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. 9381 _____ Судаков Е.В.

Преподаватель _____ Ефремов М. А.

Санкт-Петербург
2021

Цель работы.

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

Ход работы.

1. На основе шаблона, приведенного в методических указаниях, был написан текст исходного .COM модуля lab1_com.asm, который определяет тип PC и версию системы. Далее при помощи MASM.exe и компоновщика LINK.exe был скомпилирован “плохой” .EXE модуль. При помощи EXE2BIN.EXE был построен правильный .COM модуль.

2. Был написан текст исходного .exe модуля lab1_exe.asm. Далее при помощи MASM.exe и компоновщика LINK.exe был скомпилирован правильный.EXE модуль.

3. Было выполнено сравнение исходных текстов lab1_com.asm и lab1_exe.asm.

4. При помощи программы **hexdump** файлы загрузочных модулей lab1_com.com, lab1_com.exe и lab1_exe.exe были открыты в шестнадцатеричном виде.

5. Был исследован загрузочный модуль .com при помощи отладчика.

6. Был исследован загрузочный модуль .exe при помощи отладчика.

Процедуры.

Название процедуры	Назначение
TETR_TO_HEX	Перевод тетрады (4-ех младших байтов AL) в 16-ичную СС и ее представление в виде символа
BYTE_TO_HEX	Перевод байта AL в 16-ичную СС и его представление в виде символов
WORD_TO_HEX	Перевод слова AX в 16-ичную СС и его представление в виде символов
BYTE_TO_DEC	Перевод байта AL в 10-ичную СС и его представление в виде символов
PRINT_PC_TYPE	Вывод информации о PC
PRINT_MSDOS_VERSION	Печатает на экран версию ОС, серийные номера OEM и пользователя.

Ответы на контрольные вопросы.

Отличия исходных текстов COM и EXE программ

1)Сколько сегментов должна содержать COM-программа?

Один сегмент.

2)EXE-программа?

От одного и более. Зависит от модели памяти.

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

ORG 100h, Assume, END

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

Нельзя использовать операции, связанные со взятием адреса сегмента: т.к. в .com файлах нет таблицы настройки.

Отличия форматов файлов COM и EXE модулей

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

Код, данные, стек : все в одном сегменте. Код, как и данные, начинается с адреса 0h.

```
yudjin@yudjin-desktop:~/Documents/study/2kurs/4sem/OS/MASM$ hexdump -C LAB1_COM.COM
00000000  e9 c6 01 50 43 20 74 79  70 65 3a 20 24 50 43 0d  |...PC type: $PC.|
