

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. 8382

Нечепуренко Н.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.

Согласно методическим указаниям был написан .com модуль, определяющий тип устройства, версию операционной системы, серийный номер OEM и серийный номер пользователя. Результат работы .com модуля приведен на рисунке 1.

```
C:\>lab1_com.com
IBM PC TYPE IS: PS2 model 50 or 60 or AT
Version: 5.0
OEM: 0
Serial number: 0 0
```

Рисунок 1 – Результат работы .com модуля

Из исходного кода .com модуля построим .exe модуль. Результат работы «плохого» .exe файла приведен на рисунке 2.

```

FEXD 0+0-ESD+010
      000IBM PC TYPE IS:
FEXD 0+0-ESD+010
      000IBM PC TYPE IS:  5.0
FEXD 0+0-ESD+010
      000IBM PC TYPE IS:
                                     5.0
      FEXD 0+0-ESD+010
      000IBM PC TYPE IS:
FEXD 0+0-ESD+010
      000IBM PC TYPE IS:  0
FEXD 0+0-ESD+010
      000IBM PC TYPE IS:
                                     5.0
                                     0
      FEXD 0+0-ESD+010
      000IBM PC TYPE IS:
FEXD 0+0-ESD+010
      000IBM PC TYPE IS:  0 0 FEXD 0+0-ESD+010
                                     000IBM PC TYPE IS:
5.0      0      0 0 FEXD 0+0-ESD+010
                                     000IBM
PC TYPE IS:

```

Рисунок 2 – Результат работы «плохого» .exe модуля

Немного изменим код .com модуля, чтобы получить «хороший» .exe модуль, полный исходный код представлен в Приложении Б. Результат работы полученного модуля приведен на рисунке 3.

```
C:\>lab1_exe.exe
IBM PC TYPE IS: PS2 model 50 or 60 or AT
Version: 5.0
OEM: 0
Serial number: 0 0
```

Рисунок 3 – Результат работы «хорошего» .exe модуля

Таким образом, были получены .com и .exe модули, выводящие на экран необходимую информацию.

Проанализируем шестнадцатеричное представление полученных модулей с помощью утилиты hexdump. Результаты представлены на рисунках ниже.

```
nechn@LAPTOP-QR5VS0FQ:~$ hd lab1_com.com
00000000 e9 01 01 49 42 4d 20 50 43 20 54 59 50 45 20 49 |...IBM PC TYPE I|
00000010 53 3a 20 24 0d 0a 24 50 43 24 50 43 2f 58 54 24 |S: $..$PC$PC/XT$|
00000020 50 53 32 20 6d 6f 64 65 6c 20 33 30 24 50 53 32 |PS2 model 30$PS2|
