000000350: 74 65 72 20 76 61 6C 75
0000000360: 69 6E 64 65 66 69 6E 65
                                                   0A 24 50 43 06 20 43 6F 6E
0D 0A 24 52 65 67 69 73
65 20 41 58 20 3D 20 24
64 2E 0D 0A 24 50 65 6E
                                                                                      ♪■$PCjr♪■$PC Con
vertible♪■$Regis
                                                                                      ter value AX = $ indefined. №$Pen
0000000360: 69 0L 0
0000000370: 75 6C 74 69 6D 61
0000000380: 0D 0A 24 53 79 73
0000000380: 6E 3A 20 20 20 2E
75 6D 62
                                                                                      ultimate byte:oo
                                                                    72 73 69 6F
65 72 69 61
                                                                                      ♪

$System versio
                                                                                       n: . №$Seria
                                        65 72
65 72
72 3A
0F 3C
86 C4
                                                   20 4F 45 4D
69 61 6C 20
20 20 20 20
09 76 02 04
               6C 20 6E 75 6D 62
20 20 0D 0A 24 53
00000003A0: 6C
                                                                                       1 number OEM:
                                                                    3A 20 20 20
                                                   69 61 6C 20 6E 75 6D 62
20 20 20 20 20 20 20 20
09 76 02 04 07 04 30 C3
B1 04 D2 E8 E8 E6 FF 59
                                                                                         ♪≊$Serial numb
00000003B0:
                                                                                      00000003C0: 65 72 20 75 73 65 72
00000003D0: 20 20 0D 0A 24 24 0F
00000003E0: 51 8A E0 E8 EF FF 86
00000003F0: C3 53 8A FC E8 E9
                                                       4F 88 05 4F 8A C7 E8
                                        FF 88
0000000400: DE FF 88 25 4F 88 05
                                                                    E4 33 D2 B9
                                            5B
                                                       51 52
0000000410: 0A 00 F7 F1 80 CA
                                                            33 D2
                                                                    3D ØA ØØ 73
                                        30 88
0000000420: F1 3C
                        00 74 04 0C
                                                    04 5A 59
                                                                    B4 09 CD 21
                                                       FF EB 01 90
90 3C FE 75
0000000430: C3 B8 00 F0 8E C0
                                                                                          ðŽÀ& þÿë⊕⊡<ÿı
                                        26 A0
                                                                                       oºV@èåÿëa®<buoº
```

Рисунок 5 - загрузочный модуль плохого .ЕХЕ

```
0000000190: 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
 00000001B0: 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
 00000001D0: 00 00 00 00 00 00 00 00
                                                                                                          00 00 00 00 00 00 00
 00000001E0: 00 00 00 00 00 00 00 00
                                                                                                          00 00 00 00 00 00 00 00
00000001F0: 00 00 00 00 00 00 00
                                                                                                          00 00 00 00 00 00 00 00
PC/ms\PC/XT/ms\AT/
ms\PS2 model 30/ms
$PS2 model 80/ms
                                                                                                                                                                              PCjr > PC
                                                                                                                                                                                value AX = $ind efined. №$Penult
                                                                                                          0A 24 50 65 6E 75 6C 74
 00000000280: 69 6D 61 74 65 20 62
00000000290: 53 79 73 74 65 6D 20
                                                                                                          74 65 3A 09 09 0D 0A 24
                                                                                                                                                                                imate byte:00⊅≊$
                                                                                                                 72 73 69 6F 6E 3A 20
                                                                                                                                                                                System version:
                                                                                                          53 65 72 69 61 6C 20 6E
4D 3A 20 20 20 20 20 0D
 00000002A0: 20 20 2E 20 20 0D 0A 24
                                                                                                                                                                                            ⊅≊$Serial n
  00000002B0: 75 6D 62 65 72 20 4F 45
                                                                                                                                                                                umber OEM:
                                                                                                          20 6E 75 6D 62 65 72 20
20 20 20 20 20 20 20 0D
 00000002C0: 0A 24 53 65 72 69 61 6C
                                                                                                                                                                                ⊠$Serial number
 000000002D0: 75 73 65 72 3A 20 20 20 0000000002E0: 0A 24 00 00 00 00 00 00 00
                                                                                                                                                                                user:
                                                                                                          00 00 00 00 00 00 00
00000002F0: E9 6C 01 24 0F 3C 09 76
0000000300: E0 E8 EF FF 86 C4 B1 04
                                                                                                          02 04 07 04 30 C3 51 8A
                                                                                                                                                                                él⊕$¢<ov⊕♦•♦0ÃQŠ
                                                                                                                                                                                àèïÿ†Ä±+ÒèèæÿYÃS

Śüèéÿ %0°+0ŠÇèÞÿ

%0°+[ÃQR2ä3Ò¹s

֖€Ê0°¶N3Ò=s sñ
        t+90°+ZYðoÍ!Ã.

