

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра МОЭВМ**

**ОТЧЕТ**  
**по лабораторной работе №1**  
**по дисциплине «Операционные системы»**  
**Тема: Исследование структур загрузочных модулей**

Студент гр. 9383

\_\_\_\_\_

Арутюнян С.Н.

Преподаватель

\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2021

## Постановка задачи

Шаг 1. Написать текст исходного .COM модуля, который определяет тип РС и версию системы. Результатом выполнения этого шага будет «хороший» .COM модуль, а также необходимо построить «плохой» .EXE, полученный из исходного текста для .COM.

Шаг 2. Написать текст исходного .EXE модуля, который выполняет те же функции, что и модуль в шаге 1 и отладить его.

Шаг 3. Сравнить исходные тексты для .COM и .EXE модулей. Ответить на вопросы «Отличия исходных текстов COM и EXE программ».

Шаг 4. Запустить FAR и открыть файл загрузочного модуля .COM и файл «плохого» .EXE в шестнадцатеричном виде. Затем открыть файл загрузочного модуля «хорошего» .EXE и сравнить его с предыдущими файлами. Ответить на контрольные вопросы «Отличия форматов файлов COM и EXE модулей».

Шаг 5. Открыть отладчик TD.EXE и загрузить .CO. Ответить на контрольные вопросы «Загрузка COM модуля в основную память». Представить в отчете план загрузки модуля .COM в основную память.

Шаг 6. Открыть отладчик TD.EXE и загрузить «хороший» .EXE. Ответить на контрольные вопросы «Загрузка «хорошего» EXE в основную память».

Шаг 7. Оформить отчет в соответствии с требованиями. Привести скриншоты. Для файлов их вид в шестнадцатеричном виде, для загрузочных модулей — в отладчике.

Таблица 1. Используемые процедуры

Процедура	Описание
TETR_TO_HEX	Перевод десятичной цифры в код символа ASCII
BYTE_TO_HEX	Перевод байта в AL в два символа шестн. числа AX
BYTE_TO_WRD	Запись AL в строку DEC_NUMBER
PRINT_PC_TYPE	Вывод типа ПК на экран
PRINT_DOS_VERSION	Вывод версии DOS на экран

## Выполнение шагов

**Шаг 1.** В процессе выполнения лабораторной работы была разработана программа для модуля .COM, исходный код которой приведен в приложении А. После ассемблирования и линковки был получен «плохой» .EXE модуль, а из него был получен «хороший» .COM модуль.

```
C:\>lab1.com
The type of your PC is: AT
Your DOS version: 5.2
Your OEM number: 2
Your serial number: 0A0000
```

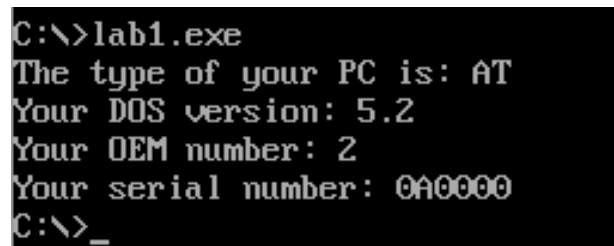
Рис. 1. Пример работы модуля .COM

Также, был создан «плохой» .EXE. Как и ожидалось, работает он неверно:

```
C:\>lab1_com.exe
The type of your PC is: PC
5.2your PC is: PS2 model 30
is: PC/XT
2your PC is: PS2 model 30
ype of your PC is: AT20A0000of your PC is: PS2 model 30
C:\>
```

Рис. 2. Пример работы «плохого» .EXE

**Шаг 2.** В процессе выполнения лабораторной работы была разработана программа для «хорошего» модуля .EXE, исходный код которой приведен в приложении Б.



```
C:\>lab1.exe
The type of your PC is: AT
Your DOS version: 5.2
Your OEM number: 2
Your serial number: 0A00000
C:\>_
```

Рис. 3. Пример работы «хорошего» .EXE

**Шаг 3.** Ответим на вопросы по теме «Отличия исходных текстов .EXE и .COM программ».

1. Сколько сегментов должна содержать COM-программа?

Такая программа должна содержать только один сегмент, т. к. код и данные находятся в одном сегменте, а стек автоматически устанавливается на последнюю ячейку сегмента.

2. EXE-программа?

Такая программа должна содержать по крайней мере один сегмент, при этом допускается использовать больше 3 сегментов. Сегменты кода, данных и стека описываются отдельно друг от друга.

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

В COM программе обязательно должна быть прописана директива `org 100h`, т. к. первые 100h байт занимает PSP. Это позволит сместить все относительные адреса на 100h байт. Также необходимо прописать (`ASSUME CS:MYSEGMENT, DS:MYSEGMENT, ES:NOTHING, SS:NOTHING`), чтобы сегмент данных и сегмент кода указывали на один и тот же сегмент.

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

Нельзя использовать команды вида «`mov register, seg segname`». Причина тому — отсутствие таблицы настроен в COM-программе.