00000030 20 6d 6f 64 65 6c 20 35 30 20 6f 72 20 36 30 20 | model 50 or 60 |
00000040 6f 72 20 41 54 24 50 53 32 20 6d 6f 64 65 6c 20 |or AT$PS2 model |
00000050 38 30 24 50 53 6a 72 24 50 53 20 43 6f 6e 76 65 |80$PSjr$PS Conve|
00000060 6e 74 69 62 6c 65 24 56 65 72 73 69 6f 6e 3a 24 |ntible$Version:$|
00000070 3c 20 32 2e 30 24 20 20 20 20 20 20 24 4f 45 4d |< 2.0$ $OEM|
00000080 3a 24 20 20 24 53 65 72 69 61 6c 20 6e 75 6d |:$ $Serial num|
00000090 62 65 72 3a 24 20 20 20 20 20 20 20 20 24 00 |ber:$ $.|
000000a0 00 00 00 00 00 00 00 00 00 00 00 00 24 0f 3c |.....$.<|
000000b0 09 76 02 04 07 04 30 c3 51 8a e0 e8 ef ff 86 c4 |.v....0.Q.....|
000000c0 b1 04 d2 e8 e8 e6 ff 59 c3 53 8a fc e8 e9 ff 88 |.....Y.S.....|
000000d0 25 4f 88 05 4f 8a c7 e8 de ff 88 25 4f 88 05 5b |%0..0.....%0..[|
000000e0 c3 51 52 32 e4 33 d2 b9 0a 00 f7 f1 80 ca 30 88 |.QR2.3.....0..|
000000f0 14 4e 33 d2 3d 0a 00 73 f1 3c 00 74 04 0c 30 88 |.N3.=...s.<.t..0.|
00000100 04 5a 59 c3 b9 00 00 ba 46 01 89 97 9f 01 83 c3 |.ZY.....F.....|
00000110 02 ba 58 01 89 97 9f 01 83 c3 02 ba 20 01 89 97 |..X.....|
00000120 9f 01 83 c3 02 ba 1a 01 89 97 9f 01 83 c3 02 ba |.....|
00000130 2d 01 89 97 9f 01 83 c3 02 ba 53 01 89 97 9f 01 |-.....S.....|
00000140 83 c3 02 ba 1a 01 89 97 9f 01 83 c3 02 ba 17 01 |.....|
00000150 89 97 9f 01 bb 00 00 ba 03 01 b4 09 cd 21 b8 00 |.....!..|
00000160 f0 8e c0 26 a0 fe ff b4 00 2c f8 d0 e0 8b d8 8b |...&.....|
00000170 97 9f 01 b4 09 cd 21 ba 14 01 cd 21 ba 67 01 cd |.....!....!..g..|
00000180 21 b4 30 cd 21 3c 00 74 2c be 76 01 56 83 c6 01 |!.0.!<.t,.v.V...|
00000190 50 e8 4d ff 58 5e 56 83 c6 02 c6 04 2e 5e 83 c6 |P.M.X^V.....^..|
000001a0 03 86 c4 e8 3b ff b4 09 ba 76 01 cd 21 ba 14 01 |....;....v..!...|
000001b0 cd 21 eb 0d 90 ba 70 01 b4 09 cd 21 ba 14 01 cd |!....p....!...|
000001c0 21 ba 7d 01 cd 21 8a c7 be 82 01 83 c6 02 e8 10 |!.}...!.....|
000001d0 ff b4 09 ba 82 01 cd 21 ba 14 01 cd 21 ba 86 01 |.....!.....|
000001e0 cd 21 be 95 01 83 c6 07 8b c1 e8 f4 fe 4e 8a c3 |!......N..|
000001f0 e8 ee fe ba 95 01 b4 09 cd 21 ba 14 01 cd 21 32 |.....!.....!2|
00000200 c0 b4 4c cd 21 |..L.!|
00000205
```

