

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

Ноздрин В.Я.

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.

Написать тексты исходных .COM и .EXE модулей, которые выводят информацию о системе.

Основные теоретические положения.

Тип IMB PC хранится в байте по адресу 0F000:0FFFEh, в предпоследнем байте ROM BIOS. Ниже приведены коды и типы IMB PC.

1. Тип PC – код FF
2. Тип PC/XT – код FE или FB
3. Тип AT – код FC
4. Тип PS2 модель 30 – код FA
5. Тип PS2 модель 50 или 60 – код FC
6. Тип PS2 модель 80 – код F8
7. Тип PCjr – код FD
8. Тип PC Convertible – код 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 – 21-битовый серийный номер пользователя

Выполнение работы.

Шаг 1. Написан текст исходного .COM модуля lab1_com.asm. С помощью компилятора MASM.EXE и компоновщика LINK.EXE был получен «плохой» .EXE модуль. Далее с помощью исполняемого файла BIN2EXE.EXE из «плохого» .EXE модуля был получен «хороший» .COM модуль lab1_com.com.

```
C:\>LAB1_COM.EXE

          5 0
          0
7This PC type 000000
                                07This PC type is PC
                                07This PC type is PC
```

Рис. 1 – Пример работы «плохого» .EXE модуля

```
C:\>LAB1_COM.COM
This PC type is AT
DOS Version: 5.0
Serial number OEM: 0
Serial number: 000000
```

Рис. 2 – Пример работы «хорошего» .COM модуля

Шаг 2. Написан текст исходного .EXE модуля lab1_exe.asm. С помощью компилятора MASM.EXE и компоновщика LINK.EXE был получен «хороший» .EXE модуль.

```
C:\>LAB1_EXE.EXE
This PC type is AT
DOS Version: 5.0
Serial number OEM: 0
Serial number: 000000
```

Рис. 3 – Пример работы «хорошего» .EXE модуля

Используемые функции

TETR_TO_HEX – функция, переводящая десятичную цифру в код символа.

BYTE_TO_HEX – функция, переводящая байт в шестнадцатеричной системе счисления в код символа.

WRD_TO_HEX – функция, переводящая шестнадцатеричное число в символьный код.

BYTE_TO_DEC – функция, переводящая байт в шестнадцатеричной системе счисления в символьный код десятичной системы счисления.

PRINT_STRING – функция, выводящая строку на экран

PRINT_PC_TYPE – функция, выводящая на экран тип IBM PC

PRINT_OS_VERSION – функция, выводящая на экран версию MS DOS и серийный номер.

Контрольные вопросы.

Отличия исходных текстов COM и EXE программ

1) Сколько сегментов должна содержать COM-программа?

COM-программа должна содержать один сегмент. В нем хранятся данные и код, а стек устанавливается автоматически на последнюю ячейку сегмента.

2) Сколько сегментов должна содержать EXE-программа?

EXE-программа может состоять как из одного так и из большего числа сегментов. Это зависит от выбранной модели памяти.

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

В тексте COM-программы обязательно должна быть директива ORG 100h, резервирующая 256 байт (100h в шестнадцатеричной системе счисления) памяти под разного рода управляющие структуры. Также должна быть директива ASSUME, сопоставляющая сегментные регистры и программные сегменты. И директива END для завершения программы.

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

В COM-программе нельзя использовать команды типа SEG NAME, где NAME – название сегмента. В COM-программах нет таблицы настройки, с помощью которой в момент запуска программы загрузчик определяет и подставляет адреса сегментов. Также нельзя использовать процедуры FAR, так как в COM-программах только один сегмент.

