

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

Щеглов Д.А

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 com.asm, который определяет тип PC и версию системы. Далее при помощи транслятора MASM.exe и компоновщика LINK.exe был скомпилирован плохой .EXE модуль. При помощи EXE2BIN.EXE был построен хороший .COM модуль.

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

3. Было выполнено сравнение исходных текстов com.asm и exe.asm.

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

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

Функции программ.

Названия функций	Описание
TETR_TO_HEX	Перевод десятичной цифры в код символа.
BYTE_TO_HEX	Перевод байта в 16-ной с/с в символьный код
WRD_TO_HEX	Перевод слова в 16-ной с/с в символьный код
BYTE_TO_DEC	Перевод байта в 16-ной с/с в символьный код в 10-ной с/с
PRINT_STRING	Вывод строки.

Ответы на контрольные вопросы.

Отличия исходных текстов COM и EXE программ

1. Сколько сегментов должна содержать COM-программа?

COM-программа должна содержать всего один сегмент, в котором находятся код и данные.

2. EXE программа?

EXE программа может содержать несколько сегментов. Это сегменты стека, данных и кода.

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

Директива `ORG 100h`. Она задает смещение для всех адресов программы на 256 байт для префикса программного сегмента (PSP).

Директива `ASSUME`. Она указывает ассемблеру, с каким сегментом или группой сегментов связан тот или иной сегментный регистр. Она не изменяет значений сегментных регистров, а только позволяет ассемблеру проверять допустимость ссылок и самостоятельно вставлять при необходимости префиксы переопределения сегментов.

Директива `END`. Этой директивой завершается любая программа на ассемблере.

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

Нет, можно использовать не все форматы команд. Нельзя использовать команды, связанные с адресом сегмента, потому что адрес сегмента до загрузки неизвестен, так как в COM-программах в DOS не содержится таблицы настройки, которая содержит описание адресов, зависящих от размещения загрузочного модуля в ОП.

Отличия форматов файлов COM и EXE модулей

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

В COM-файле код, данные и стек располагаются в одном сегменте. Код (как и данные) начинается с адреса `0h`.

```

C:\Users\niki_\OneDrive\Рабочий стол\Stud\2-2\OS\Lab1\src\EXE\MAIN_COM.COM |h|ANSI
00000000: E9 BE 01 54 79 70 65 3A 20 50 43 0D 0A 24 54 79 йs0Type: PCJ0$Ty
00000001: 70 65 3A 20 50 43 2F 58 54 0D 0A 24 54 79 70 65 pe: PC/XTJ0$Type
00000002: 3A 20 41 54 0D 0A 24 54 79 70 65 3A 20 50 53 32 : ATJ0$Type: PS2
00000003: 20 EC EE E4 E5 EB FC 20 33 30 0D 0A 24 54 79 70 модель 30J0$Typ
00000004: 65 3A 20 50 53 32 20 EC EE E4 E5 EB FC 20 38 30 e: PS2 модель 80
00000005: 0D 0A 24 54 79 70 65 3A 20 50 D1 6A 72 0D 0A 24 J0$Type: PCjrJ0$
00000006: 54 79 70 65 3A 20 50 43 20 43 6F 6E 76 65 72 74 Type: PC Convert
00000007: 69 62 6C 65 0D 0A 24 56 65 72 73 69 6F 6E 20 4D ibleJ0$Version M
00000008: 53 2D 44 4F 53 3A 20 20 2E 20 20 0D 0A 24 53 65 S-DOS: . J0$Se
00000009: 72 69 61 6C 20 6E 75 6D 62 65 72 20 4F 45 4D 3A rial number OEM:
0000000A: 20 20 0D 0A 24 55 73 65 72 20 73 65 72 69 61 6C J0$User serial
0000000B: 20 6E 75 6D 62 65 72 3A 20 20 20 20 20 20 20 48 number: H
0000000C: 20 24 24 0F 3C 09 76 02 04 07 04 30 C3 51 8A E0 $$o<ov0♦♦♦0ГQЪа
0000000D: E8 EF FF 86 C4 B1 04 D2 E8 E8 E6 FF 59 C3 53 8A ипярД±♦ТиияYГSЪ
0000000E: FC E8 E9 FF 88 25 4F 88 05 4F 8A C7 E8 DE FF 88 ыйя€%0€+0Ъзиюя€
0000000F: 25 4F 88 05 5B C3 51 52 56 32 E4 33 D2 B9 0A 00 %0€+[ГQRV2дЗТ№
00000010: F7 F1 80 CA 30 88 14 4E 33 D2 3D 0A 00 73 F1 3C чсЪK0€JNЗТ= sc<
00000011: 00 74 04 0C 30 88 04 5E 5A 59 C3 B4 09 CD 21 C3 т♦90€♦^ZYГгоH!Г
00000012: B8 00 F0 8E C0 26 A0 FE FF 3C FF 74 1C 3C FE 74 ё рЪA& юя<ятL<ют
00000013: 1E 3C FB 74 1A 3C FC 74 1C 3C FA 74 1E 3C F8 74 ▲<йт-><ьтL<ьт▲<шт
00000014: 20 3C FD 74 22 3C F9 74 24 BA 03 01 EB 25 90 BA <эт"<шт$€♥0л%ђе
00000015: 0E 01 EB 1F 90 BA 1C 01 EB 19 90 BA 27 01 EB 13 0л♥ђеL0л↓ђе'0л!!
00000016: 90 BA 3D 01 EB 0D 90 BA 53 01 EB 07 90 BA 60 01 ђе=0лJђеS0л`0л
00000017: EB 01 90 E8 A5 FF C3 B4 30 CD 21 50 BE 77 01 83 л0иГ'яГг0H!Psw0Г
00000018: C6 10 E8 71 FF 58 8A C4 83 C6 02 E8 68 FF BA 77 Ж>ицяXЪдГЖ0иияew
00000019: 01 E8 87 FF BE 8E 01 83 C6 13 8A C7 E8 57 FF BA 0иiяsЪ0ГЖ!!!ЪзиWя€
0000001A: 8E 01 E8 76 FF BF A5 01 83 C7 19 8B C1 E8 2E FF Ъ0иивяiГ'0ГЗ↓<Би.я
0000001B: 8A C3 E8 18 FF 83 EF 02 89 05 BA A5 01 E8 5B FF ЪГи↑яГп0%€Г0и[я
0000001C: C3 E8 5C FF E8 B0 FF 32 C0 B4 4C CD 21 Ги\яи°я2ArLH!