Рисунок 4 – Представление .com модуля

```

nechn@LAPTOP-QR5VS0FQ:~$ hd lab1_com.exe
00000000  4d 5a 05 01 03 00 00 00  20 00 00 00 ff ff 00 00  |MZ.....|
00000010  00 00 ef 8c 00 01 00 00  1e 00 00 00 01 00 00 00  |.....|
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |.....|
*
00000300  e9 01 01 49 42 4d 20 50  43 20 54 59 50 45 20 49  |...IBM PC TYPE I|
00000310  53 3a 20 24 0d 0a 24 50  43 24 50 43 2f 58 54 24  |S: $..$PC$PC/XT$|
00000320  50 53 32 20 6d 6f 64 65  6c 20 33 30 24 50 53 32  |PS2 model 30$PS2|
00000330  20 6d 6f 64 65 6c 20 35  30 20 6f 72 20 36 30 20  |model 50 or 60|
00000340  6f 72 20 41 54 24 50 53  32 20 6d 6f 64 65 6c 20  |or AT$PS2 model|
00000350  38 30 24 50 53 6a 72 24  50 53 20 43 6f 6e 76 65  |80$PSjr$PS Conve|
00000360  6e 74 69 62 6c 65 24 56  65 72 73 69 6f 6e 3a 24  |ntible$Version:$|
00000370  3c 20 32 2e 30 24 20 20  20 20 20 20 24 4f 45 4d  |< 2.0$ $OEM|
00000380  3a 24 20 20 20 24 53 65  72 69 61 6c 20 6e 75 6d  |:$ $Serial num|
00000390  62 65 72 3a 24 20 20 20  20 20 20 20 20 24 00 00  |ber:$ $.|
000003a0  00 00 00 00 00 00 00 00  00 00 00 00 24 0f 3c 00  |.....$.<|
000003b0  09 76 02 04 07 04 30 c3  51 8a e0 e8 ef ff 86 c4  |.v....0.Q.....|
000003c0  b1 04 d2 e8 e8 e6 ff 59  c3 53 8a fc e8 e9 ff 88  |.....Y.S.....|
000003d0  25 4f 88 05 4f 8a c7 e8  de ff 88 25 4f 88 05 5b  |%O..O.....%O..|
000003e0  c3 51 52 32 e4 33 d2 b9  0a 00 f7 f1 80 ca 30 88  |.QR2.3.....0.|
000003f0  14 4e 33 d2 3d 0a 00 73  f1 3c 00 74 04 0c 30 88  |.N3.=..s.<.t..0.|
00000400  04 5a 59 c3 b9 00 00 ba  46 01 89 97 9f 01 83 c3  |.ZY.....F.....|
00000410  02 ba 58 01 89 97 9f 01  83 c3 02 ba 20 01 89 97  |..X.....|
00000420  9f 01 83 c3 02 ba 1a 01  89 97 9f 01 83 c3 02 ba  |.....|
00000430  2d 01 89 97 9f 01 83 c3  02 ba 53 01 89 97 9f 01  |-.....S.....|
00000440  83 c3 02 ba 1a 01 89 97  9f 01 83 c3 02 ba 17 01  |.....|
00000450  89 97 9f 01 bb 00 00 ba  03 01 b4 09 cd 21 b8 00  |.....!..|
00000460  f0 8e c0 26 a0 fe ff b4  00 2c f8 d0 e0 8b d8 8b  |...&.....,.....|
00000470  97 9f 01 b4 09 cd 21 ba  14 01 cd 21 ba 67 01 cd  |.....!....!.g..|
00000480  21 b4 30 cd 21 3c 00 74  2c be 76 01 56 83 c6 01  |!.0.!<.t,.v.V...|
00000490  50 e8 4d ff 58 5e 56 83  c6 02 c6 04 2e 5e 83 c6  |P.M.X^V.....^..|
000004a0  03 86 c4 e8 3b ff b4 09  ba 76 01 cd 21 ba 14 01  |....;...v...!...|
000004b0  cd 21 eb 0d 90 ba 70 01  b4 09 cd 21 ba 14 01 cd  |.!....p...!....|
000004c0  21 ba 7d 01 cd 21 8a c7  be 82 01 83 c6 02 e8 10  |!.}...!.....|
000004d0  ff b4 09 ba 82 01 cd 21  ba 14 01 cd 21 ba 86 01  |.....!....!...|
000004e0  cd 21 be 95 01 83 c6 07  8b c1 e8 f4 fe 4e 8a c3  |.!......N..|
000004f0  e8 ee fe ba 95 01 b4 09  cd 21 ba 14 01 cd 21 32  |.....!.....!2|
00000500  c0 b4 4c cd 21  |...L.!!|
00000505