#### Шаг 4. Шестнадцатеричное представление модуля .COM:

```

00000000 e9 0e 03 54 68 65 20 74 79 70 65 20 6f 66 20 79 |...The type of y|
00000010 6f 75 72 20 50 43 20 69 73 3a 20 50 43 0d 0a 24 |our PC is: PC..$|
00000020 54 68 65 20 74 79 70 65 20 6f 66 20 79 6f 75 72 |The type of your|
00000030 20 50 43 20 69 73 3a 20 50 43 2f 58 54 0d 0a 24 | PC is: PC/XT..$|
00000040 54 68 65 20 74 79 70 65 20 6f 66 20 79 6f 75 72 |The type of your|
00000050 20 50 43 20 69 73 3a 20 41 54 0d 0a 24 54 68 65 | PC is: AT..$The|
00000060 20 74 79 70 65 20 6f 66 20 79 6f 75 72 20 50 43 | type of your PC|
00000070 20 69 73 3a 20 50 53 32 20 6d 6f 64 65 6c 20 33 | is: PS2 model 3|
00000080 30 0d 0a 24 54 68 65 20 74 79 70 65 20 6f 66 20 |0..$The type of |
00000090 79 6f 75 72 20 50 43 20 69 73 3a 20 50 53 32 20 |your PC is: PS2 |
000000a0 6d 6f 64 65 6c 20 35 30 20 6f 72 20 36 30 0d 0a |model 50 or 60..|
000000b0 24 54 68 65 20 74 79 70 65 20 6f 66 20 79 6f 75 |$The type of you|
000000c0 72 20 50 43 20 69 73 3a 20 50 53 32 20 6d 6f 64 |r PC is: PS2 mod|
000000d0 65 6c 20 38 30 0d 0a 24 54 68 65 20 74 79 70 65 |el 80..$The type|
000000e0 20 6f 66 20 79 6f 75 72 20 50 43 20 69 73 3a 20 | of your PC is: |
000000f0 50 43 6a 72 0d 0a 24 54 68 65 20 74 79 70 65 20 |PCjr..$The type |
00000100 6f 66 20 79 6f 75 72 20 50 43 20 69 73 3a 20 50 |of your PC is: P|
00000110 43 20 43 6f 6e 76 65 72 74 69 62 6c 65 0d 0a 24 |C Convertible..$|
00000120 59 6f 75 72 20 44 4f 53 20 76 65 72 73 69 6f 6e |Your DOS version|
00000130 3a 20 24 59 6f 75 72 20 4f 45 4d 20 6e 75 6d 62 |: $Your OEM numb|
00000140 65 72 3a 20 24 59 6f 75 72 20 73 65 72 69 61 6c |er: $Your serial|
00000150 20 6e 75 6d 62 65 72 3a 20 24 20 20 20 20 20 24 | number: $ $|
00000160 20 20 20 20 20 20 20 20 24 0d 0a 24 45 52 52 4f | $..$ERRO|
00000170 52 20 42 55 54 20 49 4d 50 4c 45 4d 45 4e 54 20 |R BUT IMPLEMENT |
00000180 45 52 52 4f 52 20 48 41 4e 44 4c 45 52 21 0d 0a |ERROR HANDLER!..|
00000190 24 24 0f 3c 09 76 02 04 07 04 30 c3 51 8a e0 e8 |$$.<.v....0.Q...|
000001a0 ef ff 86 c4 b1 04 d2 e8 e8 e6 ff 59 c3 53 8a fc |.....Y.S..|
000001b0 e8 e9 ff 88 25 4f 88 05 4f 8a c7 e8 de ff 88 25 |....%0..0.....%|
000001c0 4f 88 05 5b c3 51 52 32 e4 33 d2 b9 0a 00 f7 f1 |0..[.QR2.3.....|
000001d0 80 ca 30 88 14 4e 33 d2 3d 0a 00 73 f1 3c 00 74 |..0..N3.=..s.<.t|
000001e0 04 0c 30 88 04 5a 59 c3 50 51 52 57 b9 00 00 ba |..0..ZY.PQRW....|
000001f0 00 00 bf 5a 02 3d 00 00 74 0b bb 0a 00 f7 f3 52 |...Z.=..t.....R|
00000200 41 33 d2 eb f0 83 f9 00 74 09 5a 83 c2 30 89 15 |A3.....t.Z..0..|
00000210 49 eb f2 47 c6 05 24 5f 5a 59 58 c3 50 52 ba 69 |I..G..$_ZYX.PR.i|
00000220 02 b4 09 cd 21 5a 58 c3 b8 00 f0 8e c0 bf fe ff |....!ZX.....|
00000230 26 8a 1d 80 fb ff 74 2e 80 fb fe 74 2f 80 fb fb |&.....t....t/...|
00000240 74 2a 80 fb fc 74 2b 80 fb fa 74 2c 80 fb fc 74 |t*...t+...t,...t|
00000250 2d 80 fb f8 74 2e 80 fb fd 74 2f 80 fb fd 74 30 |-...t....t/...t0|
00000260 ba 6c 02 eb 31 90 ba 03 01 eb 2b 90 ba 20 01 eb |.l..1.....+.. ..|
00000270 25 90 ba 40 01 eb 1f 90 ba 5d 01 eb 19 90 ba 84 |%..@.....].....|
00000280 01 eb 13 90 ba b1 01 eb 0d 90 ba d8 01 eb 07 90 |.....|
00000290 ba f7 01 eb 01 90 b4 09 cd 21 c3 50 53 51 52 ba |.....!.PSQR.|
000002a0 20 02 b4 09 cd 21 b4 30 cd 21 25 ff 00 e8 38 ff | ....!.0.!%...8.|
000002b0 ba 5a 02 b4 09 cd 21 b2 2e b4 02 cd 21 25 00 ff |.Z....!.....!%..|
000002c0 e8 25 ff ba 5a 02 b4 09 cd 21 e8 4f ff ba 33 02 |.%..Z....!.0..3.|
000002d0 b4 09 cd 21 53 81 e3 00 ff 8b c3 e8 0a ff ba 5a |...!S.....Z|
000002e0 02 b4 09 cd 21 5b e8 33 ff ba 45 02 b4 09 cd 21 |....![.3..E....!|
000002f0 bf 60 02 83 c7 05 8b c1 e8 b2 fe 8a c3 e8 9c fe |.`.....|
00000300 83 ef 02 89 05 ba 60 02 b4 09 cd 21 5a 59 5b 58 |.....`....!ZY[X|
00000310 c3 e8 14 ff e8 84 ff 32 c0 b4 4c cd 21 |.....2..L.!|
0000031d

