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	Нистратов Д.Г.
Преподаватель	Ефремов М.А.

Санкт-Петербург 2021

Постановка задачи.

Исследовать различия в структурах исходных текстов модулей типов .COM и .EXE, структур файлов загрузочных модулей и способов их загрузки в основную память.

Функции и структуры данных

Таблица 1 – Функции и структуры данных

Название функций	Описание функций
TETR_TO_HEX	Перевод из четверичной c/c в шестнадцатеричную c/c
BYTE_TO_HEX	Перевод из двоичной c/c в шестнадцатеричную c/c
WRD_TO_HEX	Перевод 2 байтов в шестнадцатеричную c/c
BYTE_TO_DEC	Перевод из двоичной с/с в десятичную с/с
WRITE	Вывод строки на экран
OS_TYPE	Определение версии ОС, серийного номера и ОЕМ номера
PC_TYPE	Определение типа РС

Последовательность действий

- Шаг 1. Напишите текст исходного .COM модуля, который определяет тип PC и версию системы.
- Шаг 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.
- Шаг 3. Сравните исходные тексты для .COM и .EXE модулей. Ответьте на контрольные вопросы "Отличие исходных текстов COM и EXE программ"

- Шаг 4. Запустите FAR и откройте (F3/F4) файл загрузочного модуля. СОМ 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 СОМ и EXE модулей».
- Шаг 5. Откройте отладчик TD.EXE и загрузите. СОМ. Ответьте на контрольные вопросы «Загрузка СОМ модуля в основную память». Представьте в отчете план загрузки модуля. СОМ в основную память.
- Шаг 6. Откройте отладчик TD.EXE и загрузите «хороший». EXE. Ответьте на контрольные вопросы «Загрузка «хорошего» EXE модуля в основную память».
- Шаг 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. При выполнении лабораторной работы был написан .COM модуль, определяющий тип РС и версию системы см. *Изображение 1 – пример работы .COM модуляИзображение 1*, а также был построен "плохой" .EXE модуль из исходного текста для .COM модуля см. *Изображение 2*.

```
D:\LETI\OS\MASM>LAB1.COM
PC TYPE: AT
OS version: 5.0
OEM: 255
Serial number: 000000
```

Изображение 1 – пример работы .СОМ модуля

Изображение 2 – "плохой" .ЕХЕ модуль

Шаг 2. Был описан текст "хорошего" .EXE модуля, выполняющий те же функции что и модуль в Шаге 1, см. *Изображение 3*.

```
D:\LETI\OS\MASM>lab1_exe.exe
PC TYPE: AT
OS version: 5.0
OEM: 255
Serial number: 000000
```

Изображение 3 - "хороший" .ЕХЕ модуль

Шаг 3. Отличия исходных текстов COM и EXE программ

Сколько сегментов должна содержать СОМ-программа?
 Ответ: СОМ-программа должна содержать только один сегмент.

2) ЕХЕ программа?

(FF00h).

Ответ: ЕХЕ программа должна содержать один или более сегментов.

- 3) Какие директивы должны обязательно быть в тексте COM-программы? Ответ: обязательно должна быть директива *org 100h* для смещения программного сегмента, т. к. первые 256 байтов занимает PSP. Также необходима процедура ASSUME, чтобы сегмент кода и сегмент данных указывали на один общий сегмент.
- 4) Все ли форматы команд можно использовать в СОМ-программе? Ответ: в СОМ-программе нельзя использовать команды вида mov <pегистр>, <сегмент>, так как в .COM программе отсутствует таблица настроек, содержащая описание адресов, зависящих от загрузочного модуля. А также нельзя использовать команды превышающие 64Кб, так как .COM программа ограничена в размере сегмента в 64Кб