```

Рисунок 5 – Представление «плохого» .exe модуля

```

nechn@LAPTOP-QR5VS0FQ:~$ hd lab1_exe.exe
00000000  4d 5a e1 00 03 00 01 00  20 00 00 00 ff ff 00 00  |MZ.....|
00000010  c8 00 7d 14 57 00 18 00  1e 00 00 00 01 00 5c 00  |..}.W.....\.|
00000020  18 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |.....|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |.....|
*
00000200  2a 00 2a 00 2a 00 2a 00  2a 00 2a 00 2a 00 2a 00  |*.*.*.*.*.*.*|
*
000002c0  2a 00 2a 00 2a 00 2a 00  00 00 00 00 00 00 00 00  |*.*.*.*.....|
49 42 4d 20 50 43 20 54  59 50 45 20 49 53 3a 20  |IBM PC TYPE IS:|
000002e0  24 0d 0a 24 50 43 24 50  43 2f 58 54 24 50 53 32  |$..$PC$PC/XT$PS2|
000002f0  20 6d 6f 64 65 6c 20 33  30 24 50 53 32 20 6d 6f  |model 30$PS2 mo|
00000300  64 65 6c 20 35 30 20 6f  72 20 36 30 20 6f 72 20  |del 50 or 60 or|
00000310  41 54 24 50 53 32 20 6d  6f 64 65 6c 20 38 30 24  |AT$PS2 model 80$|
00000320  50 53 6a 72 24 50 53 20  43 6f 6e 76 65 6e 74 69  |PSjr$PS Conventi|
00000330  62 6c 65 24 56 65 72 73  69 6f 6e 3a 24 3c 20 32  |ble$Version:$< 2|
00000340  2e 30 24 20 20 20 20 20  20 24 4f 45 4d 3a 24 20  |.0$ $OEM:$|
00000350  20 20 24 53 65 72 69 61  6c 20 6e 75 6d 62 65 72  |$Serial number|
00000360  3a 24 20 20 20 20 20 20  20 20 20 24 00 00 00 00  |:$ $....|
00000370  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |.....|
00000380  24 0f 3c 09 76 02 04 07  04 30 c3 51 8a e0 e8 ef  |$.<.v....0.Q....|
00000390  ff 86 c4 b1 04 d2 e8 e8  e6 ff 59 c3 53 8a fc e8  |.....Y.S...|
000003a0  e9 ff 88 25 4f 88 05 4f  8a c7 e8 de ff 88 25 4f  |...%0..0.....%0|
000003b0  88 05 5b c3 51 52 32 e4  33 d2 b9 0a 00 f7 f1 80  |..[.QR2.3.....|
000003c0  ca 30 88 14 4e 33 d2 3d  0a 00 73 f1 3c 00 74 04  |.0..N3.=.s.<.t.|
000003d0  0c 30 88 04 5a 59 c3 1e  2b c0 50 b8 0d 00 8e d8  |..0..ZY..+P....|
000003e0  b9 00 00 ba 43 00 89 97  9c 00 83 c3 02 ba 55 00  |....C.....U.|
000003f0  89 97 9c 00 83 c3 02 ba  1d 00 89 97 9c 00 83 c3  |.....|
00000400  02 ba 17 00 89 97 9c 00  83 c3 02 ba 2a 00 89 97  |.....*...|
00000410  9c 00 83 c3 02 ba 50 00  89 97 9c 00 83 c3 02 ba  |.....P.....|
00000420  17 00 89 97 9c 00 83 c3  02 ba 14 00 89 97 9c 00  |.....|
00000430  bb 00 00 ba 00 00 b4 09  cd 21 b8 00 f0 8e c0 26  |.....!.....&|
00000440  a0 fe ff b4 00 2c f8 d0  e0 8b d8 8b 97 9c 00 b4  |.....,.....|
00000450  09 cd 21 ba 11 00 cd 21  ba 64 00 cd 21 b4 30 cd  |..!....!.d..!.0.|
00000460  21 3c 00 74 2c be 73 00  56 83 c6 01 50 e8 44 ff  |!<.t,.s.V...P.D.|
00000470  58 5e 56 83 c6 02 c6 04  2e 5e 83 c6 03 86 c4 e8  |X^V.....^.....|
00000480  32 ff b4 09 ba 73 00 cd  21 ba 11 00 cd 21 eb 0d  |2....s..!....!..|
00000490  90 ba 6d 00 b4 09 cd 21  ba 11 00 cd 21 ba 7a 00  |..m....!....!.z.|
000004a0  cd 21 8a c7 be 7f 00 83  c6 02 e8 07 ff b4 09 ba  |.!.|
000004b0  7f 00 cd 21 ba 11 00 cd  21 ba 83 00 cd 21 be 92  |...!....!....!..|
000004c0  00 83 c6 07 8b c1 e8 eb  fe 4e 8a c3 e8 e5 fe ba  |.....N.....|

000004d0  92 00 b4 09 cd 21 ba 11  00 cd 21 32 c0 b4 4c cd  |.....!....!2..L.|
000004e0  21  ||
000004e1

```

Рисунок 6 – Представление «хорошего» .exe модуля

Рассмотрим код модулей в отладчике TD. Результаты представлены на рисунках ниже.

CPU 80486				1=[↑][↓]	
cs:0100	E90101	jmp	0204 ↓	ax	0000 c=0
cs:0103	49	dec	cx	bx	0000 z=0
cs:0104	42	inc	dx	cx	0000 s=0
cs:0105	4D	dec	bp	dx	0000 o=0
cs:0106	205043	and	[bx+si+43],dl	si	0000 p=0
cs:0109	205459	and	[si+59],dl	di	0000 a=0
cs:010C	50	push	ax	bp	0000 i=1
cs:010D	45	inc	bp	sp	FFFE d=0
cs:010E	204953	and	[bx+di+53],cl	ds	48DD
cs:0111	3A20	cmp	ah,[bx+si]	es	48DD
cs:0113	240D	and	al,0D	ss	48DD
cs:0115	0A24	or	ah,[si]	cs	48DD
cs:0117	50	push	ax	ip	0100
ds:0000 CD 20 FF 9F 00 EA FF FF = f 0				ss:0000 20CD	
ds:0008 AD DE E4 01 C9 15 AE 01				ss:FFFE 0000	
ds:0010 C9 15 80 02 24 10 92 01					
ds:0018 01 01 01 00 02 FF FF FF					