Отличия форматов файлов COM и EXE модулей

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

COM файл содержит код, данные и стек в одном сегменте. Код располагается с адреса 0 (см рис. 4).

```

ice-jack@ice-pc ~/P/s/n/lab1 (nozdrin_lab1)> hexdump -C LAB1_COM.COM
00000000 e9 37 02 54 68 69 73 20 50 43 20 74 79 70 65 20 |.7.This PC type |
00000010 69 73 20 50 43 0d 0a 24 54 68 69 73 20 50 43 20 |is PC..$This PC |
00000020 74 79 70 65 20 69 73 20 50 43 2f 58 54 0d 0a 24 |type is PC/XT..$ |
00000030 54 68 69 73 20 50 43 20 74 79 70 65 20 69 73 20 |This PC type is |
00000040 41 54 0d 0a 24 54 68 69 73 20 50 43 20 74 79 70 |AT..$This PC typ |
00000050 65 20 69 73 20 50 53 32 20 6d 33 30 0d 0a 24 54 |e is PS2 m30..$T |
00000060 68 69 73 20 50 43 20 74 79 70 65 20 69 73 20 50 |his PC type is P |
00000070 53 32 20 6d 35 30 2f 36 30 0d 0a 24 54 68 69 73 |S2 m50/60..$This |
00000080 20 50 43 20 74 79 70 65 20 69 73 20 50 53 32 20 | PC type is PS2 |
00000090 6d 38 30 3a 20 0d 0a 24 54 68 69 73 20 50 43 20 |m80: ..$This PC |
000000a0 74 79 70 65 20 69 73 20 50 d0 a1 6a 72 0d 0a 24 |type is P..jr..$ |
000000b0 54 68 69 73 20 50 43 20 74 79 70 65 20 69 73 20 |This PC type is |
000000c0 50 43 20 43 6f 6e 76 65 72 74 69 62 6c 65 0d 0a |PC Convertible.. |
000000d0 24 44 4f 53 20 56 65 72 73 69 6f 6e 3a 20 20 2e |$DOS Version:  . |
000000e0 20 20 0d 0a 24 53 65 72 69 61 6c 20 6e 75 6d 62 | ..$Serial numb |
000000f0 65 72 20 4f 45 4d 3a 20 20 20 20 20 20 0d 0a |er OEM:  .. |
00000100 24 53 65 72 69 61 6c 20 6e 75 6d 62 65 72 3a 20 |$Serial number: |
00000110 20 20 20 20 20 20 0d 0a 24 45 52 52 4f 52 0d 0a | ..$ERROR.. |
00000120 24 24 0f 3c 09 76 02 04 07 04 30 c3 51 8a e0 e8 |$$.<.v....0.Q... |
00000130 ef ff 86 c4 b1 04 d2 e8 e8 e6 ff 59 c3 53 8a fc |.....Y.S.. |
00000140 e8 e9 ff 88 25 4f 88 05 4f 8a c7 e8 de ff 88 25 |....%O..O.....% |
00000150 4f 88 05 5b c3 51 52 32 e4 33 d2 b9 0a 00 f7 f1 |O..[.QR2.3..... |
00000160 80 ca 30 88 14 4e 33 d2 3d 0a 00 73 f1 3c 00 74 |..0..N3.=..s.<.t |
00000170 04 0c 30 88 04 5a 59 c3 50 b4 09 cd 21 58 c3 b8 |..0..ZY.P...!X.. |
00000180 00 f0 8e c0 26 a0 fe ff 3c ff 74 26 3c fe 74 28 |....&...<.t&<.t( |
00000190 3c fb 74 24 3c fc 74 26 3c fa 74 28 3c fc 74 2a |<.t$<.t&<.t(<.t* |
000001a0 3c f8 74 2c 3c fd 74 2e 3c f9 74 30 ba 19 02 eb |<.t,<.t.<.t0.... |
000001b0 31 90 ba 03 01 eb 2b 90 ba 18 01 eb 25 90 ba 30 |1.....+.....%..0 |
000001c0 01 eb 1f 90 ba 45 01 eb 19 90 ba 5f 01 eb 13 90 |.....E....._.... |
000001d0 ba 7c 01 eb 0d 90 ba 98 01 eb 07 90 ba b0 01 eb |.|..... |
000001e0 01 90 e8 93 ff c3 50 53 51 52 56 57 b4 30 cd 21 |.....PSQRVW.0.! |
000001f0 be d1 01 83 c6 0d e8 5c ff 8a c4 83 c6 03 e8 54 |.....\.....T |
00000200 ff ba d1 01 e8 71 ff be e5 01 83 c6 13 8a c7 e8 |.....q..... |
00000210 43 ff ba e5 01 e8 60 ff bf 01 02 83 c7 14 8b c1 |C.....`..... |
00000220 e8 1a ff 8a c3 e8 04 ff 83 ef 02 89 05 ba 01 02 |..... |
00000230 e8 45 ff 5a 59 5b 58 5e 5f c3 e8 42 ff e8 a6 ff |.E.ZY[X^_..B.... |
00000240 32 c0 b4 4c cd 21 |2..L.!|
00000246