                                                                                                          D2 E8 E8 E6 FF 59 C3 53
 0000000310: 8A FC E8 E9 FF 88 25 4F
                                                                                                          88 05 4F 8A C7 E8 DE FF
 0000000320: 88 25 4F 88 05 5B C3 51
                                                                                                                 32 E4 33 D2 B9 ØA ØØ
                                                                                                          33 D2 3D 0A 00 73 F1 3C
59 C3 B4 09 CD 21 C3 B8
 0000000330: F7 F1 80 CA 30 88 14 4E
 0000000340: 00 74 04 0C 30 88 04 5A
0000000350: 00 F0 8E C0 26 A0 FE FF
                                                                                                                                                                                  ðŽÀ& þÿë⊕2<ÿuoº
èåÿëq2<þuoº♣ è
                                                                                                          EB 01 90 3C FF 75 09 BA
 0000000330: 00 10 52 63 20 A6 F2 F7
00000000360: 00 00 E8 E5 FF EB 71 90
00000000370: D8 FF EB 64 90 3C FB 75
                                                                                                         3C FE 75 09 BA 05 00 E8 09 BA 05 00 E8 CB FF EB
    000000370: D8 FF EB 64 90 3C
                                                                                                                                                                                Øÿëd⊡<ûuoº♣ èËÿë
```

Рисунок 6 - загрузочный модуль хорошего .EXE

Отличия форматов файлов СОМ и ЕХЕ модулей

1. Какова структура файла COM? С какого адреса располагается код? Модуль сот-файла содержит данные и код, которые располагаются в одном сегменте, начиная с нулевого адреса. При загрузке происходит смещение в

- 256 байт (org 100h), генерируется стек, который занимает все оставшуюся память.
- 2. Какова структура файла "плохого" EXE? С какого адреса располагается код? Модуль плохого ехе-файла также содержит данные и код, которые располагаются в одном сегменте, но начиная с трехсотого (в 16 системе счисления) адреса. С 0 адреса до 300h располагаются заголовок информация для загрузчика, расположенная в начале файла, и таблица настройки адресов, 100h отводится под PSP.
- 3. Какова структура файла " хорошего " EXE? Чем он отличается от файла "плохого" EXE?

Модуль хорошего ехе-файла содержит стек, данные и код, расположенные в отдельных сегментах, в отличии от "плохого" EXE. Различается также расположение кода с 200h, так как в "хорошем" EXE нет смещения на 100h.

Загрузка СОМ модуля в основную память

- 1. Какой формат загрузки модуля COM? С какого адреса располагается код? Во время загрузки COM программы система выделяет первый свободный сегмент памяти и в его начале (первые 256 байт) размещается PSP. За PSP располагается код программы, то есть с адреса 100h. В IP счетчик команд, устанавливается значение 100h. В стек записывается адрес 0000h, для возврата по ret.
- 2. Что располагается с адреса 0?

С адреса 0 располагается PSP, занимающий 256 байт.

- 3. Какие значения имеют сегментные регистры? На какие области память они указывают?
 - Все сегментные регистры содержат адрес PSP, то есть значение 48DD.
- 4. Как определяется стек? Какую область память он занимает? Какие адреса? Указатель стека устанавливается на конец сегмента программы (FFFE). Стек, располагающийся с старших адресов, занимает всю доступную память, не зарезервированную PSP и кодом.

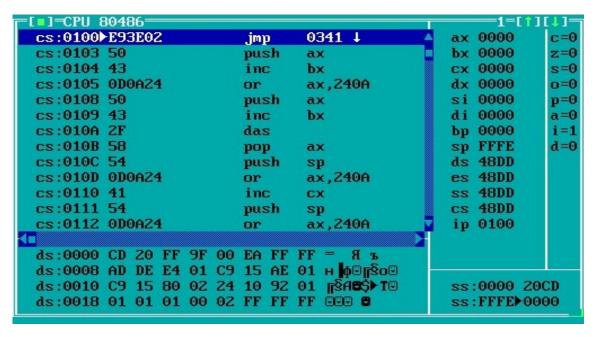


Рисунок 7 - Результат загрузки СОМ файла в память

Загрузка "хорошего" ЕХЕ модуля в основную память

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

Определяется сегментный адрес свободного участка памяти, размер которого достаточен для размещения программы. DS и ES указывают на начало PSP, CS указывает на начало сегмента кода, SS на начало сегмента стека (значения, указанные в заголовке). Управление передается по адресу, указанному в заголовке.