```

Рис. 4. Шестнадцатеричное представление модуля .COM

## Шестнадцатеричное представление «плохого» модуля .EXE:

```

00000000  4d 5a 1d 00 04 00 00 00 20 00 00 00 ff ff 00 00 |MZ.....|
00000010  00 00 02 27 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 0e 03 54 68 65 20 74 79 70 65 20 6f 66 20 79 |...The type of y|
00000310  6f 75 72 20 50 43 20 69 73 3a 20 50 43 0d 0a 24 |our PC is: PC..$|
00000320  54 68 65 20 74 79 70 65 20 6f 66 20 79 6f 75 72 |The type of your|
00000330  20 50 43 20 69 73 3a 20 50 43 2f 58 54 0d 0a 24 | PC is: PC/XT..$|
00000340  54 68 65 20 74 79 70 65 20 6f 66 20 79 6f 75 72 |The type of your|
00000350  20 50 43 20 69 73 3a 20 41 54 0d 0a 24 54 68 65 | PC is: AT..$The|
00000360  20 74 79 70 65 20 6f 66 20 79 6f 75 72 20 50 43 | type of your PC|
00000370  20 69 73 3a 20 50 53 32 20 6d 6f 64 65 6c 20 33 | is: PS2 model 3|
00000380  30 0d 0a 24 54 68 65 20 74 79 70 65 20 6f 66 20 |0..$The type of |
00000390  79 6f 75 72 20 50 43 20 69 73 3a 20 50 53 32 20 |your PC is: PS2 |
000003a0  6d 6f 64 65 6c 20 35 30 20 6f 72 20 36 30 0d 0a |model 50 or 60..|
000003b0  24 54 68 65 20 74 79 70 65 20 6f 66 20 79 6f 75 |$The type of you|
000003c0  72 20 50 43 20 69 73 3a 20 50 53 32 20 6d 6f 64 |r PC is: PS2 mod|
000003d0  65 6c 20 38 30 0d 0a 24 54 68 65 20 74 79 70 65 |el 80..$The type|
000003e0  20 6f 66 20 79 6f 75 72 20 50 43 20 69 73 3a 20 | of your PC is: |
000003f0  50 43 6a 72 0d 0a 24 54 68 65 20 74 79 70 65 20 |PCjr..$The type |
00000400  6f 66 20 79 6f 75 72 20 50 43 20 69 73 3a 20 50 |of your PC is: P|
00000410  43 20 43 6f 6e 76 65 72 74 69 62 6c 65 0d 0a 24 |C Convertible..$|
00000420  59 6f 75 72 20 44 4f 53 20 76 65 72 73 69 6f 6e |Your DOS version|
00000430  3a 20 24 59 6f 75 72 20 4f 45 4d 20 6e 75 6d 62 |: $Your OEM numb|
00000440  65 72 3a 20 24 59 6f 75 72 20 73 65 72 69 61 6c |er: $Your serial|
00000450  20 6e 75 6d 62 65 72 3a 20 24 20 20 20 20 20 24 | number: $ $|
00000460  20 20 20 20 20 20 20 20 24 0d 0a 24 45 52 52 4f | $..$ERRO|
00000470  52 20 42 55 54 20 49 4d 50 4c 45 4d 45 4e 54 20 |R BUT IMPLEMENT|
00000480  45 52 52 4f 52 20 48 41 4e 44 4c 45 52 21 0d 0a |ERROR HANDLER!..|
00000490  24 24 0f 3c 09 76 02 04 07 04 30 c3 51 8a e0 e8 |$$.<.v....0.Q...|
000004a0  ef ff 86 c4 b1 04 d2 e8 e8 e6 ff 59 c3 53 8a fc |.....Y.S..|
000004b0  e8 e9 ff 88 25 4f 88 05 4f 8a c7 e8 de ff 88 25 |....%0..0.....%|
000004c0  4f 88 05 5b c3 51 52 32 e4 33 d2 b9 0a 00 f7 f1 |0..[.QR2.3.....|
000004d0  80 ca 30 88 14 4e 33 d2 3d 0a 00 73 f1 3c 00 74 |..0..N3.=..s.<.t|
000004e0  04 0c 30 88 04 5a 59 c3 50 51 52 57 b9 00 00 ba |..0..ZY.PQRW...|
000004f0  00 00 bf 5a 02 3d 00 00 74 0b bb 0a 00 f7 f3 52 |...Z.=..t.....R|
00000500  41 33 d2 eb f0 83 f9 00 74 09 5a 83 c2 30 89 15 |A3.....t.Z..0..|
00000510  49 eb f2 47 c6 05 24 5f 5a 59 58 c3 50 52 ba 69 |I..G..$_ZYX.PR.i|
00000520  02 b4 09 cd 21 5a 58 c3 b8 00 f0 8e c0 bf fe ff |....!ZX.....|
00000530  26 8a 1d 80 fb ff 74 2e 80 fb fe 74 2f 80 fb fb |&....t....t/...|
00000540  74 2a 80 fb fc 74 2b 80 fb fa 74 2c 80 fb fc 74 |t*...t+...t,...t|
00000550  2d 80 fb f8 74 2e 80 fb fd 74 2f 80 fb fd 74 30 |-...t....t/...t0|
00000560  ba 6c 02 eb 31 90 ba 03 01 eb 2b 90 ba 20 01 eb |.l..1.....+.. ..|
00000570  25 90 ba 40 01 eb 1f 90 ba 5d 01 eb 19 90 ba 84 |%..@.....].....|
00000580  01 eb 13 90 ba b1 01 eb 0d 90 ba d8 01 eb 07 90 |.....|
00000590  ba f7 01 eb 01 90 b4 09 cd 21 c3 50 53 51 52 ba |.....!..PSQR..|
000005a0  20 02 b4 09 cd 21 b4 30 cd 21 25 ff 00 e8 38 ff | ....!.0.!%...8.|
000005b0  ba 5a 02 b4 09 cd 21 b2 2e b4 02 cd 21 25 00 ff |.Z....!.....!%..|
000005c0  e8 25 ff ba 5a 02 b4 09 cd 21 e8 4f ff ba 33 02 |.%..Z....!.0..3.|
000005d0  b4 09 cd 21 53 81 e3 00 ff 8b c3 e8 0a ff ba 5a |...!S.....Z|
000005e0  02 b4 09 cd 21 5b e8 33 ff ba 45 02 b4 09 cd 21 |....![.3..E....!|
000005f0  bf 60 02 83 c7 05 8b c1 e8 b2 fe 8a c3 e8 9c fe |.`.....|
00000600  83 ef 02 89 05 ba 60 02 b4 09 cd 21 5a 59 5b 58 |.....`.....!ZY[X|
00000610  c3 e8 14 ff e8 84 ff 32 c0 b4 4c cd 21 |.....2..L.!|
0000061d

