

**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**

Студент гр. 9383

\_\_\_\_\_

Звега А.Р.

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.

### **Задание.**

1. Напишите текст исходного .COM модуля, который определяет тип РС и версию системы.

2. Напишите текст исходного .EXE модуля, который выполняет те же функции, что и модуль в Шаге 1 и постройте и отладьте его. Таким образом, будет получен «хороший» .EXE.

3. Сравните исходные тексты для .COM и .EXE модулей. Ответьте на контрольные вопросы «Отличия исходных текстов COM и EXE программ».

4. Запустите FAR и откройте (F3/F4) файл загрузочного модуля .COM и файл «плохого» .EXE в шестнадцатеричном виде. Затем откройте (F3/F4) файл загрузочного модуля «хорошего» .EXE и сравните его с предыдущими файлами. Ответьте на контрольные вопросы «Отличия форматов файлов COM и EXE модулей».

5. Откройте отладчик TD.EXE и загрузите .COM. Ответьте на контрольные вопросы «Загрузка COM модуля в основную память». Представьте в отчете план загрузки модуля .COM в основную память.

6. Откройте отладчик TD.EXE и загрузите «хороший» .EXE. Ответьте на контрольные вопросы «Загрузка «хорошего» EXE модуля в основную память».

7. Оформление отчета в соответствии с требованиями. В отчете необходимо привести скриншоты. Для файлов их вид в шестнадцатеричном виде, для загрузочных модулей – в отладчике.

## Выполнение работы.

**Шаг 1.** На языке ассемблера был написан исходный код модуля COM определяющий тип PC и версию системы(Рисунок 1). Кроме этого был получен «плохой» EXE модуль, полученный из исходного кода для COM модуля(Рисунок 2).

```
C:\USERS\ZIPZAP\DESKTOP\LABOS\MASM>LAB1BIN.COM
IBM PC type: AT
Version: 5.0
OEM: 255
User serial number: 000000
```

Рисунок 1: результат работы COM

```
C:\USERS\ZIPZAP\DESKTOP\LABOS\MASM>LAB1BIN.EXE

                                0sIBM PC type: PC
5 0                                0sIBM PC type: PC
255                                0sIBM PC type: PC
0sIBM PC type: PC000000 PC type: PC/XT
```

Рисунок 2: результат работы „плохого“ EXE

**Шаг 2.** Был написан исходный код EXE модуля, который выполняет те же функции что и COM модуль(Рисунок 3).

```
C:\USERS\ZIPZAP\DESKTOP\LABOS\MASM>LAB1.EXE
IBM PC type: AT
Version: 5.0
OEM: 255
User serial number: 000000
```

Рисунок 3: результат работы „хорошего“ EXE

## Вопросы к шагу 3. «Отличия исходных текстов COM и EXE программ».

1. Сколько сегментов должна содержать COM-программа?

COM-программа ограничена размером одного сегмента, так как данные и код находятся в одном сегменте, стек же устанавливается на последнюю ячейку сегмента.

2. Сколько сегментов должна содержать EXE-программа?

EXE-программа должна содержать один сегмент или больше.

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

Обязательно должна быть директива `org100h`, так как в DOS первые 256 байт занимает PSP. Директивой `ASSUME` нужно связать сегменты данных и кода в один сегмент.

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

Нельзя использовать команды вида `mov <регистр>, seg <имя сегмента>`. Так как в COM-программе нет таблицы настроек, в которой содержится описание адресов, зависящих от размещения загрузочного модуля в оперативной памяти.