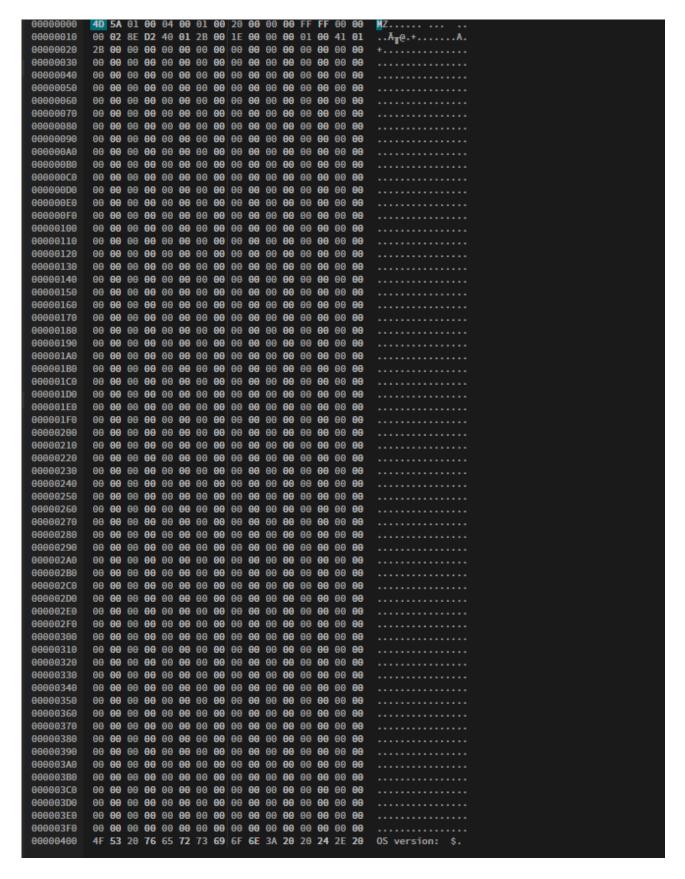
Шаг 4. Рассмотрим шестнадцатеричный вид загрузочного файла модуля .COM и "плохого" .EXE, см. Изображение 4 и Изображение 5. А также загрузочный файл "хорошего" модуля .EXE, см. Изображение 6.

```
E9 E3 01 4F 53 20 76 65 72 73 69 6F 6E 3A 20 20
                                                           8π.OS version:
          24 2E 20 20 0D 0A 24 53 65 72 69 61 6C 20 6E 75
                                                           $. ..$Serial nu
00000010
                                                                      ..$0
00000020
         6D 62 65 72 3A 20 20 20 20 20 20 0D 0A 24 4F
                                                          mber:
0000030
         45 4D 3A 20 20 20 20 0D 0A 24 50 43 20 54 59 50
                                                          EM:
                                                                 ..$PC TYP
         45 3A 20 24 50 43 0D 0A 24 50 43 2F 58 54 0D 0A
00000040
                                                          E: $PC..$PC/XT..
00000050
         24 41 54 0D 0A 24 50 53 32 20 6D 6F 64 65 6C 20
                                                          $AT..$PS2 model
         33 30 0D 0A 24 50 53 32 20 6D 6F 64 65 6C 20 35
                                                          30..$PS2 model 5
00000060
          30 20 6F 72 20 36 30 0D 0A 24 50 53 32 20 6D 6F
00000070
                                                           0 or 60..$PS2 mo
00000080
         64 65 6C 20 38 30 0D 0A 24 50 43 6A 72 0D 0A 24
                                                           del 80..$PCjr..$
00000090
         50 43 20 43 6F 6E 76 65 72 74 69 62 6C 65 0D 0A
                                                           PC Convertible..
         24 20 20 0D 0A 24 24 0F 3C 09 76 02 04 07 04 30
000000A0
                                                           $ ..$$.<.v....0
        C3 51 8A E0 E8 EF FF 86 C4 B1 04 D2 E8 E8 E6 FF
                                                           -QèαΦ∩ å--∭.-TΦΦμ
000000B0
         59 C3 53 8A FC E8 E9 FF 88 25 4F 88 05 4F 8A C7
                                                           Y-Sèn⊕⊖ ê%0ê.0è
00000C0
                                                           Φ  ê%0ê.[ QR2Σ3π
00000D0
         E8 DE FF 88 25 4F 88 05 5B C3 51 52 32 E4 33 D2