```

Рис. 5. Шестнадцатеричное представление «плохого» модуля .EXE



00000000	4d 5a fe 00 04 00 01 00	20 00 00 00 ff ff 00 00	MZ.....
00000010	00 02 a3 b2 5d 01 39 00	1e 00 00 00 01 00 5e 01	....].9.....^.
00000020	39 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	9.....
00000030	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
*			
00000400	54 68 65 20 74 79 70 65	20 6f 66 20 79 6f 75 72	The type of your
00000410	20 50 43 20 69 73 3a 20	50 43 0d 0a 24 54 68 65	PC is: PC..\$The
00000420	20 74 79 70 65 20 6f 66	20 79 6f 75 72 20 50 43	type of your PC
00000430	20 69 73 3a 20 50 43 2f	58 54 0d 0a 24 54 68 65	is: PC/XT..\$The
00000440	20 74 79 70 65 20 6f 66	20 79 6f 75 72 20 50 43	type of your PC
00000450	20 69 73 3a 20 41 54 0d	0a 24 54 68 65 20 74 79	is: AT..\$The ty
00000460	70 65 20 6f 66 20 79 6f	75 72 20 50 43 20 69 73	pe of your PC is
00000470	3a 20 50 53 32 20 6d 6f	64 65 6c 20 33 30 0d 0a	: PS2 model 30..
00000480	24 54 68 65 20 74 79 70	65 20 6f 66 20 79 6f 75	\$The type of you
00000490	72 20 50 43 20 69 73 3a	20 50 53 32 20 6d 6f 64	r PC is: PS2 mod
000004a0	65 6c 20 35 30 20 6f 72	20 36 30 0d 0a 24 54 68	el 50 or 60..\$Th
000004b0	65 20 74 79 70 65 20 6f	66 20 79 6f 75 72 20 50	e type of your P
000004c0	43 20 69 73 3a 20 50 53	32 20 6d 6f 64 65 6c 20	C is: PS2 model
000004d0	38 30 0d 0a 24 54 68 65	20 74 79 70 65 20 6f 66	80..\$The type of
000004e0	20 79 6f 75 72 20 50 43	20 69 73 3a 20 50 43 6a	your PC is: PCj
000004f0	72 0d 0a 24 54 68 65 20	74 79 70 65 20 6f 66 20	r..\$The type of
00000500	79 6f 75 72 20 50 43 20	69 73 3a 20 50 43 20 43	your PC is: PC C
00000510	6f 6e 76 65 72 74 69 62	6c 65 0d 0a 24 59 6f 75	onvertible..\$You
00000520	72 20 44 4f 53 20 76 65	72 73 69 6f 6e 3a 20 24	r DOS version: \$
00000530	59 6f 75 72 20 4f 45 4d	20 6e 75 6d 62 65 72 3a	Your OEM number:
00000540	20 24 59 6f 75 72 20 73	65 72 69 61 6c 20 6e 75	\$Your serial nu
00000550	6d 62 65 72 3a 20 24 20	20 20 20 20 24 20 20 20	mber: \$ \$
00000560	20 20 20 20 20 24 0d 0a	24 45 52 52 4f 52 20 42	\$..\$ERROR B
00000570	55 54 20 49 4d 50 4c 45	4d 45 4e 54 20 45 52 52	UT IMPLEMENT ERR
00000580	4f 52 20 48 41 4e 44 4c	45 52 21 0d 0a 24 00 00	OR HANDLER!..\$..
00000590	24 0f 3c 09 76 02 04 07	04 30 c3 51 8a e0 e8 ef	\$.<.v....0.Q....
000005a0	ff 86 c4 b1 04 d2 e8 e8	e6 ff 59 c3 53 8a fc e8	.....Y.S....
000005b0	e9 ff 88 25 4f 88 05 4f	8a c7 e8 de ff 88 25 4f	...%O..O.....%O
000005c0	88 05 5b c3 50 51 52 57	b9 00 00 ba 00 00 bf 57	..[.PQRW.....W
000005d0	01 3d 00 00 74 0b bb 0a	00 f7 f3 52 41 33 d2 eb	.=..t.....RA3..
000005e0	f0 83 f9 00 74 09 5a 83	c2 30 89 15 49 eb f2 47	....t.Z..0..I..G
000005f0	c6 05 24 5f 5a 59 58 c3	50 52 ba 66 01 b4 09 cd	..\$_ZYX.PR.f....
00000600	21 5a 58 c3 b8 00 f0 8e	c0 bf fe ff 26 8a 1d 80	!ZX.....&...
00000610	fb ff 74 2e 80 fb fe 74	2f 80 fb fb 74 2a 80 fb	..t....t/...t*..
00000620	fc 74 2b 80 fb fa 74 2c	80 fb fc 74 2d 80 fb f8	.t+...t,...t-...
00000630	74 2e 80 fb fd 74 2f 80	fb fd 74 30 ba 69 01 eb	t....t/...t0.i..
00000640	31 90 ba 00 00 eb 2b 90	ba 1d 00 eb 25 90 ba 3d	1.....+.....%..=
00000650	00 eb 1f 90 ba 5a 00 eb	19 90 ba 81 00 eb 13 90	.....Z.....
00000660	ba ae 00 eb 0d 90 ba d5	00 eb 07 90 ba f4 00 eb	.....
00000670	01 90 b4 09 cd 21 c3 50	53 51 52 ba 1d 01 b4 09	.....!.PSQR.....
00000680	cd 21 b4 30 cd 21 25 ff	00 e8 38 ff ba 57 01 b4	.!.0.!%...8..W..
00000690	09 cd 21 b2 2e b4 02 cd	21 25 00 ff e8 25 ff ba	..!.....!%...%..
000006a0	57 01 b4 09 cd 21 e8 4f	ff ba 30 01 b4 09 cd 21	W....!.O..0....!
000006b0	53 81 e3 00 ff 8b c3 e8	0a ff ba 57 01 b4 09 cd	S.....W....
000006c0	21 5b e8 33 ff ba 42 01	b4 09 cd 21 bf 5d 01 83	![.3..B.....!..
000006d0	c7 05 8b c1 e8 d5 fe 8a	c3 e8 bf fe 83 ef 02 89	.....
000006e0	05 ba 5d 01 b4 09 cd 21	5a 59 5b 58 c3 b8 20 00	..]....!ZY[X..
000006f0	8e d8 e8 0f ff e8 7f ff	32 c0 b4 4c cd 21	.....2..L.!!
000006fe			

Рис. 6. Шестнадцатеричное представление «плохого» модуля .EXE

Отличия .COM от «плохого» .EXE в том, что в .COM программа идет сразу, без всяких заголовков. В «плохом» .EXE есть заголовок, но он неправильный, потому что он создавался по .COM-программе.

Но в «хорошем» .EXE заголовок уже корректный, да и сам .EXE файл стал немного больше в размерах.