00000010  0a 24 50 43 2f 58 54 0d  0a 24 41 54 0d 0a 24 50  |.$PC/XT..$AT..$P|
00000020  53 32 20 6d 6f 64 65 6c  20 33 30 0d 0a 24 50 53  |S2 model 30..$PS|
00000030  32 20 6d 6f 64 65 6c 20  38 30 0d 0a 24 50 43 6a  |2 model 80..$PCj|
00000040  72 0d 0a 24 50 43 20 43  6f 6e 76 65 72 74 69 62  |r..$PC Convertib|
00000050  6c 65 0d 0a 24 50 43 20  75 6e 6b 6e 6f 77 6e 0d  |le..$PC unknown.|
00000060  0a 24 4d 53 44 4f 53 20  20 76 65 72 73 69 6f 6e  |.$MSDOS  version|
00000070  3a 20 24 20 20 2e 20 20  0d 0a 24 4f 45 4d 20 20  |: $  .  ..$OEM  |
00000080  76 65 72 73 69 6f 6e 3a  20 24 20 20 0d 0a 24 55  |version: $  ..$U|
00000090  73 65 72 20 73 65 72 69  61 6c 20 6e 75 6d 62 65  |ser serial numbe|
000000a0  72 20 3a 20 24 20 20 20  20 20 20 20 20 0d 0a 24  |r : $        ..$|
000000b0  24 0f 3c 09 76 02 04 07  04 30 c3 51 8a e0 e8 ef  |$.<.v....0.Q....|
000000c0  ff 86 c4 b1 04 d2 e8 e8  e6 ff 59 c3 53 8a fc e8  |..........Y.S...|
000000d0  e9 ff 88 25 4f 88 05 4f  8a c7 e8 de ff 88 25 4f  |...%O..O......%O|
000000e0  88 05 5b c3 51 52 32 e4  33 d2 b9 0a 00 f7 f1 80  |..[.QR2.3.......|
000000f0  ca 30 88 14 4e 33 d2 3d  0a 00 73 f1 3c 00 74 04  |.0..N3.=..s.<.t.|
00000100  0c 30 88 04 5a 59 c3 50  b4 09 cd 21 58 c3 b8 00  |.0..ZY.P...!X...|
00000110  f0 8e c0 26 a0 fe ff ba  03 01 e8 ea ff 3c ff 74  |...&.........<.t|
00000120  1f 3c fe 74 21 3c fb 74  1d 3c fc 74 1f 3c fa 74  |.<.t!<.t.<.t.<.t|
00000130  21 3c f8 74 23 3c fd 74  25 3c f9 74 27 eb 2b 90  |!<.t#<.t%<.t'.+.|
00000140  ba 0d 01 eb 2b 90 ba 12  01 eb 25 90 ba 1a 01 eb  |....+.....%.....|
00000150  1f 90 ba 1f 01 eb 19 90  ba 2e 01 eb 13 90 ba 3d  |...............=|
00000160  01 eb 0d 90 ba 44 01 eb  07 90 ba 55 01 eb 01 90  |.....D.....U....|
00000170  e8 94 ff c3 b4 30 cd 21  be 74 01 8a ec e8 64 ff  |.....0.!.t....d.|
00000180  8a c5 83 c6 03 e8 5c ff  ba 62 01 e8 79 ff ba 73  |......\..b..y..s|
00000190  01 e8 73 ff 8a c7 e8 22  ff be 8a 01 89 04 ba 7b  |..s...."....... {|
000001a0  01 e8 63 ff ba 8a 01 e8  5d ff 8a c3 e8 35 ff bf  |..c.....]....5..|
000001b0  a5 01 89 05 8b c1 83 c7  05 e8 10 ff ba 8f 01 e8  |................|
000001c0  45 ff ba a5 01 e8 3f ff  c3 e8 a8 ff 32 c0 b4 4c  |E.....?.....2..L|
000001d0  cd 21 c3                                          |.!.|
000001d3
```