Рисунок 7 – Код .com модуля в отладчике TD

CPU 80486				1=[↑][↓]	
cs:0057	1E	push	ds	ax	0000 c=0
cs:0058	2BC0	sub	ax,ax	bx	0000 z=0
cs:005A	50	push	ax	cx	0000 s=0
cs:005B	B8FA48	mov	ax,48FA	dx	0000 o=0
cs:005E	8ED8	mov	ds,ax	si	0000 p=0
cs:0060	B90000	mov	cx,0000	di	0000 a=0
cs:0063	BA4300	mov	dx,0043	bp	0000 i=1
cs:0066	89979C00	mov	[bx+009C],dx	sp	00C8 d=0
cs:006A	83C302	add	bx,0002	ds	48DD
cs:006D	BA5500	mov	dx,0055	es	48DD
cs:0070	89979C00	mov	[bx+009C],dx	ss	48ED
cs:0074	83C302	add	bx,0002	cs	4905
cs:0077	BA1D00	mov	dx,001D	ip	0057
ds:0000 CD 20 FF 9F 00 EA FF FF = f 0				ss:00CA 0000	
ds:0008 AD DE E4 01 C9 15 AE 01				ss:00C8 0000	
ds:0010 C9 15 80 02 24 10 92 01					
ds:0018 01 01 01 00 02 FF FF FF					

Рисунок 8 – Код «хорошего» .exe модуля в отладчике TD

Контрольные вопросы.

Отличия исходных текстов COM и EXE программ:

1. Сколько сегментов должна содержать COM программа?

COM программа должна содержать один сегмент кода.

2. EXE программа?

EXE программа может иметь несколько сегментов кода, несколько сегментов данных, сегмент стека. Количество сегментов и их вид зависит от выбранной модели памяти, например, в модели small можно использовать один сегмент данных и один сегмент кода. Соответствия между моделями памяти и списком сегментов приведены в таблице ниже.

Таблица 1 – Модели памяти

Модель памяти	Сегменты
small	один сегмент кода, один сегмент данных
compact	один сегмент кода, несколько сегментов данных
medium	несколько сегментов кода, один сегмент данных
large	несколько сегментов кода, несколько сегментов данных
huge	много сегментов кода, много сегментов данных

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

Необходима директива `assume`, устанавливающая соответствие между сегментом кода и сегментным (-ми) регистрами (например, CS и DS). С помощью директивы `org` необходимо указать смещение для адресов 100h, потому что с нулевого адреса располагается PSP.

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

Нельзя использовать команды, использующие сегментные регистры. Также вводится ограничение на размер сегмента в байтах.

Отличия форматов файлов COM и EXE модулей

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

В самом начале COM файла расположен PSP, занимающий 256 байтов, затем идет единственный сегмент, содержащий и данные, и код. Размер модуля не может превышать 64КБ. Код начинается с адреса 100h (см. рис. 7), т.к. первые 256 байтов занимает PSP.

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

«Плохой» EXE имеет один сегмент. Код располагается с адреса 300h (см. рис. 5), перед ним с адреса 0 располагается 200h байт служебной информации с заголовком и настройками, а также 100h байт того же PSP.

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

«Хороший» EXE разделен на несколько сегментов, таких как сегмент данных, кода, стека. Перед сегментом кода и данных можно увидеть область стека, заполненного «*», на рисунке 6, что вносит различия между адресами начала кода в «хорошем» и «плохом» EXE файлах.

Загрузка COM модуля в основную память

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

Операционная система выделяет сегмент, в первые 256 байт помещает PSP, затем загружает сегмент кода, указатель стека устанавливается на конец выделенного сегмента. Код начинается с адреса 100h.

2. Что располагается с адреса 0?

Program Segment Prefix – область памяти в DOS программах, ответственная за состояние программы.

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

На рисунке 7 показано, что в сегментных регистрах находится значение 48DD. Сегментные регистры указывают на PSP. Благодаря `org 100h` первая команда сегмента кода начинается с правильного смещения, т.е. не затрагивает PSP.