```

Рис. 4 – шестнадцатеричное представление файла lab1_com.com

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

«Плохой» EXE модуль содержит код, данные и стек в одном сегменте. Код располагается с вдреса 300h (см рис. 5). С адреса 0 располагается управляющая информация загрузчика, заголовок и таблица настроек (см рис. 5).

```

ice-jack@ice-pc ~/P/s/n/lab1 (nozdrin_lab1)> hexdump -C LAB1_COM.EXE
00000000  4d 5a 46 01 03 00 00 00 20 00 00 00 ff ff 00 00 |MZFE.....|
00000010  00 00 cf 15 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 37 02 54 68 69 73 20 50 43 20 74 79 70 65 20 |.7.This PC type|
00000310  69 73 20 50 43 0d 0a 24 54 68 69 73 20 50 43 20 |is PC..$This PC|
00000320  74 79 70 65 20 69 73 20 50 43 2f 58 54 0d 0a 24 |type is PC/XT..$|
00000330  54 68 69 73 20 50 43 20 74 79 70 65 20 69 73 20 |This PC type is|
00000340  41 54 0d 0a 24 54 68 69 73 20 50 43 20 74 79 70 |AT..$This PC typ|
00000350  65 20 69 73 20 50 53 32 20 6d 33 30 0d 0a 24 54 |e is PS2 m30..$T|
00000360  68 69 73 20 50 43 20 74 79 70 65 20 69 73 20 50 |his PC type is P|
00000370  53 32 20 6d 35 30 2f 36 30 0d 0a 24 54 68 69 73 |S2 m50/60..$This|
00000380  20 50 43 20 74 79 70 65 20 69 73 20 50 53 32 20 |PC type is PS2|
00000390  6d 38 30 3a 20 0d 0a 24 54 68 69 73 20 50 43 20 |m80: ..$This PC|
000003a0  74 79 70 65 20 69 73 20 50 d0 a1 6a 72 0d 0a 24 |type is P..jr..$|
000003b0  54 68 69 73 20 50 43 20 74 79 70 65 20 69 73 20 |This PC type is|
000003c0  50 43 20 43 6f 6e 76 65 72 74 69 62 6c 65 0d 0a |PC Convertible..|
000003d0  24 44 4f 53 20 56 65 72 73 69 6f 6e 3a 20 20 2e |$DOS Version:  .|
000003e0  20 20 0d 0a 24 53 65 72 69 61 6c 20 6e 75 6d 62 |..$Serial numb|
000003f0  65 72 20 4f 45 4d 3a 20 20 20 20 20 20 0d 0a |er OEM:  ..|
00000400  24 53 65 72 69 61 6c 20 6e 75 6d 62 65 72 3a 20 |$Serial number:|
00000410  20 20 20 20 20 20 0d 0a 24 45 52 52 4f 52 0d 0a |..$ERROR..|
00000420  24 24 0f 3c 09 76 02 04 07 04 30 c351 8a e0 e8 |$.<.v....0.Q...|
00000430  ef ff 86 c4 b1 04 d2 e8 e8 e6 ff 59 c3 53 8a fc |.....Y.S..|
00000440  e8 e9 ff 88 25 4f 88 05 4f 8a c7 e8 de ff 88 25 |....%0..0.....%|
00000450  4f 88 05 5b c3 51 52 32 e4 33 d2 b9 0a 00 f7 f1 |0..[.QR2.3.....|
00000460  80 ca 30 88 14 4e 33 d2 3d 0a 00 73 f1 3c 00 74 |..0..N3.=..s.<.t|
00000470  04 0c 30 88 04 5a 59 c3 50 b4 09 cd 21 58 c3 b8 |..0..ZY.P...!X..|
00000480  00 f0 8e c0 26 a0 fe ff 3c ff 74 26 3c fe 74 28 |....&...<.t&<.t(|
00000490  3c fb 74 24 3c fc 74 26 3c fa 74 28 3c fc 74 2a |<.t$<.t&<.t(<.t*|
000004a0  3c f8 74 2c 3c fd 74 2e 3c f9 74 30 ba 19 02 eb |<.t,<.t.<.t0....|
000004b0  31 90 ba 03 01 eb 2b 90 ba 18 01 eb 25 90 ba 30 |1.....+.....%..0|
000004c0  01 eb 1f 90 ba 45 01 eb 19 90 ba 5f 01 eb 13 90 |.....E....._....|
000004d0  ba 7c 01 eb 0d 90 ba 98 01 eb 07 90 ba b0 01 eb |.|. ....|
000004e0  01 90 e8 93 ff c3 50 53 51 52 56 57 b4 30 cd 21 |.....PSQRVW.0.!|
000004f0  be d1 01 83 c6 0d e8 5c ff 8a c4 83 c6 03 e8 54 |.....\.....T|
00000500  ff ba d1 01 e8 71 ff be e5 01 83 c6 13 8a c7 e8 |.....q.....|
00000510  43 ff ba e5 01 e8 60 ff bf 01 02 83 c7 14 8b c1 |C.....`.....|
00000520  e8 1a ff 8a c3 e8 04 ff 83 ef 02 89 05 ba 01 02 |.....|
00000530  e8 45 ff 5a 59 5b 58 5e 5f c3 e8 42 ff e8 a6 ff |.E.ZY[X^_..B....|
00000540  32 c0 b4 4c cd 21 |2..L.!|
00000546