- 2. На что указывают регистры DS и ES? Регистры DS и ES указывают на PSP.
- 3. Как определяется стек?

Стек определяется с помощью упрощенной директивы .stack (или стандартной имя SEGMENT STACK ... имя ENDS), задается размер стека (в соответствии с моделью памяти). SS - указывает на начало сегмента стека (значения, указанные в заголовке), SP - на его смещение.

4. Как определяется точка входа?

Точкой входа в программу может быть метка, объявленная в директиве END - директива, которой завершается программа, так как метка не является

обязательным параметром этой директивы, то точкой входа, в таком случае, будет начало сегмента кода.

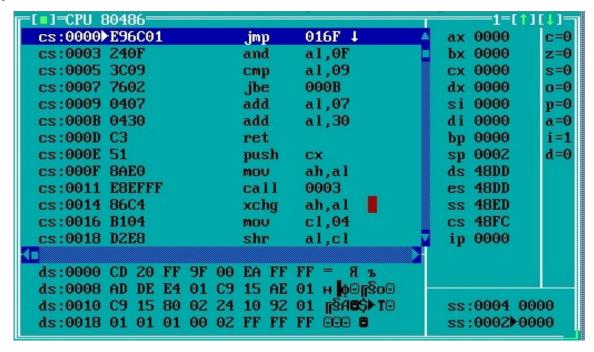


Рисунок 8 - Результат загрузки ЕХЕ файла в память

Выводы.

В результате выполнения лабораторной работы были исследованы различия в структурах исходных текстов и модулей СОМ и EXE файлов и способах их загрузки в основную память.

ПРИЛОЖЕНИЕ А

COMMENT *

МАКСИМОВА АНАСТАСИЯ, ГРУППА 8383 TESTPC SEGMENT ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING ORG 100H START: JMP BEGIN ; ДАННЫЕ 'PC', ODH, OAH, '\$' TYPE1 DB 'PC/XT', ODH, OAH, '\$' TYPE2 DB 'AT', ODH, OAH, '\$' ;TYPE5 TYPE3 DB = TYPE3 'PS2 MODEL 30', ODH, OAH, '\$' TYPE4 DB TYPE6 'PS2 MODEL 80', ODH, OAH, '\$' DB 'PCJR', ODH, OAH, '\$' TYPE7 DB 'PC CONVERTIBLE', ODH, OAH, '\$' TYPE8 DB STRING DB 'REGISTER VALUE AX = ', '\$' 'INDEFINED.', ODH, OAH, '\$' VAR0 STR1 DB 'PENULTIMATE BYTE: ', ODH, OAH, VAR0 STR2 DB '\$' 'SYSTEM VERSION: . ', ODH, OAH, '\$' SYSTEM_VERSION DB 'SERIAL NUMBER OEM: ', ODH, OAH, '\$' SERIAL NUMB OEM DB 'SERIAL NUMBER USER: ', ODH, OAH, '\$' SERIAL NUMB USER DB ;ПРОЦЕДУРЫ ;-----TETR TO HEX PROC NEAR AND AL, OFH CMP AL, 09 JBE NEXT ADD AL, 07 NEXT: ADD AL, 30H RET TETR TO HEX ENDP BYTE TO HEX PROC NEAR

;БАЙТ В АL ПЕРЕВОДИТСЯ В ДВА СИМВОЛА ШЕСТН. ЧИСЛА В АХ

```
MOV AH, AL
                 CALL TETR_TO_HEX
                      AL, AH
                 XCHG
                 MOV
                      CL, 4
                 SHR
                      AL, CL
                      TETR TO HEX ;В AL - СТАРШАЯ ЦИФРА
                 CALL
                 POP
                      CX
                                  ;В АН - МЛАДШАЯ
                 RET
BYTE TO HEX ENDP
;-----
WRD TO HEX
            PROC NEAR
;перевод в 16 С/С 16-ти разрядного числа
;В АХ - 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
                          ВХ
                 RET
WRD TO HEX
            ENDP
;-----
BYTE TO DEC
            PROC NEAR
;ПЕРЕВОД В 10 С/С, SI - АДРЕС ПОЛЯ МЛАДШЕЙ ЦИФРЫ
                 PUSH CX
                 PUSH DX
                          AH, AH
                 XOR
                 XOR
                          DX, DX
                          CX, 10