**Шаг 4.** Было выполнено сравнение файлов COM и EXE в шестнадцатеричном виде (см. Рисунки 4,5,6).

00000000	E9	73	01	49	42	40	20	50	43	20	74	79	70	65	3A	20	8.s.IBM PC type:
00000010	50	43	00	0A	24	49	42	40	20	50	43	20	74	79	70	65	PC.s.IBM PC type
00000020	3A	20	50	43	2F	58	54	00	0A	24	49	42	40	20	50	43	: PC/XT.s.IBM PC
00000030	20	74	79	70	65	3A	20	51	4A	00	0A	24	49	42	40	20	type: AT.s.IBM
00000040	50	43	20	74	79	70	65	3A	20	50	53	32	20	6D	6F	64	PC type: PS2 mod
00000050	65	6C	20	33	30	00	0A	24	49	42	40	20	50	43	20	74	el 30.s.IBM PC t
00000060	70	70	65	3A	20	50	53	32	20	6D	6F	64	65	6C	20	33	type: PS2 model
00000070	30	20	6F	72	20	35	30	00	0A	24	49	42	40	20	50	43	0 or 50.s.IBM PC
00000080	20	74	79	70	65	3A	20	50	53	32	20	6D	6F	64	65	6C	type: PS2 model
00000090	20	38	30	00	0A	24	49	42	40	20	50	43	20	74	79	70	80.s.IBM PC typ
000000A0	65	3A	20	50	43	6A	72	00	0A	24	49	42	40	20	50	43	e: PCjr.s.IBM PC
000000B0	20	74	79	70	65	3A	20	50	43	20	43	6F	66	76	65	72	type: PC Conver
000000C0	74	69	62	6C	65	00	0A	24	49	42	40	20	50	43	20	74	table.s.IBM PC t
000000D0	70	70	65	3A	20	20	20	20	0D	0A	24	56	65	72	73	69	type: .\$.Versi
000000E0	6F	6E	3A	20	20	2E	20	00	0A	24	4F	45	40	3A	20	20	on: .\$.OEM:
000000F0	20	20	20	20	20	20	20	20	20	20	20	0D	0A	24	55	73	\$.\$.Us
00001000	65	72	20	73	65	72	69	61	6C	20	6F	75	6D	62	65	72	er serial number
00001010	3A	20	20	20	20	20	20	20	0A	24	50	53	51	52	32	32	\$.\$.SPSRQR
00001020	E4	33	02	D2	B9	0A	00	F7	F1	80	00	3A	88	14	4E	33	D2
00001030	30	0A	00	73	F1	3C	00	74	04	0C	30	88	5A	59	58	58	\$.\$.sz.c: .00.YZ
00001040	58	33	24	0F	3C	00	79	62	04	07	30	30	C3	51	5A	58	X\$.\$.<v...00.Q0
00001050	E8	EF	FF	86	C4	B1	04	02	E8	E8	6E	FF	59	C3	53	8A	on 43\$.\$.y0p.Y\$.S
00001060	FC	E8	FF	8F	88	25	4F	88	05	4F	8A	C7	E8	DE	FF	88	000.E00.00.Y\$.S
00001070	25	4F	88	05	58	C3	33	C0	B8	00	0F	8E	C0	2A	0A	FE	%00.[.34.=A00.
00001080	FF	3C	FF	74	22	3C	FE	74	23	34	FB	74	20	3C	FC	74	< t%.<\$.<0t <t
00001090	22	3C	FA	74	2C	3C	FC	74	26	3C	F8	74	28	3C	FD	74	*<\$.t.u0].<0E%]
000010A0	2A	3C	F9	74	2C	75	30	BA	01	E8	36	90	BA	15	01	74	\$.<t.u0].<0E%]
000010B0	E8	30	90	BA	2A	01	EB	2A	90	BA	3C	01	EB	2A	90	BA	00E%].<0E%]<0E\$
000010C0	58	01	EB	1E	90	BA	7A	01	EB	18	90	BA	96	01</			