```

Рис. 5 – шестнадцатеричное представление файла lab1_com.exe

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

«Хороший» EXE файл содержит код, данные и стек в отдельных сегментах. Это отличает его от «плохого» EXE модуля в котором они хранятся в одном сегменте. В «хорошем» EXE файле код начинается с адреса 400h после заголовка с таблицей настроек (200h) и стека (200h) (см. рис.6). PSP на диске не хранится, память для него выделяется при запуске исполняемого файла.

```

ice-jack@ice-pc ~/P/s/n/lab1 (nozdrin_lab1)> hexdump -C LAB1_EXE.EXE
00000000  4d 5a 4a 01 03 00 01 00  20 00 00 00 ff ff 00 00  |MZJ.....|
00000010  00 01 69 2a 19 01 22 00  1e 00 00 00 01 00 1a 01  |...i*..."|
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  |.....|
*
00000300  54 68 69 73 20 50 43 20  74 79 70 65 20 69 73 20  |This PC type is|
00000310  50 43 0d 0a 24 54 68 69  73 20 50 43 20 74 79 70  |PC..$This PC typ|
00000320  65 20 69 73 20 50 43 2f  58 54 0d 0a 24 54 68 69  |e is PC/XT..$Thi|
00000330  73 20 50 43 20 74 79 70  65 20 69 73 20 41 54 0d  |s PC type is AT.|
00000340  0a 24 54 68 69 73 20 50  43 20 74 79 70 65 20 69  |.$This PC type i|
00000350  73 20 50 53 32 20 6d 33  30 0d 0a 24 54 68 69 73  |s PS2 m30..$This|
00000360  20 50 43 20 74 79 70 65  20 69 73 20 50 53 32 20  |PC type is PS2|
00000370  6d 35 30 2f 36 30 0d 0a  24 54 68 69 73 20 50 43  |m50/60..$This PC|
00000380  20 74 79 70 65 20 69 73  20 50 53 32 20 6d 38 30  |type is PS2 m80|
00000390  3a 20 0d 0a 24 54 68 69  73 20 50 43 20 74 79 70  |: ..$This PC typ|
000003a0  65 20 69 73 20 50 d0 a1  6a 72 0d 0a 24 54 68 69  |e is P..jr..$Thi|
000003b0  73 20 50 43 20 74 79 70  65 20 69 73 20 50 43 20  |s PC type is PC|
000003c0  43 6f 6e 76 65 72 74 69  62 6c 65 0d 0a 24 44 4f  |Convertible..$DO|
000003d0  53 20 56 65 72 73 69 6f  6e 3a 20 20 2e 20 20 0d  |S Version: . . .|
000003e0  0a 24 53 65 72 69 61 6c  20 6e 75 6d 62 65 72 20  |.$Serial number|
000003f0  4f 45 4d 3a 20 20 20 20  20 20 20 0d 0a 24 53 65  |OEM: ..$Se|
00000400  72 69 61 6c 20 6e 75 6d  62 65 72 3a 20 20 20 20  |rial number:|
00000410  20 20 20 0d 0a 24 45 52  52 4f 52 0d 0a 24 00 00  |..$ERROR..$..|
00000420  24 0f 3c 09 76 02 04 07  04 30 c3 51 8a e0 e8 ef  |$.<.v....0.Q....|
00000430  ff 86 c4 b1 04 d2 e8 e8  e6 ff 59 c3 53 8a fc e8  |.....Y.S....|
00000440  e9 ff 88 25 4f 88 05 4f  8a c7 e8 de ff 88 25 4f  |...%0..0.....%0|