Рис.1. Файл правильного .com модуля

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

Код, данные, стек : все в одном сегменте. Код, как и данные, начинается с адреса 300h.

```
yudjin@yudjin-desktop:~/Documents/study/2kurs/4sem/OS/MASM$ hexdump -C LAB1_COM.EXE
00000000  4d 5a d3 00 03 00 00 00  20 00 00 00 ff ff 00 00  |MZ..... .......|
00000010  00 00 ca 82 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 c6 01 50 43 20 74 79  70 65 3a 20 24 50 43 0d  |...PC type: $PC.|
00000310  0a 24 50 43 2f 58 54 0d  0a 24 41 54 0d 0a 24 50  |.$PC/XT..$AT..$P|
00000320  53 32 20 6d 6f 64 65 6c  20 33 30 0d 0a 24 50 53  |S2 model 30..$PS|
00000330  32 20 6d 6f 64 65 6c 20  38 30 0d 0a 24 50 43 6a  |2 model 80..$PCj|
00000340  72 0d 0a 24 50 43 20 43  6f 6e 76 65 72 74 69 62  |r..$PC Convertib|
00000350  6c 65 0d 0a 24 50 43 20  75 6e 6b 6e 6f 77 6e 0d  |le..$PC unknown.|
00000360  0a 24 4d 53 44 4f 53 20  20 76 65 72 73 69 6f 6e  |.$MSDOS  version|
00000370  3a 20 24 20 20 2e 20 20  0d 0a 24 4f 45 4d 20 20  |: $  .  ..$OEM  |
00000380  76 65 72 73 69 6f 6e 3a  20 24 20 20 0d 0a 24 55  |version: $  ..$U|
00000390  73 65 72 20 73 65 72 69  61 6c 20 6e 75 6d 62 65  |ser serial numbe|
000003a0  72 20 3a 20 24 20 20 20  20 20 20 20 20 0d 0a 24  |r : $        ..$|
000003b0  24 0f 3c 09 76 02 04 07  04 30 c3 51 8a e0 e8 ef  |$.<.v....0.Q....|
000003c0  ff 86 c4 b1 04 d2 e8 e8  e6 ff 59 c3 53 8a fc e8  |..........Y.S...|
000003d0  e9 ff 88 25 4f 88 05 4f  8a c7 e8 de ff 88 25 4f  |...%O..O......%O|
000003e0  88 05 5b c3 51 52 32 e4  33 d2 b9 0a 00 f7 f1 80  |..[.QR2.3.......|
000003f0  ca 30 88 14 4e 33 d2 3d  0a 00 73 f1 3c 00 74 04  |.0..N3.=..s.<.t.|
00000400  0c 30 88 04 5a 59 c3 50  b4 09 cd 21 58 c3 b8 00  |.0..ZY.P...!X...|
00000410  f0 8e c0 26 a0 fe ff ba  03 01 e8 ea ff 3c ff 74  |...&.........<.t|
00000420  1f 3c fe 74 21 3c fb 74  1d 3c fc 74 1f 3c fa 74  |.<.t!<.t.<.t.<.t|
00000430  21 3c f8 74 23 3c fd 74  25 3c f9 74 27 eb 2b 90  |!<.t#<.t%<.t'.+.|
00000440  ba 0d 01 eb 2b 90 ba 12  01 eb 25 90 ba 1a 01 eb  |....+.....%.....|
00000450  1f 90 ba 1f 01 eb 19 90  ba 2e 01 eb 13 90 ba 3d  |...............=|
00000460  01 eb 0d 90 ba 44 01 eb  07 90 ba 55 01 eb 01 90  |.....D.....U....|
00000470  e8 94 ff c3 b4 30 cd 21  be 74 01 8a ec e8 64 ff  |.....0.!.t....d.|
00000480  8a c5 83 c6 03 e8 5c ff  ba 62 01 e8 79 ff ba 73  |......\..b..y..s|
00000490  01 e8 73 ff 8a c7 e8 22  ff be 8a 01 89 04 ba 7b  |..s...."......{|
000004a0  01 e8 63 ff ba 8a 01 e8  5d ff 8a c3 e8 35 ff bf  |..c.....]....5..|
000004b0  a5 01 89 05 8b c1 83 c7  05 e8 10 ff ba 8f 01 e8  |................|
000004c0  45 ff ba a5 01 e8 3f ff  c3 e8 a8 ff 32 c0 b4 4c  |E.....?.....2..L|
000004d0  cd 21 c3                                          |.!.|
000004d3
```

Рис.2. Файл плохого .exe модуля

Что располагается с адреса 0?

С адреса 0h располагается управляющая информация для загрузчика, которая содержит заголовок и таблицу настройки адресов.

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

Каждый сегмент имеет свое собственное пространство, а также правильный EXE-файл не содержит директивы ORG 100h (которая выделяет память под PSP), поэтому код начинается с адреса 200h.

До этого адреса вставляется структура заголовка, содержащая следующие поля:

Offset		Field	Size	Description
0	0x00	Signature	word	0x5A4D (ASCII for 'M' and 'Z')
2	0x02	Extra bytes	word	Number of bytes in the last page.
4	0x04	Pages	word	Number of whole/partial pages.
6	0x06	Relocation items	word	Number of entries in the relocation table.
8	0x08	Header size	word	The number of paragraphs taken up by the header.
10	0x0A	Minimum allocation	word	The number of paragraphs **required** by the program, excluding the PSP and program image. If no free block is big enough, the loading stops.
12	0x0C	Maximum allocation	word	The number of paragraphs **requested** by the program. If no free block is big enough, the biggest one possible is allocated.
14	0x0E	Initial SS	word	Relocatable segment address for SS.
16	0x10	Initial SP	word	Initial value for SP.
18	0x12	Checksum	word	When added to the sum of all other words in the file, the result should be zero.
20	0x14	Initial IP	word	Initial value for IP.
22	0x16	Initial CS	word	Relocatable segment address for CS.
24	0x18	Relocation table	word	The (absolute) offset to the relocation table.
26	0x1A	Overlay	word	Value used for overlay management. If zero, this is the main executable.
28	0x1C	Overlay information	N/A	Files sometimes contain extra information for the main's program overlay management.