                                                           ╣..≈±Ç些0ê.N3⊤=..
         B9 0A 00 F7 F1 80 CA 30 88 14 4E 33 D2 3D 0A 00
00000E0
                                                           s±<.t..0ê.ZY P.
         73 F1 3C 00 74 04 0C 30 88 04 5A 59 C3 50 B4 09
00000F0
                                                           =!X | | 0=!PSQRV₩
         CD 21 58 C3 B4 30 CD 21 50 53 51 52 56 57 BE 03
00000100
                                                           .â = . Φ − | . . Φα = .
00000110 01 83 C6 OC E8 C3 FF BA 03 O1 E8 E0 FF BE 11 O1
                                                           00000120 83 C6 01 8A C4 E8 B2 FF BA 11 01 E8 CF FF BE 2F
.â⊧.è ⊕í  /.æ ┐
00000140 17 01 83 C7 14 8A C3 E8 67 FF 8B C1 E8 73 FF 83
                                                           ..â⊩.e⊨ag ï⊢as â
                                                           n.ë.∥..∳ú _^ZY[X
         EF 02 89 05 BA 17 01 E8 A3 FF 5F 5E 5A 59 5B 58
00000150
         C3 50 53 51 52 56 57 06 B8 00 F0 8E C0 26 A0 FE
                                                           PSQRVW.¬.≡ÄL&á•
00000160
00000170
         FF BA 3A 01 E8 86 FF 3C FF 74 23 3C FE 74 25 3C
                                                            :.⊕å < t#<•t%<
00000180
         FB 74 21 3C FC 74 23 3C FA 74 25 3C FC 74 27 3C
                                                           √t!<nt#<·t%<nt'<
         F8 74 29 3C FD 74 2B 3C F9 74 2D EB 31 90 BA 44
                                                           °t)<2t+<·t-51ÉD
00000190
                                                           .δ8έ Ι.δ2έ Q.δ,έ
         01 EB 38 90 BA 49 01 EB 32 90 BA 51 01 EB 2C 90
000001A0
000001B0 BA 56 01 EB 26 90 BA 65 01 EB 20 90 BA 7A 01 EB
                                                           V.δ&É e.δ É z.δ
                                                           .É ë.δ.É É.δ.ÉΦα
000001C0
         1A 90 BA 89 01 EB 14 90 BA 90 01 EB 0E 90 E8 E0
                                                           •¶i.ê.êg.ï┖⊕. _^
000001D0
          FE BB A1 01 88 07 88 67 01 8B D3 E8 1F FF 5F 5E
                                                           ZΥ[X. +x +. 2 4 L
          5A 59 5B 58 07 C3 E8 78 FF E8 18 FF 32 C0 B4 4C
000001E0
000001F0
         CD 21 +
```