00000450  88 05 5b c3 51 52 32 e4  33 d2 b9 0a 00 f7 f1 80  |..[.QR2.3.....|
00000460  ca 30 88 14 4e 33 d2 3d  0a 00 73 f1 3c 00 74 04  |.0..N3.=.s.<.t.|
00000470  0c 30 88 04 5a 59 c3 50  b4 09 cd 21 58 c3 b8 00  |.0..ZY.P...!X...|
00000480  f0 8e c0 26 a0 fe ff 3c  ff 74 26 3c fe 74 28 3c  |...&...<.t&<.t(<|
00000490  fb 74 24 3c fc 74 26 3c  fa 74 28 3c fc 74 2a 3c  |.t$<.t&<.t(<.t*|
000004a0  f8 74 2c 3c fd 74 2e 3c  f9 74 30 ba 16 01 eb 31  |.t,<.t.<.t0....1|
000004b0  90 ba 00 00 eb 2b 90 ba  15 00 eb 25 90 ba 2d 00  |.....+.....%..-|
000004c0  eb 1f 90 ba 42 00 eb 19  90 ba 5c 00 eb 13 90 ba  |....B.....\....|
000004d0  79 00 eb 0d 90 ba 95 00  eb 07 90 ba ad 00 eb 01  |y.....|
000004e0  90 e8 93 ff c3 50 53 51  52 56 57 b4 30 cd 21 be  |....PSQRVW.0.!.|
000004f0  ce 00 83 c6 0d e8 5c ff  8a c4 83 c6 03 e8 54 ff  |.....\.....T.|
00000500  ba ce 00 e8 71 ff be e2  00 83 c6 13 8a c7 e8 43  |....q.....C|
00000510  ff ba e2 00 e8 60 ff bf  fe 00 83 c7 14 8b c1 e8  |.....|
00000520  1a ff 8a c3 e8 04 ff 83  ef 02 89 05 ba fe 00 e8  |.....|
00000530  45 ff 5a 59 5b 58 5e 5f  c3 b8 10 00 8e d8 e8 3d  |E.ZY[X^_.....=|
00000540  ff e8 a1 ff 32 c0 b4 4c  cd 21  |....2..L.!!|
0000054a

```

Рис. 6 – шестнадцатеричное представление файла lab1_exe.exe

Загрузка СОМ модуля в основную память

1) Какой формат загрузки модуля СОМ? С какого адреса располагается код?
Первые 256 байт (100h) занимает блок PSP, далее, с адреса 100h располагается код.

2) Что располагается с адреса 0?

С адреса 0 располагается PSP размером в 256 байт (100h).

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

Сегментные регистры CS, SS, DS, ES – кода, стека, данных и дополнительный, все указывают на начало блока PSP.

- 4) Как определяется стек? Какую область памяти он занимает? Какие адреса? Стек в .COM модулях определяется автоматически. Сегментный регистр стека (регистр SS) устанавливается на начало блока PSP, а указатель на стек (регистр SP) – на последнюю доступную ячейку памяти в сегменте. То есть адреса с 0h по FFFFh.

Загрузка «хорошего» EXE модуля в основную память

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

При загрузке «хорошего» EXE модуля считается заголовок EXE, выполняется перемещение адресов сегментов, регистры DS и ES устанавливаются в начало PSP, в IP загружается смещение точки входа в программу.