Рис.3. Структура заголовка, занимающая адреса с 0 до 200h в правильном .exe файле

```
yudjin@yudjin-desktop:~/Documents/study/2kurs/4sem/OS/MASM$ hexdump -C LAB1_EXE.EXE
00000000  4d 5a e2 00 03 00 01 00  20 00 00 00 ff ff 00 00  |MZ..... .......|
00000010  00 00 f7 6b 00 01 0b 00  1e 00 00 00 01 00 21 02  |...k..........!.|
00000020  0b 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  50 43 20 74 79 70 65 3a  20 24 50 43 0d 0a 24 50  |PC type: $PC..$P|
00000210  43 2f 58 54 0d 0a 24 41  54 0d 0a 24 50 53 32 20  |C/XT..$AT..$PS2 |
00000220  6d 6f 64 65 6c 20 33 30  0d 0a 24 50 53 32 20 6d  |model 30..$PS2 m|
00000230  6f 64 65 6c 20 38 30 0d  0a 24 50 43 6a 72 0d 0a  |odel 80..$PCjr..|
00000240  24 50 43 20 43 6f 6e 76  65 72 74 69 62 6c 65 0d  |$PC Convertible.|
00000250  0a 24 50 43 20 75 6e 6b  6e 6f 77 6e 0d 0a 24 4d  |.$PC unknown..$M|
00000260  53 44 4f 53 20 20 76 65  72 73 69 6f 6e 3a 20 24  |SDOS  version: $|
00000270  20 20 2e 20 20 0d 0a 24  4f 45 4d 20 20 76 65 72  |  .  ..$OEM  ver|
00000280  73 69 6f 6e 3a 20 24 20  20 0d 0a 24 55 73 65 72  |sion: $  ..$User|
00000290  20 73 65 72 69 61 6c 20  6e 75 6d 62 65 72 20 3a  | serial number :|
000002a0  20 24 20 20 20 20 20 20  20 20 20 0d 0a 24 00 00  | $         ..$..|
000002b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000003b0  e9 19 01 24 0f 3c 09 76  02 04 07 04 30 c3 51 8a  |...$.<.v....0.Q.|
000003c0  e0 e8 ef ff 86 c4 b1 04  d2 e8 e8 e6 ff 59 c3 53  |.............Y.S|
000003d0  8a fc e8 e9 ff 88 25 4f  88 05 4f 8a c7 e8 de ff  |......%O..O.....|
000003e0  88 25 4f 88 05 5b c3 51  52 32 e4 33 d2 b9 0a 00  |.%O..[.QR2.3....|
000003f0  f7 f1 80 ca 30 88 14 4e  33 d2 3d 0a 00 73 f1 3c  |....0..N3.=..s.<|
00000400  00 74 04 0c 30 88 04 5a  59 c3 50 b4 09 cd 21 58  |.t..0..ZY.P...!X|
00000410  c3 b8 00 f0 8e c0 26 a0  fe ff ba 00 00 e8 ea ff  |......&.........|
00000420  3c ff 74 1f 3c fe 74 21  3c fb 74 1d 3c fc 74 1f  |<.t.<.t!<.t.<.t.|
00000430  3c fa 74 21 3c f8 74 23  3c fd 74 25 3c f9 74 27  |<.t!<.t#<.t%<.t'|
00000440  eb 2b 90 ba 0a 00 eb 2b  90 ba 0f 00 eb 25 90 ba  |.+.....+.....%..|
00000450  17 00 eb 1f 90 ba 1c 00  eb 19 90 ba 2b 00 eb 13  |............+...|
00000460  90 ba 3a 00 eb 0d 90 ba  41 00 eb 07 90 ba 52 00  |..:.....A.....R.|
00000470  eb 01 90 e8 94 ff c3 b4  30 cd 21 be 71 00 8a ec  |........0.!.q...|
00000480  e8 64 ff 8a c5 83 c6 03  e8 5c ff ba 5f 00 e8 79  |.d.......\.._..y|
00000490  ff ba 70 00 e8 73 ff 8a  c7 e8 22 ff bf 87 00 89  |..p..s....".....|
000004a0  05 ba 78 00 e8 63 ff ba  87 00 e8 5d ff 8a c3 e8  |..x..c.....]....|
000004b0  35 ff bf a2 00 89 05 8b  c1 83 c7 05 e8 10 ff ba  |5...............|
000004c0  8c 00 e8 45 ff ba a2 00  e8 3f ff c3 1e 2b c0 50  |...E.....?...+.P|
000004d0  b8 00 00 8e d8 e8 39 ff  e8 9c ff 32 c0 b4 4c cd  |......9....2..L.|
000004e0  21 c3                                             |!.|
000004e2
```