### Рисунок 4: COM

```

00000000 4D 5A 31 01 03 00 00 00 20 00 00 00 FF FF 00 MZ1.....
00000010 00 00 FD 16 00 01 00 00 1E 00 00 00 01 00 00 00
00000300 E9 73 01 49 42 4D 20 50 43 20 74 79 65 3A 20 6S.IBM PC type:
00000310 50 43 00 0A 24 49 42 4D 20 50 43 20 74 79 70 65 PC, $IBM PC type
00000320 3A 20 50 43 2F 58 50 0A 24 49 42 4D 20 50 43 PC/XT, $IBM PC
00000330 20 74 79 70 65 3A 20 41 5A 00 0A 24 49 42 4D 20 type: AT, $IBM
00000340 50 43 20 74 79 65 3A 20 50 53 32 20 6D 6F 6A PC type: PS2 mod
00000350 65 6C 20 33 30 00 0A 24 49 42 4D 20 50 43 20 el 30, $IBM PC t
00000360 79 70 65 3A 20 50 53 32 00 6F 64 65 6C 20 33 ype: PS2 model 3
00000370 30 20 6F 72 20 35 30 00 0A 24 49 42 4D 20 50 43 0 or 50, $IBM PC
00000380 20 74 79 65 3A 20 50 53 20 6D 6F 64 65 6C type: PS2 model
00000390 20 38 30 00 0A 24 49 42 4D 20 50 43 20 74 79 70 80, $IBM PC typ
000003A0 65 3A 20 50 43 6A 72 00 0A 24 49 42 4D 20 50 43 e: PCjr, $IBM PC
000003B0 20 74 79 70 65 3A 20 50 43 20 43 6F 6E 76 65 72 type: PC Conver
000003C0 74 69 62 6C 65 00 0A 24 49 42 4D 20 50 43 20 74 table, $IBM PC b
000003D0 79 70 65 3A 20 20 20 20 0A 24 56 65 72 73 69 ype: $Versi
000003E0 6F 6E 3A 20 20 2E 20 00 0A 24 4F 45 4D 7A 20 on: ..$OEM:
000003F0 20 20 20 20 20 20 20 20 20 20 20 00 0A 24 55 73 ..$US
00000400 65 72 20 73 65 72 69 61 6C 20 6E 75 6D 62 65 72 er serial number
00000410 : ..$PSQR2
00000420 3A 20 20 20 20 20 20 20 20 00 0A 24 50 53 51 52 32
00000430 E4 33 D2 B9 0A 00 F7 F1 80 CA 30 88 51 4E 33 D2
00000440 30 0A 00 73 F1 3C 00 74 04 6C 30 88 04 5A 59 58
00000450 58 C3 24 0F 89 70 62 04 07 64 30 C3 51 8A E0
00000460 FC E8 FF 86 C4 B1 04 D2 E8 E6 FF 59 C3 53 8A
00000470 25 FF 88 95 58 C3 33 C8 80 09 7E 0C 26 A0 FE
00000480 3F 3C FF 74 22 3C FE 74 2A 3C FB 2A 3C FC 74
00000490 22 3C FA 2C 3C FC 74 26 3C FB 2A 3C FD 74
000004A0 2A 3C F9 74 2C 75 30 BA 01 E8 36 90 BA 15 01
000004B0 E8 30 90 BA 2A 01 E8 2A 90 BA 3C 01 E8 2A 90 BA
000004C0 50 01 E8 1E 90 BA 7A 01 E8 18 90 BA 96 01 E8 12
000004D0 90 BA AA 01 E8 0C 90 E8 73 FF BF 05 01 89 05 BA
000004E0 C8 01 BA 09 CD 21 33 D8 33 C8 B4 30 21 BE EA
000004F0 01 E8 27 FF 8A CA BE 01 E8 1F FF BA DB 01 B4
00000500 09 CD 21 BA C7 BE F1 01 E8 1F FF BA EA 01 B4 09
00000510 02 21 8A C3 E8 36 FF BF 12 02 89 05 BA 01 BF 17
00000520 02 E8 3A FF BA FE 01 B4 09 CD 21 32 C0 B4 4C CD
00000530 21

```

### Рисунок 5: "плохой" EXE

```

00000000 4D 5A 3E 01 03 00 01 00 20 00 00 00 FF FF 00 00 MZ;.....
00000010 00 01 3C 03 5B 00 22 00 1E 00 00 00 01 00 5C 00 .E[.".....\
00000020 22 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 .....
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 .....
00000200 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 .....
00000220 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000230 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 .....
00000250 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 .....
00000270 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 .....
00000290 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 .....
000002B0 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 .....
000002D0 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 .....
000002F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000300 49 42 4D 20 50 43 20 74 79 70 65 3A 20 50 43 0D IBM PC type: PC.
00000310 0A 24 49 42 4D 20 50 43 20 74 79 70 65 3A 20 50 .IBM PC type: P
00000320 43 2F 58 54 0D 0A 24 49 42 4D 20 50 43 20 74 79 C/XT..IBM PC ty
00000330 70 65 3A 20 41 54 0D 0A 24 49 42 4D 20 50 43 20 pe: AT..IBM PC
00000340 74 79 70 65 3A 20 50 53 32 20 6D 6F 64 65 6C 20 type: PS2 model
00000350 33 30 0D 0A 24 49 42 4D 20 50 43 20 74 79 70 65 30..IBM PC type
00000360 3A 20 50 53 32 20 6D 6F 64 65 6C 20 33 30 20 6F : PS2 model 30 o
00000370 72 20 35 30 0D 0A 24 49 42 4D 20 50 43 20 74 79 r 50..IBM PC ty
00000380 70 65 3A 20 50 53 32 20 6D 6F 64 65 6C 20 38 30 pe: PS2 model 80
00000390 0D 0A 24 49 42 4D 20 50 43 20 74 79 70 65 3A 20 ..IBM PC type:
000003A0 50 43 6A 72 0D 0A 24 49 42 4D 20 50 43 20 74 79 PCjr..IBM PC ty
000003B0 70 65 3A 20 50 43 20 43 6F 6E 76 65 72 74 69 62 pe: PC Convertib
000003C0 6C 65 0D 0A 24 49 42 4D 20 50 43 20 74 79 70 65 le..IBM PC type
000003D0 3A 20 20 20 20 0D 0A 24 56 65 72 73 69 6F 6E 3A : ..$Version:
000003E0 20 20 2E 20 0D 0A 24 4F 45 4D 3A 20 20 20 20 20 ..$OEM:
000003F0 20 20 20 20 20 20 20 20 0D 0A 24 55 73 65 72 20 ..$User
00000400 73 65 72 69 61 6C 20 6E 75 6D 62 65 72 3A 20 20 serial number:
00000410 20 20 20 20 20 0D 0A 24 00 00 00 00 00 00 00 00 ..$.
00000420 50 53 51 52 32 E4 33 D2 B9 0A 00 F7 F1 80 CA 30 PSQR2E3...sC40
00000430 88 14 4E 33 D2 3D 0A 00 73 F1 3C 00 74 04 0C 30 8.N3...s<.t..o
00000440 88 04 5A 59 58 58 C3 24 0F 3C 09 76 02 04 07 04 8.zv[X]$.<.v....
00000450 30 C3 51 8A E0 E8 EF FF 86 C4 B1 04 D2 E8 E8 E6 0|Q&4n à-...T&4μ