- 2) На что указывают регистры DS и ES?

Регистры DS и ES указывают на начало PSP.

- 3) Как определяется стек?

Стек определяется при объявлении сегмента стека, где необходимо указать сколько памяти будет выделено. Регистры SS и SP указывают на начало и конец стека соответственно.

- 4) Как определяется точка входа?

Точка входа определяется параметром директивы END в конце программы

Выводы.

В процессе выполнения лабораторной работы были изучены структуры и отличительные черты и способы загрузки в основную память загрузочных модулей типа .COM и .EXE.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД.

Название файла: lab1_com.asm

```

TESTPC  SEGMENT
        ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
        ORG 100H
START:  JMP BEGIN
; ДАННЫЕ
TYPE_PC                                DB      'This  PC  type  is
PC'                                     ,0DH,0AH,'$'
TYPE_PC_XT                             DB      'This  PC  type  is
PC/XT'                                 ,0DH,0AH,'$'
TYPE_AT                                DB      'This  PC  type  is
AT'                                   ,0DH,0AH,'$'
TYPE_PS2_M30                           DB      'This  PC  type  is  PS2
m30'                                  ,0DH,0AH,'$'
TYPE_PS2_M50_M60                       DB      'This  PC  type  is  PS2
m50/60',0DH,0AH,'$'
TYPE_PS2_M80                           DB      'This  PC  type  is  PS2 m80:
' ,0DH,0AH,'$'
TYPE_PC_jr                             DB      'This  PC  type  is
PCjr'                                 ,0DH,0AH,'$'
TYPE_PC_CNV                            DB      'This  PC  type  is  PC
Convertible',0DH,0AH,'$'
DOS_VERSION                            DB      'DOS  Version:      .
'                                     ,0DH,0AH,'$'
SERIAL_NUMBER_OEM                      DB      'Serial  number  OEM:
' ,0DH,0AH,'$'
USER_NUMBER                            DB      'Serial  number:
'                                     ,0DH,0AH,'$'
ERROR                                  DB
'ERROR'                               ,0DH,0AH,'$'

```

; ПРОЦЕДУРЫ

;-----

TETR_TO_HEX PROC near

and AL,0Fh

cmp AL,09

jbe NEXT

ADD AL,07

NEXT:

ADD AL,30h

RET

TETR_TO_HEX ENDP

;-----

BYTE_TO_HEX PROC near

; Байт в AL переводится в два символа шест. 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_l
or      AL,30h
MOV     [SI],AL

```

```
end_l:
```

```

        pop     DX
        pop     CX
        RET
BYTE_TO_DEC ENDP
;-----
-----
;-----
-----
;-----
-----
PRINT_STRING PROC near
        PUSH    AX
        MOV     AH,09h
        INT     21h
        pop     AX
        RET
PRINT_STRING endp
;-----
-----
PRINT_PC_TYPE PROC near
; тип IBM PC хранится в байте по адресу 0F000:0FFFEh
; в предпоследнем байте ROM BIOS
        MOV     AX, 0F000h
        MOV     ES, AX
        MOV     AL, ES:[0FFFEh]
;-----
-----
        cmp     AL, 0ffh
        je      print_pc
        cmp     AL, 0feh
        je      print_pc_xt
        cmp     AL, 0fbh
        je      print_pc_xt
        cmp     AL, 0fch

```

```

        je      print_pc_at
        cmp     AL, 0fah
        je      print_pc_ps2_m30
        cmp     AL, 0fch
        je      print_pc_ps2_m50_m60
        cmp     AL, 0f8h
        je      print_pc_ps2_m80
        cmp     AL, 0fdh
        je      print_pc_jr
        cmp     AL, 0f9h
        je      print_pc_cnv
        MOV     DX, offset ERROR
        jmp     print_type
print_pc:
        MOV     DX, offset TYPE_PC
        jmp     print_type
print_pc_xt:
        MOV     DX, offset TYPE_PC_XT
        jmp     print_type
print_pc_at:
        MOV     DX, offset TYPE_AT
        jmp     print_type
print_pc_ps2_m30:
        MOV     DX, offset TYPE_PS2_M30
        jmp     print_type
print_pc_ps2_m50_m60:
        MOV     DX, offset TYPE_PS2_M50_M60
        jmp     print_type
print_pc_ps2_m80:
        MOV     DX, offset TYPE_PS2_M80
        jmp     print_type
print_pc_jr:
        MOV     DX, offset TYPE_PC_jr
        jmp     print_type

```