                 MOV
            DIV CX
LOOP BD:
                          DL, 30H
                 OR
                 MOV
                          [SI], DL
                 DEC
                          SI
                 XOR
                          DX, DX
```

PUSH CX

AX, 10 CMP JAE LOOP_BD AL, 00H CMP END L JΕ AL, 30H OR MOV [SI], AL END L: POP DX POP CX RET BYTE TO DEC ENDP ;-----PRINTF PROC NEAR MOV AH, 09H INT 21H RET ENDP ;-----TYPE PC PROC NEAR MOV AX, OFOOOH MOV ES, AX MOV AL, ES:[OFFFEH] JMP SEARCH TYPE SEARCH_TYPE: CMP AL, OFFH VAR2 1 JNE MOV DX, OFFSET TYPE1 CALL PRINTF JMP EXIT VAR2 1: CMP AL, OFEH VAR2 2 JNE VOM DX, OFFSET TYPE2 CALL PRINTF JMP EXIT VAR2 2: CMP AL, OFBH VAR3 JNE DX, OFFSET TYPE2 MOV CALL PRINTF JMP EXIT

VAR3: CMP AL, OFCH

JNE VAR4

MOV DX, OFFSET TYPE3

CALL PRINTF

JMP EXIT

VAR4: CMP AL, OFAH

JNE VAR6

MOV DX, OFFSET TYPE4

CALL PRINTF

JMP EXIT

VAR6: CMP AL, 0F8H

JNE VAR7

MOV DX, OFFSET TYPE6

CALL PRINTF

JMP EXIT

VAR7: CMP AL, OFDH

JNE VAR8

MOV DX, OFFSET TYPE7

CALL PRINTF

JMP EXIT

VAR8: CMP AL, OF9H

JNE VAR0

MOV DX, OFFSET TYPE8

CALL PRINTF

JMP EXIT

VAR0:

MOV DX, OFFSET VAR0_STR1

CALL PRINTF

MOV DI, OFFSET VARO_STR2

ADD DI, 21

CALL WRD_TO_HEX

MOV DX, OFFSET VAR0_STR2

CALL PRINTF

EXIT:

RET

```
TYPE PC
      ENDP
;-----
CORRECT
              PROC NEAR
              PUSH BX
               MOV BL, 16 ;BL - ДЕЛИТЕЛЬ
              DIV BL ;АН - ОСТАТОК AL - ЦЕЛОЕ
               ADD AH, 'O'
              ADD AL, '0'
              POP BX
               RET
CORRECT
              ENDP
;-----
VERSION SYSTEM PROC NEAR
              MOV
                     АН, 30Н
               INT
                     21H
               PUSH CX
               MOV CX, AX ;SAVE
               ;AL
               CALL CORRECT
               MOV SI, OFFSET SYSTEM_VERSION
                     SI, 17
               ADD
               MOV
                      [SI], AH
               DEC
                      SI
                 [SI], AL
               MOV
               ;AH
               XOR AX, AX
               MOV AH, CH
               CALL CORRECT
               ADD SI, 4
               MOV
                      [SI], AH
               DEC SI
               MOV
                     [SI], AL
               MOV DX, OFFSET SYSTEM_VERSION
```

CALL PRINTF

;SERIAL NUMB OEM

XOR AX, AX

MOV AL, BH

CALL CORRECT

MOV SI, OFFSET SERIAL_NUMB_OEM

ADD SI, 20 MOV [SI], AH

DEC SI

MOV [SI], AL

MOV DX, OFFSET SERIAL_NUMB_OEM

CALL PRINTF

;SERIAL NUMB USER

;BL

POP CX

XOR AX, AX

MOV AL, BL

CALL CORRECT

MOV SI, OFFSET SERIAL_NUMB_USER

ADD SI, 21 MOV [SI], AH

DEC SI

MOV [SI], AL

;CH

XOR AX, AX

MOV AL, CH

CALL CORRECT

ADD SI, 3

MOV [SI], AH