```

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

В плохом EXE-файле код, данные и стек располагаются в одном сегменте. Код (как и данные) начинается с адреса 300h. С адреса 0h располагается управляющая информация для загрузчика, которая содержит заголовок и таблицу настройки адресов.

C:\Users\niki_\OneDrive\Рабочий стол\Stud\2-2\OS\Lab1\src\EXE\MAIN_COM.EXE																		
0000000000:	4D	5A	CD	00	03	00	00	00	20	00	00	00	FF	FF	00	00	MZH ▼	яя
0000000010:	00	00	95	42	00	01	00	00	1E	00	00	00	01	00	00	00	•B @ ▲	@
0000000020:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000030:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000040:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000050:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000060:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000070:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000080:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000090:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000000A0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000000B0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000000C0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000000D0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000000E0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000000F0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000100:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000110:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000120:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000130:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000140:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000150:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000160:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000170:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000180:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000190:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000001A0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000001B0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000001C0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000001D0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000001E0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000001F0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000200:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000210:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000220:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000230:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000240:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000250:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000260:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000270:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000280:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000290:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000002A0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000002B0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000002C0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000002D0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000002E0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000002F0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000300:	E9	BE	01	54	79	70	65	3A	20	50	43	0D	0A	24	54	79	йs@Type: PC	йs@Type: PC
0000000310:	70	65	3A	20	50	43	2F	58	54	0D	0A	24	54	79	70	65	pe: PC/XT	pe: PC/XT
0000000320:	3A	20	41	54	0D	0A	24	54	79	70	65	3A	20	50	53	32	: AT	: AT
0000000330:	20	EC	EE	E4	E5	EB	FC	20	33	30	0D	0A	24	54	79	70	модель 30	модель 30
0000000340:	65	3A	20	50	53	32	20	EC	EE	E4	E5	EB	FC	20	38	30	e: PS2 модель 80	e: PS2 модель 80
0000000350:	0D	0A	24	54	79	70	65	3A	20	50	D1	6A	72	0D	0A	24	йs@Type: PCjr	йs@Type: PCjr
0000000360:	54	79	70	65	3A	20	50	43	20	43	6F	6E	76	65	72	74	Type: PC Convert	Type: PC Convert
0000000370:	69	62	6C	65	0D	0A	24	56	65	72	73	69	6F	6E	20	4D	ible	ible
0000000380:	53	2D	44	4F	53	3A	20	20	2E	20	20	0D	0A	24	53	65	S-DOS: .	S-DOS: .
0000000390:	72	69	61	6C	20	6E	75	6D	62	65	72	20	4F	45	4D	3A	rial number OEM:	rial number OEM:
00000003A0:	20	20	0D	0A	24	55	73	65	72	20	73	65	72	69	61	6C	йs@User serial	йs@User serial
00000003B0:	20	6E	75	6D	62	65	72	3A	20	20	20	20	20	20	20	48	number:	number:
00000003C0:	20	24	24	0F	3C	09	76	02	04	07	04	30	C3	51	8A	E0	\$s<ov@♦♦♦0GQ/ba	\$s<ov@♦♦♦0GQ/ba