Рис.4. Файл правильного .exe модуля

Загрузка COM модуля в основную память

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

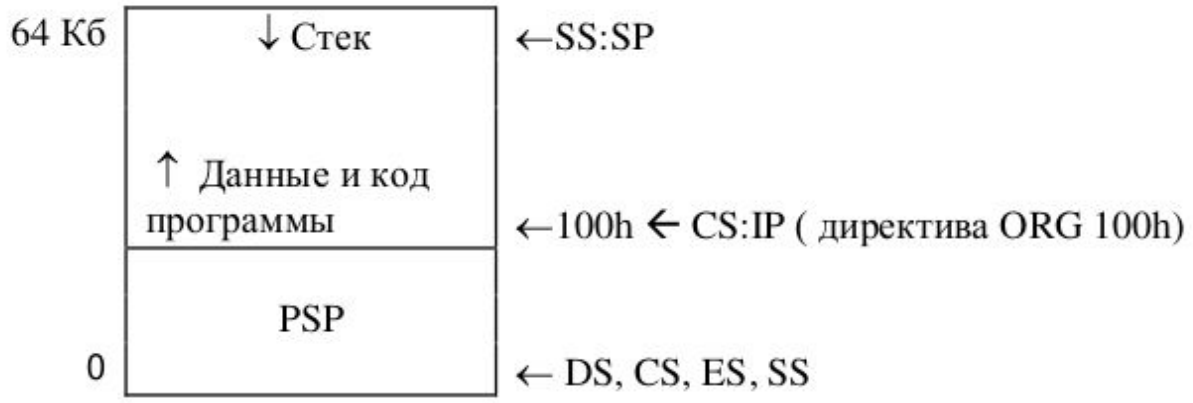

Рис.5.Структура размещения в памяти файла типа COM

Как видно из рис.5., код располагается после PSP: по адресу 100h

2) Что располагается с адреса 0?

PSP - префикс программного сегмента.

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

DS, CS, ES, SS - все указывают на начало PSP.

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

Через Stack Pointer внутри сегмента SS, т.е. внутри единственного, доступного .com программе сегмента. Стек заполняется с конца модуля, и до начала. Т.е. с адреса SS:FFFFh до SS:0000h.

Загрузка «хорошего» EXE модуля в основную память

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

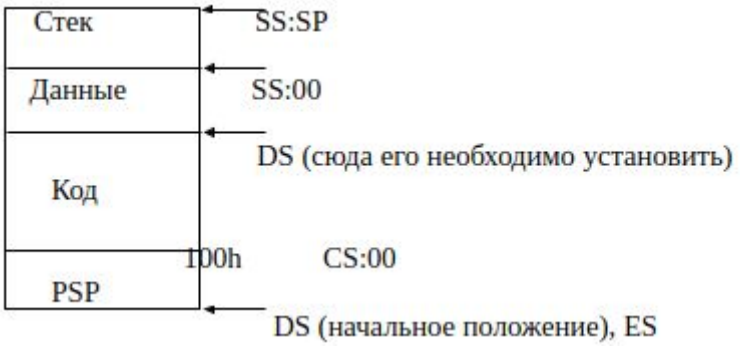

Рис.6. Структура размещения в памяти файла типа EXE

Схема загрузки хорошего .exe файла показана на рисунке 5. Регистры DS и ES указывают на начало PSP, CS - начало кода, SS - начало стека.

2)На что указывают регистры DS и ES?

Регистры DS и ES указывают на начало PSP

3)Как определяется стек?

Регистр SS указывает на начало сегмента стека, а SS:SP – на конец сегмента стека.

4)Как определяется точка входа?

С помощью директивы END, где параметр - метка на начало команд.

Заключение.

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

Приложение А. Исходный код.