```

#### Вопросы к шагу 4. «Отличия форматов файлов COM и EXE модулей».

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

Файл типа COM содержит команды и данные. Код располагается с адреса 0h.

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

Данные и код находятся в одном сегменте. Код расположен по адресу 300h. По адресу 0h располагается управляющая информация, заголовок и таблица настроек.

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

Файл состоит из трех сегментов: стек, данные и код. У файла неограниченный размер. С 0h адреса расположен заголовок, после него идет таблица настроек. У «хорошего» EXE выделяется память под стек между PSP и кодом. PSP - 256 байтный блок, который содержится как в exe-файле, так и в com-файле. Различия: разделение сегментов и размер смещения (300h и 400h), так как в com-файлах отсутствует блок начальной загрузки и следовательно он занимает меньше места, чем exe-файл.

#### **Вопросы к шагу 5. «Загрузка COM модуля в основную память».**

1) Какой формат загрузки модуля COM? С какого адреса располагается код? Сначала загружается PSP, затем код. Код располагается после PSP с адреса 100h.

2) Что располагается с адреса 0?

Располагается PSP(0h-100h).

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

При загрузке сегментные регистры указывают на начало PSP.

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

В COM модуле стек создается автоматически и занимает всю область сегмента. Адреса расположены в диапазоне 0h-ffffh.

#### **Вопросы к шагу 6. «Загрузка «хорошего» EXE модуля в основную память».**

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

Также как и COM. Сегментные регистры DS и ES устанавливаются на начало PSP, SS - на начало сегмента стека, CS — на начало сегмента команд.

2) На что указывают регистры DS и ES?

Сегментные регистры DS и ES указывают на начало PSP

3) Как определяется стек?

С помощью регистров SS и SP.

SS указывает на начало, а SS:SP на конец.

4) Как определяется точка входа?

Точка входа определяется директивой END + <метка или процедура>.