DEC SI

MOV [SI], AL

;CL

XOR AX, AX
MOV AL, CL

CALL CORRECT

ADD SI, 3

MOV [SI], AH

DEC SI

MOV [SI], AL

MOV DX, OFFSET SERIAL_NUMB_USER

CALL PRINTF

RET

VERSION_SYSTEM ENDP

;-----

BEGIN:

;ВЫВОД СТРОКИ ТЕКСТА ИЗ ПОЛЯ STRING

MOV DX, OFFSET STRING

MOV AH, 09H

INT 21H

; ВЫПОЛНЕНИЕ ЗАДАНИЙ

CALL TYPE_PC

CALL VERSION_SYSTEM

;ВЫХОД В DOS

XOR AL, AL

MOV AH, 4CH

INT 21H

TESTPC ENDS

END START ; КОНЕЦ МОДУЛЯ, START - ТОЧКА ВХОДА

приложение б

COMMENT * МАКСИМОВА АНАСТАСИЯ, ГРУППА 8383 ASTACK SEGMENT STACK DW 100H ASTACK ENDS DATA SEGMENT ; ДАННЫЕ 'PC', ODH, OAH, '\$' TYPE1 DB 'PC/XT', ODH, OAH, '\$' TYPE2 DB 'AT', ODH, OAH, '\$' TYPE3 DB ; TYPE5 = TYPE3 'PS2 MODEL 30', ODH, OAH, '\$' TYPE4 DB 'PS2 MODEL 80', ODH, OAH, '\$' TYPE6 DB TYPE7 DB 'PCJR', ODH, OAH, '\$' TYPE8 'PC CONVERTIBLE', ODH, OAH, '\$' DB DB STRING 'REGISTER VALUE AX = ', '\$' 'INDEFINED.', ODH, OAH, '\$' VAR0 STR1 DB VAR0 STR2 'PENULTIMATE BYTE: ', ODH, OAH, DB 'SYSTEM VERSION: . ', ODH, OAH, '\$' SYSTEM VERSION DB 'SERIAL NUMBER OEM: ', ODH, OAH, '\$' SERIAL NUMB OEM DB 'SERIAL NUMBER USER: ', ODH, OAH, '\$' SERIAL NUMB USER DB DATA ENDS CODE SEGMENT ASSUME CS:CODE, DS:DATA, ES:NOTHING, SS:ASTACK START: JMP BEGIN ;ПРОЦЕДУРЫ ;-----

1\$1

TETR TO HEX PROC NEAR

```
AND AL, OFH
          CMP
              AL, 09
         JBE NEXT
         ADD AL, 07
NEXT: ADD AL, 30H
         RET
TETR_TO_HEX
         ENDP
;-----
BYTE TO HEX
          PROC NEAR
;БАЙТ В АL ПЕРЕВОДИТСЯ В ДВА СИМВОЛА ШЕСТН. ЧИСЛА В АХ
               PUSH CX
               MOV AH, AL
               CALL TETR TO HEX
               XCHG AL, AH
               MOV
                    CL, 4
               SHR AL, CL
               CALL
                    ТЕТЯ ТО НЕХ ;В AL - СТАРШАЯ ЦИФРА
               POP CX
                               ;В АН - МЛАДШАЯ
               RET
BYTE_TO_HEX ENDP
;-----
WRD TO HEX PROC NEAR
;перевод в 16 С/С 16-ти разрядного числа
; В АХ - 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
                       ВХ
               RET
WRD TO HEX ENDP
;-----
BYTE TO DEC PROC NEAR
```

;ПЕРЕВОД В 10 С/С, SI - АДРЕС ПОЛЯ МЛАДШЕЙ ЦИФРЫ

			DHOH	017	
			PUSH		
			PUSH		
			XOR		AH, AH
			XOR		DX, DX
			VOM		CX, 10
LOOP_BD:		DIV		CX	
			OR		DL, 30H
			VOM		[SI], DL
			DEC		SI
			XOR		DX, DX
			CMP		AX, 10
			JAE		LOOP_BD
			CMP		AL, OOH
			JE		END L
			OR		AL, 30H
			MOV		[SI], AL
END L:			POP		
_			POP		CX
			RET		
BYTE_TO_D	EC	ENDP			
;					
PRINTF		PROC	NEAR		
	MOV		AH, 09	Э Н	
	INT		1111, 03		
	RET	2111			
PRINTF	ТШТ	ENDP			
•		ENDI			
,		DDAC	NEAR		
TYPE_PC		PROC			AV 00000H
					AX, OFOOOH
					ES, AX
					ES:[OFFFEH]
			JMP	SEARC	CH_TYPE
SEARCH_TY	PE:				
			CMP	AL, C	
			JNE		VAR2_1
			MOV		DX, OFFSET TYPE1
			CALL	PRINT	r.