Файл lab1_com.asm.

```
LAB1 SEGMENT
          ASSUME CS:LAB1, DS:LAB1, ES:NOTHING, SS:NOTHING
          ORG 100H
START:    JMP  BEGIN

PC_INFO     db 'PC type: ', '$'
PC          db 'PC',13,10,'$'
PC_XT       db 'PC/XT',13,10,'$'
AT          db 'AT',13,10,'$'
PS2_30      db 'PS2 model 30',13,10,'$'
PS2_80      db 'PS2 model 80',13,10,'$'
PC_jr       db 'PCjr',13,10,'$'
PC_Convertible db 'PC Convertible',13,10,'$'
PC_Unknown  db 'PC unknown',13,10,'$'

MSDOS_VERSION_INFO db 'MSDOS  version: ', '$'
MSDOS_VERSION DB '  .  ', 13, 10, '$'
OEM_VERSION_INFO   db 'OEM  version: ', '$'
OEM_VERSION   db '  ',13,10,'$'
USER_NUMBER_INFO   db 'User serial number : $'
USER_NUMBER   db '        ',13,10,'$'

;ПРОЦЕДУРЫ
;-----------------------------------------------------------------
---------------
; Перевод тетрады (4-ех младших байтов AL) в 16-ичную СС и ее
представление в виде символа
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

; Перевод байта AL в 16-ичную СС и его представление в виде
символов
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

; Перевод слова AX в 16-ичную СС и его представление в виде
символов
WORD_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
```

```
WORD_TO_HEX ENDP

; Перевод байта AL в 10-ичную СС и его представление в виде
символов
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

; Вызывает функцию вывода строки на экран
PRINT PROC NEAR
     push ax
     mov ah, 09h
     int 21h
     pop ax
     ret
```

```
PRINT ENDP

;-------------------------------------------------------------------
---------------


PRINT_PC_TYPE PROC NEAR
    mov ax, 0F000h
     mov es, ax
     mov al, es:[0FFFEh]
    mov dx, offset PC_INFO
    call PRINT
    cmp al, 0ffh
    je pc_msg
    cmp al, 0feh
    je pc_xt_msg
    cmp al, 0fbh
    je pc_xt_msg
    cmp al, 0fch
    je at_msg
    cmp al, 0fah
    je ps2_30_msg
    cmp al, 0f8h
    je ps2_80_msg
    cmp al, 0fdh
    je pcjr_msg
    cmp al, 0f9h
    je pc_conv_msg
    jmp pc_undef_msg
pc_msg:
    mov dx, offset PC
    jmp print_call
pc_xt_msg:
    mov dx, offset PC_XT
    jmp print_call
```

```
at_msg:
    mov dx, offset AT
    jmp print_call
ps2_30_msg:
    mov dx, offset PS2_30
    jmp print_call
ps2_80_msg:
    mov dx, offset PS2_80
    jmp print_call
pcjr_msg:
    mov dx, offset PC_jr
    jmp print_call
pc_conv_msg:
    mov dx, offset PC_Convertible
    jmp print_call
pc_undef_msg:
    mov dx, offset PC_Unknown
    jmp print_call
print_call:
    call PRINT
    ret
PRINT_PC_TYPE ENDP


PRINT_MSDOS_VERSION PROC NEAR
    mov ah, 30h
    int 21h

    mov si, offset MSDOS_VERSION + 1
    mov ch, ah
    call BYTE_TO_DEC
    mov al, ch
    add si, 3
    call BYTE_TO_DEC
    mov dx, offset MSDOS_VERSION_INFO
```

```
    call PRINT
    mov dx, offset MSDOS_VERSION
    call PRINT

    mov al, bh
    call BYTE_TO_HEX
    mov si, offset OEM_VERSION
    mov [si], ax
    mov dx, offset OEM_VERSION_INFO
    call PRINT
    mov dx, offset OEM_VERSION
    call PRINT

    mov al, bl
    call BYTE_TO_DEC
    mov di, offset USER_NUMBER
    mov [di], ax
    mov ax, cx
    add di,5
    call WORD_TO_HEX
    mov dx, offset USER_NUMBER_INFO
    call PRINT
    mov dx, offset USER_NUMBER
    call PRINT

    ret
PRINT_MSDOS_VERSION ENDP


BEGIN:

        ;call PRINT_PC_TYPE
        call PRINT_MSDOS_VERSION
```

```
          xor       al, al
          mov  ah, 4ch
          int       21h
          ret

LAB1      ENDS
          END       START
```