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

Стек расположен в конце выделенной памяти. Хотя он и не определяется явным образом в исходном коде, им можно корректно пользоваться, но необходимо следить, чтобы размер стека не превысил некоторого порога и SP не пересекался с адресами команд. По адресу FFFF находится код возврата, SP указывает на FFFE и растет по уменьшению адресов.

Загрузка «хорошего» EXE модуля в основную память

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

SS=48ED, CS=4905, DS=ES=48DD. Регистры сегмента стека и сегмента кода устанавливаются на начала соответствующих сегментов. Регистры ES и DS устанавливаются на PSP.

2. На что указывают регистры ES и DS?

На Program Segment Prefix.

3. Как определяется стек?

Схожим образом с сегментом данных. Также выделяется блок памяти заданного объема. Для корректной работы необходимо в директиве `assume` указать `SS:<stack segment name>`.

4. Как определяется точка входа?

Точка входа в программу определяется с помощью директивы `END` (см. Приложение Б).

Выводы.

В процессе выполнения работы были реализованы исполняемые модули .com и .exe, выводящие на экран некоторые характеристики устройства и операционной системы. Были исследованы различия между вышеупомянутыми форматами модулей. Была проанализирована разница между «хорошим» и «плохим» .exe файлами.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД .COM МОДУЛЯ

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

```
IBMT db 'IBM PC TYPE IS: ', '$'
BR db 0dh, 0ah, '$'
PC db 'PC', '$'
PCXT db 'PC/XT', '$'
;AT db 'AT', '$'
PS230 db 'PS2 model 30', '$'
PS250 db 'PS2 model 50 or 60 or AT', '$'
PS280 db 'PS2 model 80', '$'
PSJR db 'PSjr', '$'
PSC db 'PS Conventible', '$'
```

```
VHEADER db 'Version:', '$'
LT2S db '< 2.0', '$'
VNUMBER db ' ', '$'
OEMHEADER db 'OEM:', '$'
OEMNUMBER db ' ', '$'
SHEADER db 'Serial number:', '$'
SNUMBER db ' ', '$'
```

```
IBMPS dw 7 dup(0) ; 7 types, easy index access
```

```
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 AH, AL
    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
```

```
    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

    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_l
    or AL,30h
    mov [SI],AL
end_l: pop DX
    pop CX
    ret
BYTE_TO_DEC ENDP

BEGIN:
PCMODEL:
    mov CX, 0

    mov DX, offset PS280
    mov [IBMPS + BX], DX
    add BX, 2

    mov DX, offset PSC
    mov [IBMPS + BX], DX
    add BX, 2

    mov DX, offset PS230
    mov [IBMPS + BX], DX
    add BX, 2

    mov DX, offset PCXT
    mov [IBMPS + BX], DX
    add BX, 2

    mov DX, offset PS250
    mov [IBMPS + BX], DX
    add BX, 2

    mov DX, offset PSJR
    mov [IBMPS + BX], DX
    add BX, 2

    mov DX, offset PCXT
    mov [IBMPS + BX], DX
    add BX, 2

    mov DX, offset PC
    mov [IBMPS + BX], DX

```

```

mov BX, 0

mov DX, offset IBMT
mov AH, 09h
int 21h
mov AX, 0F000h
mov ES, AX
mov AL, ES:[0FFFEh]
mov AH, 0
sub AL, 0f8h
shl AL, 1
mov BX, AX

mov DX, [IBMPS + BX]
mov AH, 09h
int 21h
mov dx, offset br
int 21h

VERSION:
mov DX, offset VHEADER
int 21h
mov AH, 30h
int 21h

cmp al, 0
je LT2
mov si, offset VNUMBER
push si
add si, 1
push ax
call BYTE_TO_DEC
pop ax
pop si
push si
add si, 2
mov byte ptr [si], 2eh
pop si
add si, 3
xchg al, ah
call BYTE_TO_DEC
mov ah, 09h
mov dx, offset VNUMBER
int 21h
mov dx, offset br
int 21h
jmp OEM

LT2:
mov DX, offset LT2S
mov AH, 09h
int 21h
mov dx, offset br
int 21h

OEM:
mov dx, offset OEMHEADER
int 21h
mov al, bh
mov si, offset OEMNUMBER
add si, 2
call BYTE_TO_DEC