Изображение 4 - шестнадцатеричное преставление модуля .СОМ

-Untitled: X	LABIT.EX																
00000000	4D 5			99	83	88	88	88	20	88	00	88	FF	FF	88	88	IZ≥
00000000	99 8				88	81	88	88	1E	88	88	88	81	88	88	88	/
00000020	99 8	0 (90	88	88	88	88	88	88	88	88	88	88	88	88	88	
0000030	99 8			88	88	88	88	88	88	88	88	88	88	88	88	88	
00000040 00000050	99 8			88 88	99	99	88	88	99	88	88	88	98 98	99	88	66	
00000030	99 8			88	88	88	88 88	88	99	88	88	88	88	88	88	88	
00000070				88	88	88	88	88	88	88	88	88	88	88	88	88	
08000080	99 9	8 (90	88	88	88	88	88	99	88	88	88	88	88	88	88	
00000090	99 8			88	88	88	88	88	88	88	88	88	88	88	88	88	
989999A9	99 9			88	88	99	88	88	99	88	88	88	88	88	88	88	
000000B0 000000C0	99 9			88 88	99	99	88	88 88	99	88	88	88	99 98	99	99	66	
000000D0				88	88	88	88	88	88	88	88	88	88	88	88	88	
000000E0	99 8	8 (88	88	88	88	88	88	88	88	88	88	88	88	88	88	
000000F0				88	88	88	88	88	88	88	88	88	88	88	88	88	
00000100 00000110	99 9			88 88	99	99	88	88	99	88	88	88	88	88	88	88	
00000110	99 8			88	99	88	88	88	99	88	88	88	99 99	99	99	88	
00000130				88	88	88	88	88	88	88	88	88	88	88	88	88	
00000140	99 8	8 (88	88	88	88	88	88	88	88	88	88	88	88	88	88	
00000150					88	88	88	88	88	88	88	88	88	88	88	88	
00000170	99 9			88	88	88	88	88	88	88	88	88	88	88	88	88	
00000170 00000180	99 8			88 88	99	99	88	88	99	88	88	88	99 99	99	99	66	
00000180					98	98	88	88	98	88	88	88	88	88	88	88	
000001A0	88 8			88	88	88	88	88	88	88	88	88	88	88	88	88	
000001B0	99 8	8 (90	88	88	88	88	88	88	88	88	88	88	88	88	88	
000001C0	88 8			88	88	88	88	88	88	88	88	88	88	88	88	88	
000001D0 000001E0					99	99	99	88	99	88	88	88	88	99	88	66	
000001E0	99 8			88 88	99	98	88	88	99	88	66	88	98 98	88	88	66 66	
8888288	88 8				88	88	88	88	88	88	88	88	88	88	88	88	
00000210	88 B	8 (88	88	88	88	88	88	88	88	88	88	88	88	88	88	
00000220				88	88	88	88	88	88	88	88	88	88	88	88	88	
00000230 00000240					88	88	88	88	88	88	88	88	88	88	88	88	
88888258	99 8				99	98	88	88	98	88	88	88	98 98	88	88	66 66	
88888268				88	88	88	88	88	88	88	88	88	88	88	88	88	
88888278			88	88	88	88	88	88	88	88	88	88	88	88	88	88	
00000280	88 8	8 (88	88	88	88	88	88	88	88	88	88	88	88	88	88	
00000290					88	88	88	88	88	88	88	88	88	88	88	88	
000002A0	99 9			88 88	99	98	88	88	98	88	88	88	98 98	88	88	66 66	
888882C8	99 8				88	88	88	88	88	88	88	88	88	88	88	88	
000002D0		8 (88	88	88	88	88	88	88	88	88	88	88	88	
000002E0	99 8	0 (80	88	88	88	88	88	88	88	88	88	88	88	88	88	
000002F0		8 (88	88	88	88	88	88	88	88	88	88	88	88	
00000300 00000310		3 (53 en	28	76	65	72	73	69	6F	6E	34	28	28	em.05 version:
00000310	24 Z 60 6	E .			BD 3A	8A 28	24 28	53 28	65 28	72 28	69 28	61 28	ec ed	28 8A	6E 24	75 4F	\$\$Serial nu mber:\$0
00000330			BA :		20	28	28		BA	24		43					EM:\$PC TYP
00000340	45 3	A :	20	24	50	43	8D	ØA.	24	50	43	2F	58	54	8D	8A	E: \$PC\$PC/XT
00000350		11. !			BA.		50	53	32			6F	64		60	28	\$AT\$PS2 model
00000360	33 3				24	58	53	32	28	6D	6F	64	65	60	28	35	30\$PS2 model 5
00000370 00000380		9 (6F 6C				38 8D			24 58			32 72		6D BA	6F 24	0 or 60\$P52 mo
00000380		_	20	43	6F	6E	76	65	72	74	69	67	60	65	9D	BA.	del 80\$PCjr\$ PC Convertible
000003A0	24 2			8D	ва	24	24	8F	30	89	76	82	84	87	84	38	\$\$\$.<.v8
00000380	C3 5	1.	BA.	E0	E8	EF	FF	86	C4	B1	64	02	E8	E8	E6	FF	Qésen a B. yeou
000003C0	59 C												05				Y Se os exos.os
000003D0	E8 D																o∥ č®0č.[QR2x3
000003E0	89 6 73 F												D2 C3			89	=±Ç=8ê.N3⊤= 5±<.t8ê.ZY ₽ .
88888488	CD 2																-!XHe-!PSQRVW .
00000410	91.8																.á .o o J
88988428	83 C												CF			2F	ā .e-o∎ e/
00000440	91 8																.āḥ.ēļoi [/.ol]
00000440 00000450	17 8 EF 8																a .e og Tlos a
00000450	C3 5												CB				n.ē. eŭ_^ZY[X PSQRVW.q.=Ä ^L &å-
88988478	FF B																:.oa c t#c-t%c
00000480	FB 7	4 :	21	3C	FC	74	23	30	FA	74	25	30	FC	74	27	30	√t!c*t#c-t%c*t'c
00000490	F8 7																't)<'t+<-t-alf D
999994A9	91 E												81				.08É I.02É Q.0,É
00000480 000004C0	BA 5												BA BE				V.c&£ e.c £ z.c .£ ē.c.£ £.c.£oc
000004C0	FE B																-1.6.6g.140^
000004E0	5A 5																ZY[X. 0x 0. 24L
000004F0	CD 2	1															-:
		_															