### **Отличия форматов файлов COM и EXE модулей**

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

В COM вся программа идет сразу же, без заголовков и чего угодно лишнего. Код располагается по адресу 100h от начала отведенного под программу сегмента.

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

«Плохой» EXE содержит EXE-маркер (байты MZ), заголовок и таблицу настроек. В заголовке содержится различная служебная информация для загрузчика, например, длина файла в блоках, длина заголовка, значение SP при входе в программу, значение IP, контрольная сумма и т.д. В таблице настроек сохранились записи в формате сегмент: смещение. К смещениям в загрузочном модуле, на которые указывают значения в таблице, после загрузки программы в память будет прибавлен сегментный адрес. Код и данные располагаются в одном сегменте. Код располагается с 300h байта (т. к. размер PSP — 100h, размер заголовка+таблицы настроек — 100h, смещение - 100h). С адреса 2 располагаются заголовок и таблица настроек после него.

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

В «хорошем» EXE код, данные и стек разделены на сегменты, а также EXE может иметь любой размер. С адреса 2 в «хорошем» EXE располагается заголовок, а после него идет таблица настроек. Код начинается с 400h, т. к. размер PSP — 100h, размер заголовка+таблицы — 100h, размер стека — 200h.



## **Шаг 5.**

### **Загрузка COM модуля в основную память**

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

Первым делом ОС ищет подходящее по размеру место в оперативной памяти для COM модуля. Далее она помещает в это место PSP, а по смещению в 100h помещает считанный с диска модуль.

2. Что располагается с адреса 0?

С адреса 0 располагается PSP размером в 100h байт.

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

CS, ES, DS и SS указывают на PSP (48DD), а SP указывает на конец сегмента (FFFE).

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

Стек определяется автоматически при ассемблировании и линковке. Стек, теоретически, занимает всю область сегмента, но если он начнет пересекаться с кодом, то последствия неопределены.

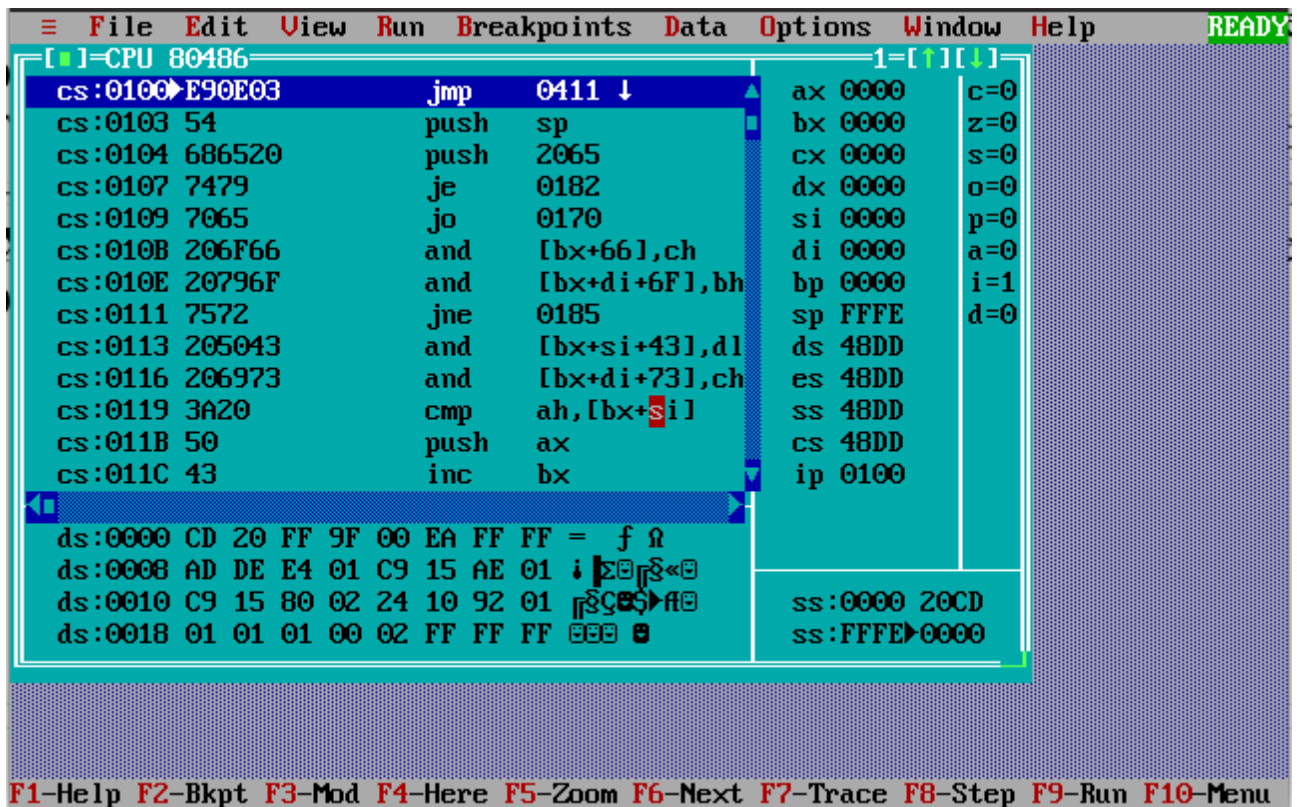


Рис. 7. Интерфейс TD.EXE (.COM)

## Шаг 6.

### Загрузка «хорошего» EXE модуля в основную память

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

Подготовка загрузки аналогична .COM. EXE загружается по смещению 100h от PSP. В процессе загрузки считывается информация EXE-заголовка и выполняется перемещение адресов сегментов. CS указывает на начало сегмента команд (4926h). В IP загружается смещение точки входа в программу.

2. На что указывают регистры DS и ES?

На начало сегмента PSP (48DDh).

3. Как определяется стек?

Стек определяется на основе директивы .stack и SP указывает на конец сегмента стека.

4. Как определяется точка входа?

Точка входа определяется с помощью директивы END.