```

```

mov ah, 09h
mov dx, offset OEMNUMBER
int 21h
mov dx, offset br
int 21h

SERIAL:
mov dx, offset SHEADER
int 21h
mov si, offset SNUMBER
add si, 7
mov ax, cx
call BYTE_TO_DEC
dec si
mov al, bl
call BYTE_TO_DEC
mov dx, offset SNUMBER
mov ah, 09h
int 21h
mov dx, offset br
int 21h

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

```

ПРИЛОЖЕНИЕ Б. ИСХОДНЫЙ КОД .EXE МОДУЛЯ

```
AStack SEGMENT STACK
    DW 100 DUP('*')
AStack ENDS

DATA SEGMENT

IBMT db 'IBM PC TYPE IS: ', '$'
BR db 0dh, 0ah, '$'
PC db 'PC', '$'
PCXT db 'PC/XT', '$'
;AT db 'AT', '$'
PS230 db 'PS2 model 30', '$'
PS250 db 'PS2 model 50 or 60 or AT', '$'
PS280 db 'PS2 model 80', '$'
PSJR db 'PSjr', '$'
PSC db 'PS Conventible', '$'

VHEADER db 'Version:', '$'
LT2S db '< 2.0', '$'
VNUMBER db ' ', '$'
OEMHEADER db 'OEM:', '$'
OEMNUMBER db ' ', '$'
SHEADER db 'Serial number:', '$'
SNUMBER db ' ', '$'

IBMPS dw 7 dup(0) ; 7 types, easy index access

DATA ENDS

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

TETR_TO_HEX PROC near
and AL,0Fh
    cmp AL,09
    jbe NEXT
    add AL,07
NEXT: add AL,30h
    ret
TETR_TO_HEX ENDP

BYTE_TO_HEX PROC near

    push CX
    mov AH,AL
    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

    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

    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_l
    or AL,30h
    mov [SI],AL
end_l: pop DX
    pop CX
    ret
BYTE_TO_DEC ENDP

MAIN PROC FAR
    push ds
    xor ax, ax
    push ax
    mov ax, DATA
    mov ds, ax
PCMODEL:
    mov CX, 0

    mov DX, offset PS280
    mov [IBMPS + BX], DX
    add BX, 2

    mov DX, offset PSC
    mov [IBMPS + BX], DX
    add BX, 2

    mov DX, offset PS230
    mov [IBMPS + BX], DX
    add BX, 2

    mov DX, offset PCXT
    mov [IBMPS + BX], DX
    add BX, 2

    mov DX, offset PS250
    mov [IBMPS + BX], DX
    add BX, 2

```



```

mov DX, offset PSJR
mov [IBMPS + BX], DX
add BX, 2

mov DX, offset PCXT
mov [IBMPS + BX], DX
add BX, 2

mov DX, offset PC
mov [IBMPS + BX], DX
mov BX, 0

mov DX, offset IBMT
mov AH, 09h
int 21h
mov AX, 0F000h
mov ES, AX
mov AL, ES:[0FFFEh]
mov AH, 0
sub AL, 0f8h
shl AL, 1
mov BX, AX

mov DX, [IBMPS + BX]
mov AH, 09h
int 21h
mov dx, offset br
int 21h

VERSION:
mov DX, offset VHEADER
int 21h
mov AH, 30h
int 21h

cmp al, 0
je LT2
mov si, offset VNUMBER
push si
add si, 1
push ax
call BYTE_TO_DEC
pop ax
pop si
push si
add si, 2
mov byte ptr [si], 2eh
pop si
add si, 3
xchg al, ah
call BYTE_TO_DEC
mov ah, 09h
mov dx, offset VNUMBER
int 21h
mov dx, offset br
int 21h
jmp OEM

LT2:
mov DX, offset LT2S
mov AH, 09h
int 21h

```

```

mov dx, offset br
int 21h

OEM:
mov dx, offset OEMHEADER
int 21h
mov al, bh
mov si, offset OEMNUMBER
add si, 2
call BYTE_TO_DEC
mov ah, 09h
mov dx, offset OEMNUMBER
int 21h
mov dx, offset br
int 21h

SERIAL:
mov dx, offset SHEADER
int 21h
mov si, offset SNUMBER
add si, 7
mov ax, cx
call BYTE_TO_DEC
dec si
mov al, bl
call BYTE_TO_DEC
mov dx, offset SNUMBER
mov ah, 09h
int 21h
mov dx, offset br
int 21h

xor al, al
mov ah, 4Ch
int 21H

MAIN ENDP
CODE ENDS
END MAIN

```