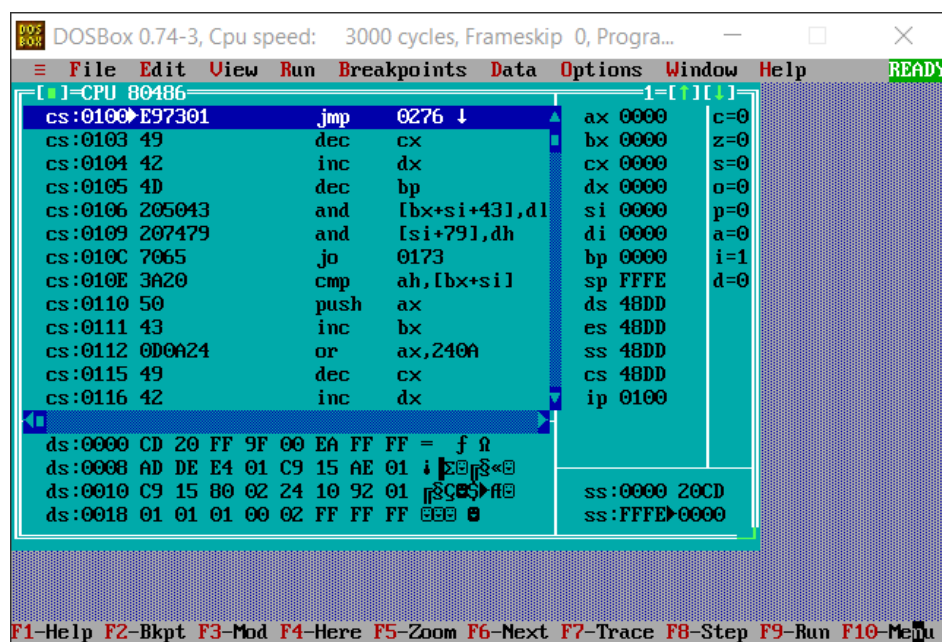


Рисунок 7: td lab1bin.com

## Выводы.

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



## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММ

Название файла: lab1bin.asm

TESTPC SEGMENT

ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING  
ORG 100H

START: JMP BEGIN

pc db 'IBM PC type: PC',0dh,0ah,'\$'  
pc\_xt db 'IBM PC type: PC/XT',0dh,0ah,'\$'  
at db 'IBM PC type: AT',0dh,0ah,'\$'  
ps2\_30 db 'IBM PC type: PS2 model 30',0dh,0ah,'\$'  
ps2\_50\_60 db 'IBM PC type: PS2 model 30 or 50',0dh,0ah,'\$'  
ps2\_80 db 'IBM PC type: PS2 model 80',0dh,0ah,'\$'  
pcjr db 'IBM PC type: PCjr',0dh,0ah,'\$'  
pc\_convertible db 'IBM PC type: PC Convertible',0dh,0ah,'\$'  
unknown db 'IBM PC type: ',0dh,0ah,'\$'  
version db 'Version: . ',0dh,0ah,'\$'  
oem db 'OEM: ',0dh,0ah,'\$'  
user\_serial\_number db 'User serial number: ',0dh,0ah,'\$'

BYTE\_TO\_DEC PROC near

PUSH AX

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
    POP AX
RET
BYTE_TO_DEC ENDP

```

```

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_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
```

```
BEGIN:
    XOR ax,ax
    MOV ax,0f000h
    MOV es,ax
    MOV al,es:[0fffeh]

    CMP al,0ffh
    JE Label_pc
    CMP al,0feh
    JE Label_pc_xt
```

```
CMP al,0fbh
JE Label_pc_xt
CMP al,0fch
JE Label_at
CMP al,0fah
JE Label_ps2_30
CMP al,0fch
JE Label_ps2_50_60
CMP al,0f8h
JE Label_ps2_80
CMP al,0fdh
JE Label_pcjr
CMP al,0f9h
JE Label_pc_convertible
JNE label_unknown
```

label\_pc:

```
MOV dx,offset pc
JMP print_version
```

label\_pc\_xt:

```
MOV dx,offset pc_xt
JMP print_version
```

label\_at:

```
MOV dx,offset at
JMP print_version
```

label\_ps2\_30:

```
MOV dx,offset ps2_30
JMP print_version
```

label\_ps2\_50\_60:

```
MOV dx,offset ps2_50_60
JMP print_version
```

label\_ps2\_80:

```
MOV dx,offset ps2_80
JMP print_version
```

```
label_pcjr:
    MOV dx,offset pcjr
    JMP print_version
label_PC_convertible:
    MOV dx,offset pc_convertible
    JMP print_version
label_unknown:
    CALL BYTE_TO_HEX
    MOV di, offset unknown+13
    MOV [di], ax
    MOV dx,offset unknown
```

```
print_version:
    MOV ah,09h
    INT 21h

    XOR bx, bx
    XOR ax, ax
    MOV ah, 30h
    INT 21h

    MOV si, offset version+9
    CALL BYTE_TO_DEC

    MOV al, ah
    MOV si, offset version+11
    CALL BYTE_TO_DEC

    MOV dx, offset version
    MOV ah,09h
    INT 21h

    MOV al, bh
```

```
MOV si, offset oem+7  
CALL BYTE_TO_DEC
```

```
MOV dx, offset oem  
MOV ah,09h  
INT 21h
```

```
MOV al, bl  
CALL BYTE_TO_HEX
```

```
MOV di, offset user_serial_number+20  
MOV [di], ax
```

```
MOV ax, cx  
MOV di, offset user_serial_number+25  
CALL WRD_TO_HEX
```

```
MOV dx, offset user_serial_number  
MOV ah,09h  
INT 21h
```

```
exit:  
    XOR al,al  
    MOV ah,4ch  
    INT 21h  
TESTPC ENDS  
END START
```