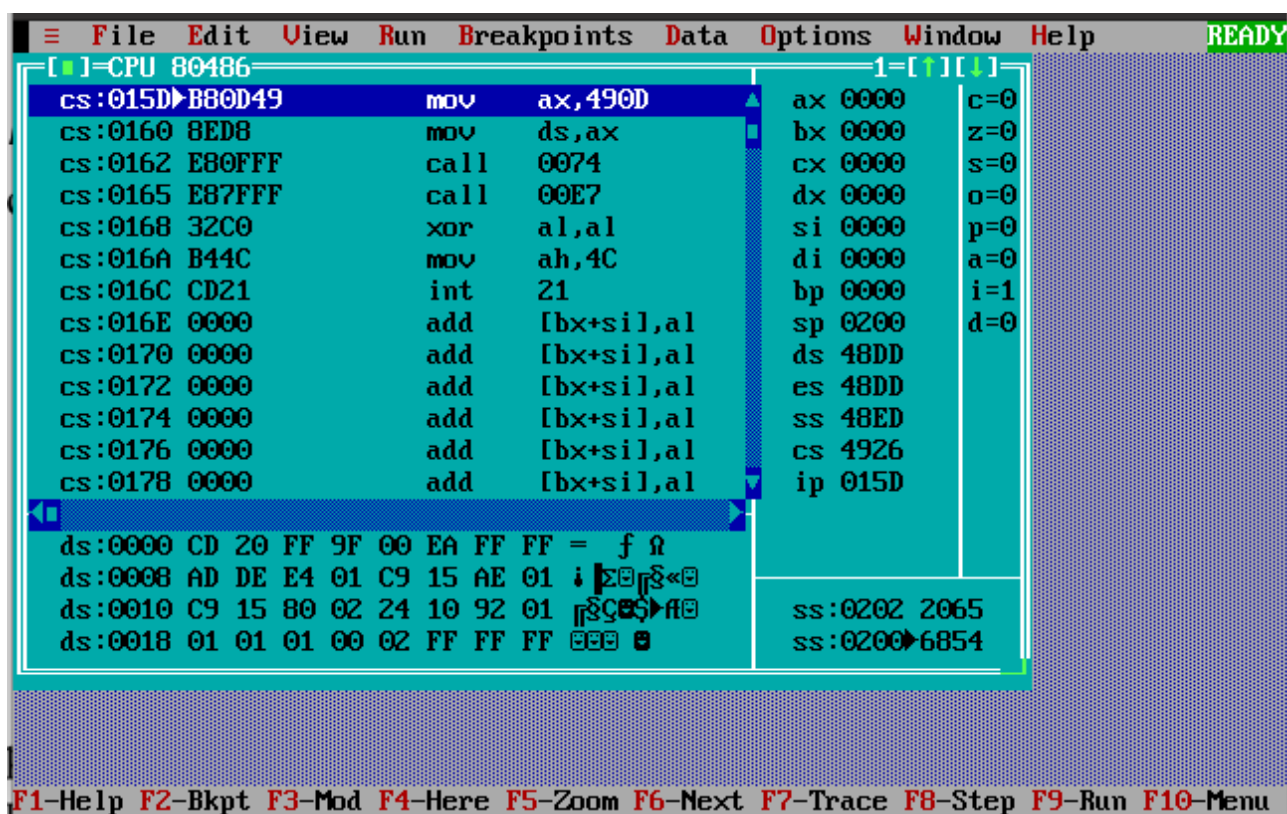


Рис. 8. Интерфейс TD.EXE (.EXE)

## **Заключение**

В процессе выполнения лабораторной работы были изучены структурные отличия .COM и .EXE модулей и получены навыки работы с отладчиком TD.EXE.

## Приложение А.

TESTPC SEGMENT

ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING  
org 100h

start: jmp begin

PCTYPE db 'The type of your PC is: PC', 0dh, 0ah, '\$'  
PCXTTYPE db 'The type of your PC is: PC/XT', 0dh, 0ah, '\$'  
ATTYPE db 'The type of your PC is: AT', 0dh, 0ah, '\$'  
PS230TYPE db 'The type of your PC is: PS2 model 30', 0dh, 0ah, '\$'  
PS25060TYPE db 'The type of your PC is: PS2 model 50 or 60', 0dh, 0ah, '\$'  
PS280TYPE db 'The type of your PC is: PS2 model 80', 0dh, 0ah, '\$'  
PCJRTYPE db 'The type of your PC is: PCjr', 0dh, 0ah, '\$'  
PCCTYPE db 'The type of your PC is: PC Convertible', 0dh, 0ah, '\$'  
DOS\_VERSION\_GREETINGS db 'Your DOS version: \$'  
DOS\_OEM\_GREETINGS db 'Your OEM number: \$'  
DOS\_SERIAL\_GREETINGS db 'Your serial number: \$'  
DEC\_NUMBER db ' \$'  
SERIAL\_NUMBER db ' \$'  
NEWLINE db 0dh, 0ah, '\$'  
ERROR\_REIMPLEMENT db 'ERROR BUT IMPLEMENT ERROR HANDLER!', 0dh, 0ah, '\$'

TETR\_TO\_HEX proc near  
and al, 0fh  
cmp al, 09  
jbe next  
add al, 07

next:  
add al, 30h  
ret

TETR\_TO\_HEX endp

BYTE\_TO\_HEX proc near  
push cx  
mov ah, al  
call TETR\_TO\_HEX  
xchg al, ah  
mov cl, 4  
shr al, cl  
call TETR\_TO\_HEX  
pop cx  
ret  
BYTE\_TO\_HEX endp

WRD\_TO\_HEX proc near  
push bx  
mov bh, ah



```

    call BYTE_TO_HEX
    mov [di], ah
    dec di
    mov [di], al
    dec di
    mov al, bh
    call BYTE_TO_HEX
    mov [di], ah
    dec di
    mov [di], al
    pop bx
    ret
WRD_TO_HEX endp

```

```

BYTE_TO_DEC proc near

```

```

    push cx
    push dx
    xor ah, ah
    xor dx, dx
    mov cx, 10

```

```

loop_bd:
    div cx
    or dl, 30h
    mov [si], dl
    dec si
    xor dx, dx
    cmp ax, 10
    jae loop_bd
    cmp al, 00h
    je end_l
    or al, 30h
    mov [si], al