```

print_pc_cnv:
    MOV     DX, offset TYPE_PC_CNV
    jmp     print_type
print_type:
    CALL    PRINT_STRING
    RET
PRINT_PC_TYPE ENDP
;-----

```

```

-----
PRINT_OS_VERSION PROC near
; Сохранение значений регистров
    PUSH    AX
    PUSH    BX
    PUSH    CX
    PUSH    DX
    PUSH    SI
    PUSH    DI
; Определении версии DOS
    MOV     AH, 30h
    INT     21h
; Версия ОС
    MOV     SI, offset DOS_VERSION
    ADD     SI, 13
    CALL    BYTE_TO_DEC
    MOV     AL, AH
    ADD     SI, 3
    CALL    BYTE_TO_DEC
    MOV     DX, offset DOS_VERSION
    CALL    PRINT_STRING
; Серийный номер OEM
    MOV     SI, offset SERIAL_NUMBER_OEM
    ADD     SI, 19
    MOV     AL, bh
    CALL    BYTE_TO_DEC

```

```

        MOV DX, offset SERIAL_NUMBER_OEM
        CALL PRINT_STRING
; Серийный номер пользователя
        MOV DI, offset USER_NUMBER
        ADD DI, 20
        MOV AX, CX
        CALL WRD_TO_HEX
        MOV AL, bl
        CALL BYTE_TO_HEX
        sub DI, 2
        MOV [DI], AX
        MOV DX, offset USER_NUMBER
        CALL PRINT_STRING
; Восстановление значений регистров
        pop DX
        pop CX
        pop BX
        pop AX
        pop SI
        pop DI
        RET

```

```

PRINT_OS_VERSION ENDP

```

```

; -----

```

```

; КОД
BEGIN:
; Задание лабораторной работы
        CALL PRINT_PC_TYPE
        CALL PRINT_OS_VERSION
; Выход в MS_DOS
        xor     AL,AL
        MOV     AH,4Ch
        INT     21h

```

TESTPC ENDS

END START ; конец модуля, START - точка входа

Название файла: lab1_exe.asm

AStack SEGMENT STACK

DW 128 DUP(?)

AStack ENDS

DATA SEGMENT

; ДАННЫЕ

```

        TYPE_PC                      DB    'This PC type is
PC '          ,0DH,0AH,'$'
        TYPE_PC_XT                    DB    'This PC type is
PC/XT'        ,0DH,0AH,'$'
        TYPE_AT                       DB    'This PC type is
AT '          ,0DH,0AH,'$'
        TYPE_PS2_M30                  DB    'This PC type is PS2
m30'          ,0DH,0AH,'$'
        TYPE_PS2_M50_M60              DB    'This PC type is PS2
m50/60' ,0DH,0AH,'$'
        TYPE_PS2_M80                  DB    'This PC type is PS2 m80:
' ,0DH,0AH,'$'
        TYPE_PC_jr                    DB    'This PC type is
PCjr'         ,0DH,0AH,'$'
        TYPE_PC_CNV                   DB    'This PC type is PC
Convertible',0DH,0AH,'$'
        DOS_VERSION                   DB    'DOS Version:  .
'          ,0DH,0AH,'$'
        SERIAL_NUMBER_OEM             DB    'Serial number OEM:
' ,0DH,0AH,'$'
        USER_NUMBER                   DB    'Serial number:
'          ,0DH,0AH,'$'
        ERROR                          DB
'ERROR'          ,0DH,0AH,'$'
```

DATA ENDS

CODE SEGMENT

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

; ПРОЦЕДУРЫ

;-----

TETR_TO_HEX PROC near

and AL,0Fh

cmp AL,09

jbe NEXT

ADD AL,07

NEXT:

ADD AL,30h

RET

TETR_TO_HEX ENDP

;-----

BYTE_TO_HEX PROC near

; Байт в AL переводится в два символа шест. 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_l
        or      AL,30h
        MOV     [SI],AL
end_l:
        pop     DX
        pop     CX
        RET
BYTE_TO_DEC ENDP
;-----
-----
;-----
-----
;-----
-----
PRINT_STRING PROC near
        PUSH    AX
        MOV     AH,09h
        INT     21h
        pop     AX
        RET
PRINT_STRING endp
;-----
-----
PRINT_PC_TYPE PROC near
; тип IBM PC хранится в байте по адресу 0F000:0FFFEh
; в предпоследнем байте ROM BIOS
        MOV     AX, 0F000h
        MOV     ES, AX
        MOV     AL, ES:[0FFFEh]
;-----
-----
        cmp     AL, 0ffh
        je      print_pc
        cmp     AL, 0feh