Файл lab1_exe.asm.

```
AStack    SEGMENT  STACK
AStack    ENDS

DATA SEGMENT
    PC_INFO     db 'PC type: ', '$'
    PC          db 'PC',13,10,'$'
    PC_XT       db 'PC/XT',13,10,'$'
    AT          db 'AT',13,10,'$'
    PS2_30      db 'PS2 model 30',13,10,'$'
    PS2_80      db 'PS2 model 80',13,10,'$'
    PC_jr       db 'PCjr',13,10,'$'
    PC_Convertible db 'PC Convertible',13,10,'$'
    PC_Unknown  db 'PC unknown',13,10,'$'

    MSDOS_VERSION_INFO db 'MSDOS  version: ', '$'
    MSDOS_VERSION DB '  .  ', 13, 10, '$'
    OEM_VERSION_INFO   db 'OEM  version: ', '$'
    OEM_VERSION   db '  ',13,10,'$'
    USER_NUMBER_INFO   db 'User serial number : $'
    USER_NUMBER   db '        ',13,10,'$'
DATA ENDS

CODE SEGMENT
          ASSUME CS:CODE, DS:DATA, SS:AStack
          ORG 100H
```

```
START:    JMP  BEGIN

;ПРОЦЕДУРЫ
;-----------------------------------------------------------------
---------------
; Перевод тетрады (4-ех младших байтов AL) в 16-ичную СС и ее
представление в виде символа
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

; Перевод байта AL в 16-ичную СС и его представление в виде
символов
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

; Перевод слова AX в 16-ичную СС и его представление в виде
символов
WORD_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
WORD_TO_HEX ENDP

; Перевод байта AL в 10-ичную СС и его представление в виде
символов
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

; Вызывает функцию вывода строки на экран
PRINT PROC NEAR
     push ax
     mov ah, 09h
     int 21h
     pop ax
     ret
PRINT ENDP

;-------------------------------------------------------------------
---------------


PRINT_PC_TYPE PROC NEAR
    mov ax, 0F000h
     mov es, ax
     mov al, es:[0FFFEh]
    mov dx, offset PC_INFO
    call PRINT
    cmp al, 0ffh
    je pc_msg
    cmp al, 0feh
    je pc_xt_msg
    cmp al, 0fbh
    je pc_xt_msg
    cmp al, 0fch
    je at_msg
    cmp al, 0fah
    je ps2_30_msg
```

```
    cmp al, 0f8h
    je ps2_80_msg
    cmp al, 0fdh
    je pcjr_msg
    cmp al, 0f9h
    je pc_conv_msg
    jmp pc_undef_msg
pc_msg:
    mov dx, offset PC
    jmp print_call
pc_xt_msg:
    mov dx, offset PC_XT
    jmp print_call
at_msg:
    mov dx, offset AT
    jmp print_call
ps2_30_msg:
    mov dx, offset PS2_30
    jmp print_call
ps2_80_msg:
    mov dx, offset PS2_80
    jmp print_call
pcjr_msg:
    mov dx, offset PC_jr
    jmp print_call
pc_conv_msg:
    mov dx, offset PC_Convertible
    jmp print_call
pc_undef_msg:
    mov dx, offset PC_Unknown
    jmp print_call
print_call:
    call PRINT
    ret
PRINT_PC_TYPE ENDP
```

```
PRINT_MSDOS_VERSION PROC NEAR
    mov ah, 30h
    int 21h

    mov si, offset MSDOS_VERSION + 1
    mov ch, ah
    call BYTE_TO_DEC
    mov al, ch
    add si, 3
    call BYTE_TO_DEC
    mov dx, offset MSDOS_VERSION_INFO
    call PRINT
    mov dx, offset MSDOS_VERSION
    call PRINT

    mov al, bh
    call BYTE_TO_HEX
    mov di, offset OEM_VERSION
    mov [di], ax
    mov dx, offset OEM_VERSION_INFO
    call PRINT
    mov dx, offset OEM_VERSION
    call PRINT

    mov al, bl
    call BYTE_TO_DEC
    mov di, offset USER_NUMBER
    mov [di], ax
    mov ax, cx
    add di,5
    call WORD_TO_HEX
    mov dx, offset USER_NUMBER_INFO
    call PRINT
```

```
    mov dx, offset USER_NUMBER
    call PRINT

    ret
PRINT_MSDOS_VERSION ENDP


BEGIN:
     push  ds
    sub   ax, ax
    push  ax
    mov   ax, DATA
    mov   ds, ax
     call PRINT_PC_TYPE
     call PRINT_MSDOS_VERSION

     xor       al, al
     mov  ah, 4ch
     int       21h
     ret

CODE      ENDS
          END       START
```