```
Название файла: lab1.asm  
AStack  SEGMENT STACK  
    DW 128 DUP(?)  
AStack  ENDS
```

## DATA SEGMENT

```
pc db 'IBM PC type: PC',0dh,0ah,'$'
pc_xt db 'IBM PC type: PC/XT',0dh,0ah,'$'
at db 'IBM PC type: AT',0dh,0ah,'$'
ps2_30 db 'IBM PC type: PS2 model 30',0dh,0ah,'$'
ps2_50_60 db 'IBM PC type: PS2 model 30 or 50',0dh,0ah,'$'
ps2_80 db 'IBM PC type: PS2 model 80',0dh,0ah,'$'
pcjr db 'IBM PC type: PCjr',0dh,0ah,'$'
pc_convertible db 'IBM PC type: PC Convertible',0dh,0ah,'$'
unknown db 'IBM PC type:  ',0dh,0ah,'$'
version db 'Version:  . ',0dh,0ah,'$'
oem db 'OEM:          ',0dh,0ah,'$'
user_serial_number db 'User serial number:    ',0dh,0ah,'$'
```

## DATA ENDS

## CODE SEGMENT

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

## BYTE\_TO\_DEC PROC near

```
    PUSH AX
    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
    POP AX
    RET
BYTE_TO_DEC ENDP

```

```

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

```

```

Main PROC FAR
    MOV ax, DATA
    MOV ds, ax

    XOR ax,ax
    MOV ax,0f000h
    MOV es,ax
    MOV al,es:[0fffeh]

    CMP al,0ffh
    JE label_pc
    CMP al,0feh
    JE label_pc_xt
    CMP al,0fbh

```

```
JE label_pc_xt
CMP al,0fch
JE label_at
CMP al,0fah
JE label_ps2_30
CMP al,0fch
JE label_ps2_50_60
CMP al,0f8h
JE label_ps2_80
CMP al,0fdh
JE label_pcjr
CMP al,0f9h
JE label_pc_convertible
JNE label_unknown
```

```
label_pc:
    MOV dx,offset pc
    JMP print_ibm_pc_version
label_pc_xt:
    MOV dx,offset pc_xt
    JMP print_ibm_pc_version
label_at:
    MOV dx,offset at
    JMP print_ibm_pc_version
label_ps2_30:
    MOV dx,offset ps2_30
    JMP print_ibm_pc_version
label_ps2_50_60:
    MOV dx,offset ps2_50_60
    JMP print_ibm_pc_version
label_ps2_80:
    MOV dx,offset ps2_80
    JMP print_ibm_pc_version
label_pcjr:
```

```
MOV dx,offset pcjr
JMP print_ibm_pc_version
label_pc_convertible:
MOV dx,offset pc_convertible
JMP print_ibm_pc_version
label_unknown:
CALL BYTE_TO_HEX
MOV di, offset unknown+13
MOV [di], ax
MOV dx,offset unknown
```

```
print_ibm_pc_version:
```

```
MOV ah,09h
INT 21h
```

```
XOR bx, bx
XOR ax, ax
MOV ah, 30h
INT 21h
```

```
MOV si, offset version+9
CALL BYTE_TO_DEC
```

```
MOV al, ah
MOV si, offset version+11
CALL BYTE_TO_DEC
```

```
MOV dx, offset version
MOV ah,09h
INT 21h
```

```
MOV al, bh
MOV si, offset oem+7
```

CALL BYTE\_TO\_DEC

MOV dx, offset oem

MOV ah,09h

INT 21h

MOV al, bl

CALL BYTE\_TO\_HEX

MOV di, offset user\_serial\_number+20

MOV [di], ax

MOV ax, cx

MOV di, offset user\_serial\_number+25

CALL WRD\_TO\_HEX

MOV dx, offset user\_serial\_number

MOV ah,09h

INT 21h

exit:

XOR al,al

MOV ah,4ch

INT 21h

Main ENDP

CODE ENDS

END Main