Изображение 5 - шестнадцатеричное представление "плохого" модуля .EXE



Изображение 6 - шестнадцатеричное представление "хорошего" модуля .EXE

Шаг 5. Отличие форматов файлов СОМ и ЕХЕ модулей

- Какова структура файла СОМ? С какого адреса располагается код?
 Ответ: СОМ содержит данные и машинные команды. Код располагается в начале 0h, но при загрузке модуля смещается на 100h.
- 2) Какова структура "плохого" EXE? С какого адреса располагается код? Что располагается с адреса 0?

Ответ: В плохом ЕХЕ данные и код содержаться в одном сегменте. Код располагается с 300h. С адреса 0h начинается управляющая информация загрузчика, содержащая заголовок и таблицу настроек, а также смещение 100h.

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

Ответ: В хорошем ЕХЕ стек и данные разделены по сегментам. В отличии от плохого ЕХЕ, в хорошем ЕХЕ PSP и память под стек, см. Изображение 7, выделяется до кода (400h).

```
5 STACKSG SEGMENT STACK
6 DW 256 DUP(?)
7 STACKSG ENDS
```

Изображение 7 - сегмент стека

Шаг5. Загрузка модуля .СОМ в основную память

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

Ответ: определяется сегментный участок, с достаточным местом для загрузки программы. Создается блок памяти для PSP и программы, затем подгружается СОМ-файл. Код располагается с 100h.

2) Что располагается с адреса 0?

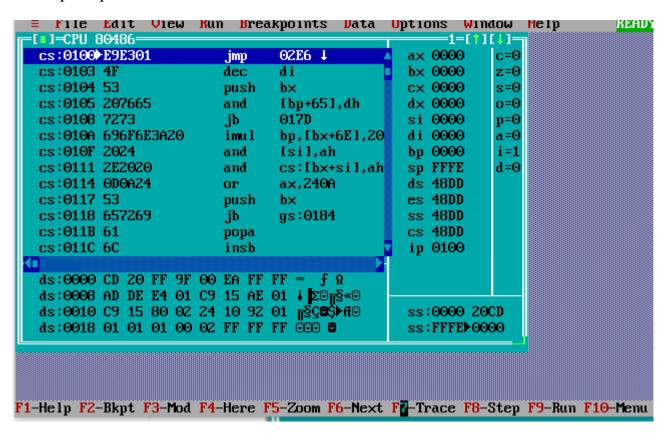
Ответ: PSP

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

Ответ: ds, es, ss, cs имеют значение 48DD и указывают на PSP. SP имеет значение FFFE и указывает на конец сегмента.

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

Ответ: Стек определяется автоматически. Стек занимает всю область. Адреса расположены от 0h до FFFEh.



Изображение 8 - Загрузка модуля СОМ

Шаг 6. Загрузка "хорошего" ЕХЕ модуля в память

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

Ответ: .EXE загружается по смещению 100h от PSP, считывает информацию заголовка и осуществляет перемещение адресов сегментов. DS – 48DD, ES – 48DD, SS – 48DD, CS – 4918.

2) На что указывают регистры DS и ES?

Ответ: PSP

3) Как определяется стек?

Ответ: с помощью директивы .stack

4) Как определяется точка входа?

Ответ: с помощью директивы END

Заключение.