```

```

end_l:
    pop dx
    pop cx
    ret

```

```

BYTE_TO_DEC endp

```

```

; Пишет al в DEC_NUMBER
BYTE_TO_WRD proc near

```

```

    push ax
    push cx
    push dx
    push di

```

```

    mov cx, 0
    mov dx, 0

```

```
mov di, offset DEC_NUMBER
```

```
loop_label:
```

```
    cmp ax, 0
```

```
    je write_to_str
```

```
    mov bx, 10
```

```
    div bx
```

```
    push dx
```

```
    inc cx
```

```
    xor dx, dx
```

```
    jmp loop_label
```

```
write_to_str:
```

```
    cmp cx, 0
```

```
    je byte_to_wrd_exit
```

```
    pop dx
```

```
    add dx, 48
```

```
    mov [di], dx
```

```
    dec cx
```

```
    jmp write_to_str
```

```
byte_to_wrd_exit:
```

```
    inc di
```

```
    mov byte ptr [di], '$' ; закрываем строчку
```

```
    pop di
```

```
    pop dx
```

```
    pop cx
```

```
    pop ax
```

```
    ret
```

```
BYTE_TO_WRD endp
```

```
PRINT_NEWLINE proc near
```

```
    push ax
```

```
    push dx
```

```
    mov dx, offset NEWLINE
```

```
    mov ah, 9h
```

```
    int 21h
```

```
    pop dx
```

```
    pop ax
```

ret

PRINT\_NEWLINE endp

PRINT\_PC\_TYPE proc near

mov ax, 0f000h  
mov es, ax  
mov di, 0fffeh  
mov bl, es:[di]

; PC - 0xff  
cmp bl, 0ffh  
je pc\_write\_dx

; PC/XT - 0xfe, 0xfb  
cmp bl, 00feh  
je pcxt\_write\_dx  
cmp bl, 00fbh  
je pcxt\_write\_dx

; AT - 0xfc  
cmp bl, 0fch  
je at\_write\_dx

; PS2 model 30  
cmp bl, 0fah  
je ps230\_write\_dx

; PS2 model 50 or 60  
cmp bl, 0fch  
je ps25060\_write\_dx

; PS2 model 80  
cmp bl, 0f8h  
je ps280\_write\_dx

; PCjr  
cmp bl, 0fdh  
je pcjr\_write\_dx

; PC Convertible  
cmp bl, 0fdh  
je pcc\_write\_dx

; al to hex string, string ptr in dx  
mov dx, offset ERROR\_REIMPLEMENT  
jmp print\_type

pc\_write\_dx:

```

    mov dx, offset PCTYPE
    jmp print_type

pcxt_write_dx:
    mov dx, offset PCXTTYPE
    jmp print_type

at_write_dx:
    mov dx, offset ATTYPE
    jmp print_type

ps230_write_dx:
    mov dx, offset PS230TYPE
    jmp print_type

ps25060_write_dx:
    mov dx, offset PS25060TYPE
    jmp print_type

ps280_write_dx:
    mov dx, offset PS280TYPE
    jmp print_type

pcjr_write_dx:
    mov dx, offset PCJRTYPE
    jmp print_type

pcc_write_dx:
    mov dx, offset PCCTYPE
    ; тут можно было бы и без jmp print_type, но это чревато ошибками при
    ; добавлении новых типов компуктеров
    jmp print_type

print_type:
    ; string ptr в dx
    mov ah, 9
    int 21h

print_pc_type_exit:
    ret

PRINT_PC_TYPE endp

PRINT_DOS_VERSION proc near

    push ax
    push bx
    push cx
    push dx

    ; DOS VERSION
    mov dx, offset DOS_VERSION_GREETINGS

```

```
mov ah, 9h
int 21h
```

```
mov ah, 30h
int 21h
; оставляем только al
and ax, 0ffh
call BYTE_TO_WRD
```

```
mov dx, offset DEC_NUMBER
mov ah, 9h
int 21h
```

```
; print "."
mov dl, '.'
mov ah, 02h
int 21h
```

```
; оставляем только ah
and ax, 0ff00h
call BYTE_TO_WRD
```

```
mov dx, offset DEC_NUMBER
mov ah, 9h
int 21h
```

```
call PRINT_NEWLINE
```

```
; OEM
mov dx, offset DOS_OEM_GREETINGS
mov ah, 9h
int 21h
```

```
push bx      ; сохраняем bx, потому что в bl тоже есть нужная информация
and bx, 0ff00h ; оставляем только bh
mov ax, bx    ; заносим в ax для процедуры BYTE_TO_WRD
call BYTE_TO_WRD
```

```
; и печатаем
mov dx, offset DEC_NUMBER
mov ah, 9h
int 21h
pop bx      ; восстанавливаемся
```

```
call PRINT_NEWLINE
```

```
; SERIAL
mov dx, offset DOS_SERIAL_GREETINGS
mov ah, 9h
int 21h
```

```
; cx в SERIAL_NUMBER
```



```
mov di, offset SERIAL_NUMBER
add di, 5          ; ?????????????????????????????
mov ax, cx
call WRD_TO_HEX
```

```
; в al и ah коды символов цифр числа из bl
mov al, bl
call BYTE_TO_HEX
sub di, 2
mov [di], ax
```

```
; выводим номер
mov dx, offset SERIAL_NUMBER
mov ah, 9h
int 21h
```

print\_dos\_version\_exit:

```
pop dx
pop cx
pop bx
pop ax
```

```
ret
```

PRINT\_DOS\_VERSION endp

begin:

```
; 1. Вывести на экран тип компуктера. Если тип не совпадает ни
; с одним из известных, то переводим двоичный код в 16-чную строку
; 2. Определить версию DOS и вывести ее в формате xx.yy, xx - основная версия, yy - номер
модификации.
```

```
; Вывод осуществить в 16-чной системе
call PRINT_PC_TYPE
call PRINT_DOS_VERSION
```

```
; выход в DOS
xor al, al
mov ah, 4ch
int 21h
```

TESTPC ENDS

END start

## Приложение Б.

AStack SEGMENT STACK

dw 256 DUP(?) ; 1 килобайт

AStack ENDS

DATA SEGMENT

PCTYPE db 'The type of your PC is: PC', 0dh, 0ah, '\$'

PCXTTYPE db 'The type of your PC is: PC/XT', 0dh, 0ah, '\$'

ATTYPE db 'The type of your PC is: AT', 0dh, 0ah, '\$'

PS230TYPE db 'The type of your PC is: PS2 model 30', 0dh, 0ah, '\$'

PS25060TYPE db 'The type of your PC is: PS2 model 50 or 60', 0dh, 0ah, '\$'