			JMP		EXIT
VAR2_1:					
			CMP	AL, C	Эген
			CMP JNE	AL, (FEH VAR2_2
				AL, 0	

CALL PRINTF

JMP EXIT

VAR2 2: CMP AL, OFBH

JNE VAR3

MOV DX, OFFSET TYPE2

CALL PRINTF

JMP EXIT

VAR3: CMP AL, OFCH

JNE VAR4

MOV DX, OFFSET TYPE3

CALL PRINTF

JMP EXIT

VAR4: CMP AL, OFAH

JNE VAR6

MOV DX, OFFSET TYPE4

CALL PRINTF

JMP EXIT

VAR6: CMP AL, OF8H

JNE VAR7

MOV DX, OFFSET TYPE6

CALL PRINTF

JMP EXIT

VAR7: CMP AL, OFDH

JNE VAR8

MOV DX, OFFSET TYPE7

CALL PRINTF

JMP EXIT

VAR8: CMP AL, OF9H

JNE VAR0

MOV DX, OFFSET TYPE8

CALL PRINTF

JMP EXIT

VAR0:

MOV DX, OFFSET VAR0 STR1

CALL PRINTF

MOV DI, OFFSET VARO_STR2

ADD DI, 21

CALL WRD_TO_HEX

MOV DX, OFFSET VAR0 STR2

CALL PRINTF

EXIT:

RET

TYPE PC ENDP

;-----

CORRECT PROC NEAR

PUSH BX

MOV BL, 16 ;BL - ДЕЛИТЕЛЬ

DIV BL ;АН - ОСТАТОК AL - ЦЕЛОЕ

ADD AH, 'O'

ADD AL, '0'

POP BX

RET

CORRECT ENDP

;-----

VERSION_SYSTEM PROC NEAR

MOV AH, 30H

INT 21H

PUSH CX

MOV CX, AX ;SAVE

;AL

CALL CORRECT

MOV SI, OFFSET SYSTEM_VERSION

ADD SI, 17

MOV [SI], AH

DEC SI

MOV [SI], AL

;AH

XOR AX, AX

MOV AH, CH

CALL CORRECT

ADD SI, 4

MOV [SI], AH

DEC SI

MOV [SI], AL

MOV DX, OFFSET SYSTEM VERSION

CALL PRINTF

;SERIAL_NUMB_OEM

XOR AX, AX

MOV AL, BH

CALL CORRECT

MOV SI, OFFSET SERIAL NUMB OEM

ADD SI, 20

MOV [SI], AH

DEC SI

MOV [SI], AL

MOV DX, OFFSET SERIAL_NUMB_OEM

CALL PRINTF

;SERIAL_NUMB_USER

;BL

POP CX

XOR AX, AX

MOV AL, BL

CALL CORRECT

MOV SI, OFFSET SERIAL_NUMB_USER

ADD SI, 21

MOV [SI], AH

DEC SI

MOV [SI], AL

;CH

XOR AX, AX

MOV AL, CH

CALL CORRECT

ADD SI, 3

MOV [SI], AH

DEC SI

MOV [SI], AL

;CL

XOR AX, AX

MOV AL, CL

CALL CORRECT

ADD SI, 3

MOV [SI], AH

DEC SI

MOV [SI], AL

MOV DX, OFFSET SERIAL NUMB USER

CALL PRINTF

RET

VERSION SYSTEM ENDP

;-----

BEGIN:

PUSH DS

XOR AX, AX

PUSH AX

MOV AX, DATA

MOV DS, AX

;ВЫВОД СТРОКИ ТЕКСТА ИЗ ПОЛЯ STRING

MOV DX, OFFSET STRING

MOV AH, 09H

INT 21H

; выполнение заданий

CALL TYPE PC

CALL VERSION_SYSTEM

;выход в dos

XOR AL, AL

MOV AH, 4CH

INT 21H

CODE ENDS

END START ; КОНЕЦ МОДУЛЯ, START - ТОЧКА ВХОДА