В ходе лабораторный работы были исследованы загрузочные модули .COM и .EXE, их структура и загрузка в основную память.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab1.asm

```
; .COM and .EXE diferences in strucures
; 15.02.2021
; Nistratov Dmitry
; Шаблон текста программы на ассемблере для модуля типа . СОМ
TESTPC SEGMENT
       ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
       ORG 100H
START: JMP BEGIN
; Данные
VERSION DB "OS version: $"; 12
DOT VERSION DB ". ", ODH, OAH, '$'; 1
SERIAL DB "Serial number: ", ODH, OAH, '$'; 15 + 6 - 2
OEM DB "OEM: ",ODH, OAH, '$'; 5
PC TYPE MSG DB "PC TYPE: $"
PC TYPE 1 DB "PC", ODH, OAH, '$'
PC TYPE 2 DB "PC/XT", ODH, OAH, '$'
PC_TYPE_3 DB "AT", ODH, OAH, '$'
PC TYPE 4 DB "PS2 model 30", ODH, OAH, '$'
PC TYPE 5 DB "PS2 model 50 or 60", 0DH, 0AH, '$'
PC TYPE 6 DB "PS2 model 80", 0DH, 0AH, '$'
PC_TYPE_7 DB "PCjr", ODH, OAH, '$'
PC TYPE 8 DB "PC Convertible", ODH, OAH, '$'
PC TYPE 9 DB " ", ODH, OAH, '$'
; 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 AL переводится в два символа шестн. числа АХ
           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
           рор СХ ; В АН младшая цифра
           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 BX
       ret
WRD TO HEX ENDP
;-----
BYTE TO DEC PROC near
; Перевод в 10c/c, 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 1: pop DX
     pop CX
     ret
BYTE TO DEC ENDP
; КОД
WRITE PROC NEAR
       push ax
       mov ah, 09h
       int 21h
       pop ax
       ret
WRITE ENDP
OS_TYPE PROC NEAR
       mov ah, 30h
       int 21H
       push ax
       push bx
       push cx
       push dx
       push si
       push di
```

```
mov si, offset VERSION
        add si, 12
        call BYTE TO DEC
        mov dx, offset VERSION
        call WRITE
        mov si, offset DOT_VERSION
        add si, 1
        mov al, ah
        call BYTE TO DEC
        mov dx, offset DOT_VERSION
        call WRITE
        mov si, offset OEM
        add si, 7
        mov al, bh
        call BYTE TO DEC
        mov dx, offset OEM
        call WRITE
        mov di, offset SERIAL
        add di, 20
        mov al, bl
        call BYTE TO HEX
        mov ax, cx
        call WRD TO HEX
        sub di, \overline{2}
        mov [di], ax
        mov dx, offset SERIAL
        call WRITE
        pop di
        pop si
        pop dx
        pop cx
        pop bx
        pop ax
        ret
OS TYPE ENDP
PC TYPE PROC NEAR
        push ax
        push bx
        push cx
        push dx
        push si
        push di
        push es
        mov ax, 0f000h
        mov es, ax
        mov al, es:[0fffeh]
        mov dx, offset PC_TYPE_MSG
        call WRITE
        cmp al, Offh
        jz pc 1
        cmp al, Ofeh
```

```
jz pc 2
        cmp al, Ofbh
        jz pc 2
        cmp al, Ofch
        jz pc_3
        cmp al, Ofah
        jz pc 4
        cmp al, Ofch
        jz pc_5
        cmp al, 0f8h
        jz pc 6
        cmp al, 0fdh
        jz pc_7
        cmp al, 0f9h
        jz pc 8
        jmp pc unknown
pc_1:
        mov dx, offset PC TYPE 1
        jmp print
pc_2:
        mov dx, offset PC_TYPE 2
        jmp print
pc 3:
        mov dx, offset PC_TYPE_3
        jmp print
pc_4:
        mov dx, offset PC TYPE 4
        jmp print
pc_5:
        mov dx, offset PC_TYPE 5
        jmp print
pc 6:
        mov dx, offset PC_TYPE_6
        jmp print
pc_7:
        mov dx, offset PC TYPE 7
        jmp print
pc_8:
        mov dx, offset PC TYPE 8
        jmp print
pc_unknown:
        call BYTE TO HEX
        mov bx, offset PC TYPE 9
        mov [bx], al
        mov [bx+1], ah
        mov dx, bx
print:
        call WRITE