```

```

        je      print_pc_xt
        cmp     AL, 0fbh
        je      print_pc_xt
        cmp     AL, 0fch
        je      print_pc_at
        cmp     AL, 0fah
        je      print_pc_ps2_m30
        cmp     AL, 0fch
        je      print_pc_ps2_m50_m60
        cmp     AL, 0f8h
        je      print_pc_ps2_m80
        cmp     AL, 0fdh
        je      print_pc_jr
        cmp     AL, 0f9h
        je      print_pc_cnv
        MOV     DX, offset ERROR
        jmp     print_type
print_pc:
        MOV     DX, offset TYPE_PC
        jmp     print_type
print_pc_xt:
        MOV     DX, offset TYPE_PC_XT
        jmp     print_type
print_pc_at:
        MOV     DX, offset TYPE_AT
        jmp     print_type
print_pc_ps2_m30:
        MOV     DX, offset TYPE_PS2_M30
        jmp     print_type
print_pc_ps2_m50_m60:
        MOV     DX, offset TYPE_PS2_M50_M60
        jmp     print_type
print_pc_ps2_m80:
        MOV     DX, offset TYPE_PS2_M80

```

```

        jmp      print_type
print_pc_jr:
        MOV      DX, offset TYPE_PC_jr
        jmp      print_type
print_pc_cnv:
        MOV      DX, offset TYPE_PC_CNV
        jmp      print_type
print_type:
        CALL     PRINT_STRING
        RET
PRINT_PC_TYPE ENDP
;-----

```

```

PRINT_OS_VERSION PROC near
; Сохранение значений регистров
        PUSH     AX
        PUSH     BX
        PUSH     CX
        PUSH     DX
        PUSH     SI
        PUSH     DI
; Определении версии DOS
        MOV      AH, 30h
        INT      21h
; Версия ОС
        MOV      SI, offset DOS_VERSION
        ADD      SI, 13
        CALL     BYTE_TO_DEC
        MOV      AL, AH
        ADD      SI, 3
        CALL     BYTE_TO_DEC
        MOV      DX, offset DOS_VERSION
        CALL     PRINT_STRING
; Серийный номер OEM

```

```

MOV SI, offset SERIAL_NUMBER_OEM
ADD SI, 19
MOV AL, bh
CALL BYTE_TO_DEC
MOV DX, offset SERIAL_NUMBER_OEM
CALL PRINT_STRING
; Серийный номер пользователя
MOV DI, offset USER_NUMBER
ADD DI, 20
MOV AX, CX
CALL WRD_TO_HEX
MOV AL, bl
CALL BYTE_TO_HEX
sub DI, 2
MOV [DI], AX
MOV DX, offset USER_NUMBER
CALL PRINT_STRING
; Восстановление значений регистров
pop DX
pop CX
pop BX
pop AX
pop SI
pop DI
RET
PRINT_OS_VERSION ENDP

```

```

;-----

```

```

-----

```

```

; КОД
Main PROC FAR
    MOV     AX, DATA
    mov     DS, AX
; Задание лабораторной работы
    CALL PRINT_PC_TYPE

```

```
        CALL PRINT_OS_VERSION
; Выход в MS_DOS
        xor     AL,AL
        MOV     AH,4Ch
        INT     21h
Main ENDP

CODE    ENDS
        END     Main
```