PS280TYPE db 'The type of your PC is: PS2 model 80', 0dh, 0ah, '\$'

PCJRTYPE db 'The type of your PC is: PCjr', 0dh, 0ah, '\$'

PCCTYPE db 'The type of your PC is: PC Convertible', 0dh, 0ah, '\$'

DOS\_VERSION\_GREETINGS db 'Your DOS version: \$'

DOS\_OEM\_GREETINGS db 'Your OEM number: \$'

DOS\_SERIAL\_GREETINGS db 'Your serial number: \$'

DEC\_NUMBER db ' \$'

SERIAL\_NUMBER db ' \$'

NEWLINE db 0dh, 0ah, '\$'

ERROR\_REIMPLEMENT db 'ERROR BUT IMPLEMENT ERROR HANDLER!', 0dh, 0ah, '\$'

DATA ENDS

CODE SEGMENT

ASSUME cs:CODE, ds:DATA, ss:AStack

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

; Пишет al в DEC\_NUMBER

BYTE\_TO\_WRD proc near

push ax

push cx

push dx

push di

mov cx, 0

mov dx, 0

mov di, offset DEC\_NUMBER

loop\_label:

cmp ax, 0

je write\_to\_str

mov bx, 10

div bx

push dx

inc cx

xor dx, dx

jmp loop\_label

write\_to\_str:

cmp cx, 0

je byte\_to\_wrd\_exit

pop dx

add dx, 48

mov [di], dx

dec cx

jmp write\_to\_str

byte\_to\_wrd\_exit:

inc di

mov byte ptr [di], '\$' ; закрываем строчку

pop di

pop dx

pop cx

pop ax

ret

BYTE\_TO\_WRD endp

PRINT\_NEWLINE proc near

push ax

push dx

mov dx, offset NEWLINE

mov ah, 9h

int 21h

pop dx

pop ax

ret

PRINT\_NEWLINE endp

PRINT\_PC\_TYPE proc near

mov ax, 0f000h



```
mov es, ax
mov di, 0fffeh
mov bl, es:[di]
```

```
; PC - 0xff
cmp bl, 0ffh
je pc_write_dx
```

```
; PC/XT - 0xfe, 0xfb
cmp bl, 00feh
je pcxt_write_dx
cmp bl, 00fbh
je pcxt_write_dx
```

```
; AT - 0xfc
cmp bl, 0fch
je at_write_dx
```

```
; PS2 model 30
cmp bl, 0fah
je ps230_write_dx
```

```
; PS2 model 50 or 60
cmp bl, 0fch
je ps25060_write_dx
```

```
; PS2 model 80
cmp bl, 0f8h
je ps280_write_dx
```

```
; PCjr
cmp bl, 0fdh
je pcjr_write_dx
```

```
; PC Convertible
```

```
cmp bl, 0fdh
je pcc_write_dx
```

```
; al to hex string, string ptr in dx
mov dx, offset ERROR_REIMPLEMENT
jmp print_type
```

```
pc_write_dx:
    mov dx, offset PCTYPE
    jmp print_type
```

```
pcxt_write_dx:
    mov dx, offset PCXTTYPE
    jmp print_type
```

```
at_write_dx:
    mov dx, offset ATTYPE
    jmp print_type
```

```
ps230_write_dx:
    mov dx, offset PS230TYPE
    jmp print_type
```

```
ps25060_write_dx:
    mov dx, offset PS25060TYPE
    jmp print_type
```

```
ps280_write_dx:
    mov dx, offset PS280TYPE
    jmp print_type
```

```
pcjr_write_dx:
    mov dx, offset PCJRTYPE
    jmp print_type
```

pcc\_write\_dx:

mov dx, offset PCCTYPE

; тут можно было бы и без jmp print\_type, но это чревато ошибками при

; добавлении новых типов компуктеров

jmp print\_type

print\_type:

; string ptr в dx

mov ah, 9

int 21h

print\_pc\_type\_exit:

ret

PRINT\_PC\_TYPE endp

PRINT\_DOS\_VERSION proc near

push ax

push bx

push cx

push dx

; DOS VERSION

mov dx, offset DOS\_VERSION\_GREETINGS

mov ah, 9h

int 21h

mov ah, 30h

int 21h

; оставляем только al

and ax, 0ffh

call BYTE\_TO\_WRD

```
mov dx, offset DEC_NUMBER
mov ah, 9h
int 21h
```

```
; print "."
mov dl, '.'
mov ah, 02h
int 21h
```

```
; оставляем только ah
and ax, 0ff00h
call BYTE_TO_WRD
```

```
mov dx, offset DEC_NUMBER
mov ah, 9h
int 21h
```

```
call PRINT_NEWLINE
```

```
; OEM
mov dx, offset DOS_OEM_GREETINGS
mov ah, 9h
int 21h
```

```
push bx      ; сохраняем bx, потому что в bl тоже есть нужная информация
and bx, 0ff00h ; оставляем только bh
mov ax, bx    ; заносим в ax для процедуры BYTE_TO_WRD
call BYTE_TO_WRD
```

```
; и печатаем
mov dx, offset DEC_NUMBER
mov ah, 9h
int 21h
pop bx      ; восстанавливаемся
```

```
call PRINT_NEWLINE
```

```
; SERIAL
```

```
mov dx, offset DOS_SERIAL_GREETINGS
```

```
mov ah, 9h
```

```
int 21h
```

```
; cx в SERIAL_NUMBER
```

```
mov di, offset SERIAL_NUMBER
```

```
add di, 5 ; ??????????????????????????????
```

```
mov ax, cx
```

```
call WRD_TO_HEX
```

```
; в al и ah коды символов цифр числа из bl
```

```
mov al, bl
```

```
call BYTE_TO_HEX
```

```
sub di, 2
```

```
mov [di], ax
```

```
; выводим номер
```

```
mov dx, offset SERIAL_NUMBER
```

```
mov ah, 9h
```

```
int 21h
```

```
print_dos_version_exit:
```

```
pop dx
```

```
pop cx
```

```
pop bx
```

```
pop ax
```

```
ret
```

```
PRINT_DOS_VERSION endp
```



Main proc far

mov ax, DATA

mov ds, ax

call PRINT\_PC\_TYPE

call PRINT\_DOS\_VERSION

; выход в DOS

xor al, al

mov ah, 4ch

int 21h

Main endp

CODE ENDS

END Main