        pop di
        pop si
        pop dx
        pop cx
        pop bx
        pop ax
        pop es
```

```
ret
PC TYPE ENDP
BEGIN:
       call PC TYPE
       call OS_TYPE
; Выход в DOS
       xor AL, AL
       mov AH, 4Ch
       int 21H
TESTPC ENDS
       END START ; Конец модуля, START - точка входа
Название файла: lab1_exe.asm
; .COM and .EXE diferences in strucures
; 15.02.2021
; Nistratov Dmitry
STACKSG SEGMENT STACK
    DW 256 DUP(?)
STACKSG ENDS
DATASG SEGMENT
    VERSION DB "OS version: $"; 12
    DOT VERSION DB ". ", ODH, OAH, '$'; 1
    SERIAL DB "Serial number: ", ODH, OAH, '$'; 15 + 6 - 2
                 ",ODH, OAH, '$'; 5
    OEM DB "OEM:
    PC TYPE MSG DB "PC TYPE: $"
      PC_TYPE_2 DB "PC/XT", ODH, OAH, '$'
    PC TYPE 3 DB "AT", ODH, OAH, '$'
    PC TYPE 4 DB "PS2 model 30", ODH, OAH, '$'
    PC TYPE 5 DB "PS2 model 50 or 60", 0DH, 0AH, '$'
    PC TYPE 6 DB "PS2 model 80", 0DH, 0AH, '$'
    PC TYPE 7 DB "PCjr", ODH, OAH, '$'
    PC TYPE 8 DB "PC Convertible", ODH, OAH, '$'
    PC TYPE 9 DB " ", ODH, OAH, '$'
DATASG ENDS
CODESG SEGMENT
    ASSUME SS:STACKSG, DS:DATASG, CS:CODESG
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 AL переводится в два символа шестн. числа АХ
    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
              ; В АН младшая цифра
   pop CX
   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 BX
   ret
WRD TO HEX ENDP
;-----
BYTE TO DEC PROC near
; Перевод в 10c/c, 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 1: pop DX
   pop CX
   ret
BYTE TO DEC ENDP
;-----
; КОД
WRITE PROC NEAR
   push ax
   mov ah, 09h
   int 21h
   pop ax
```

```
ret
WRITE ENDP
OS TYPE PROC NEAR
    mov ah, 30h
    int 21H
    push ax
    push bx
    push cx
    push dx
    push si
    push di
    mov si, offset VERSION
    add si, 12
    call BYTE TO DEC
    mov dx, offset VERSION
    call WRITE
    mov si, offset DOT VERSION
    add si, 1
    mov al, ah
    call BYTE TO DEC
    mov dx, offset DOT VERSION
    call WRITE
    mov si, offset OEM
    add si, 7
    mov al, bh
    call BYTE TO DEC
    mov dx, offset OEM
    call WRITE
    mov di, offset SERIAL
    add di, 20
    mov al, bl
    call BYTE TO HEX
    mov ax, cx
    call WRD TO HEX
    sub di, 2
    mov [di], ax
    mov dx, offset SERIAL
    call WRITE
    pop di
    pop si
    pop dx
    pop cx
    pop bx
    pop ax
    ret
OS TYPE ENDP
PC_TYPE PROC NEAR
    push ax
    push bx
    push cx
```

```
push dx
    push si
    push di
    push es
    mov ax, 0f000h
    mov es, ax
    mov al, es:[0fffeh]
    mov dx, offset PC TYPE MSG
    call WRITE
    cmp al, Offh
    jz pc 1
    cmp al, Ofeh
    jz pc 2
    cmp al, Ofbh
    jz pc 2
    cmp al, Ofch
    jz pc_3
    cmp al, Ofah
    jz pc 4
    cmp al, Ofch
    jz pc_5
    cmp al, 0f8h
    jz pc 6
    cmp al, Ofdh
    jz pc_7
    cmp al, 0f9h
    jz pc 8
    jmp pc_unknown
pc 1:
    mov dx, offset PC_TYPE_1
    jmp print
    mov dx, offset PC TYPE 2
    jmp print
pc 3:
    mov dx, offset PC TYPE 3
    jmp print
pc 4:
    mov dx, offset PC_TYPE_4
    jmp print
    mov dx, offset PC TYPE 5
    jmp print
    mov dx, offset PC_TYPE_6
    jmp print
pc 7:
    mov dx, offset PC TYPE 7
    jmp print
pc 8:
    mov dx, offset PC_TYPE_8
    jmp print
pc_unknown:
    call BYTE TO HEX
    mov bx, offset PC TYPE 9
    mov [bx], al
```

```
mov [bx+1], ah
    mov dx, bx
print:
    call WRITE
   pop di
    pop si
    pop dx
    рор сх
    pop bx
    pop ax
    pop es
   ret
PC_TYPE ENDP
MAIN PROC FAR
   mov ax, DATASG
   mov ds,ax
    call PC TYPE
    call OS_TYPE
; Выход в DOS
   xor AL, AL
   mov AH,4Ch
   int 21H
MAIN ENDP
CODESG ENDS
```

END MAIN