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

В отличие от плохого, хороший EXE-файл не содержит директивы ORG 100h (которая выделяет память под PSP), поэтому код начинается с адреса 200h. В хорошем EXE-файле код, данные и стек находятся в различных сегментах, а в плохом – в одном и том же сегменте.

```

C:\Users\niki\OneDrive\Рабочий стол\Stud\2-2\OS\Lab1\src\EXE\MAIN_EXE.EXE
00000000: 4D 5A F3 00 03 00 01 00 20 00 00 00 FF FF 00 00 MZy  ▼  @  яя
00000001: 00 01 9E AB FF 00 1E 00 1E 00 00 00 01 00 03 01  @h«я  ▲  ▲  @  ▼  @
00000002: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ▲
00000003: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000004: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000005: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000006: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000007: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000008: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000009: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000B: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000D: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000011: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000012: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000013: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000014: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000015: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000016: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000017: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000018: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000019: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001B: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001D: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000021: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000022: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000023: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000024: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000025: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000026: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000027: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000028: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000029: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000002A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000002B: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000002C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000002D: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000002E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000002F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000030: 54 79 70 65 3A 20 50 43 0D 0A 24 54 79 70 65 3A Type: PC$Type:
00000031: 20 50 43 2F 58 54 0D 0A 24 54 79 70 65 3A 20 41 PC/XT$Type: A
00000032: 54 0D 0A 24 54 79 70 65 3A 20 50 53 32 20 EC EE T$Type: PS2 мо
00000033: E4 E5 EB FC 20 33 30 0D 0A 24 54 79 70 65 3A 20 дель 30$Type:
00000034: 50 53 32 20 EC EE E4 E5 EB FC 20 35 30 20 E8 EB PS2 модель 50 ил
00000035: E8 20 36 30 0D 0A 24 54 79 70 65 3A 20 50 53 32 и 60$Type: PS2
00000036: 20 EC EE E4 E5 EB FC 20 38 30 0D 0A 24 54 79 70 модель 80$Type:
00000037: 65 3A 20 50 D1 6A 72 0D 0A 24 54 79 70 65 3A 20 e: PCjr$Type:
00000038: 50 43 20 43 6F 6E 76 65 72 74 69 62 6C 65 0D 0A PC Convertible$
00000039: 24 56 65 72 73 69 6F 6E 20 4D 53 2D 44 4F 53 3A $Version MS-DOS:
0000003A: 20 20 2E 20 20 0D 0A 24 53 65 72 69 61 6C 20 6E . $Serial n
0000003B: 75 6D 62 65 72 20 4F 45 4D 3A 20 20 0D 0A 24 55 umber OEM: $U
0000003C: 73 65 72 20 73 65 72 69 61 6C 20 6E 75 6D 62 65 ser serial numbe

```

1. Что располагается с 0 адреса?

С адреса 0 располагается префикс программного сегмента (PSP).

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

Регистры DS, ES, CS, SS указывают на начало блока PSP.

3. Как определяется стек? Какую область памяти он занимает? Какие адреса?

Стек создается автоматически. Регистр SS указывает на начало блока PSP (0h), а SP указывает на конец модуля (FFFFh). То есть стек расположен между адресами SS:0000h и SS:FFFFh и заполняется с конца модуля в сторону уменьшения адресов.

Загрузка «хорошего» EXE модуля в память

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

Загружается «хороший» EXE со считыванием информации заголовка EXE, выполняется перемещение адресов сегментов, ES и DS устанавливаются на начало PSP, SS – на начало сегмента стека, а CS – на начало сегмента команд.

DOSBox 0.74-3, Cpu speed: 3000 cycles, Fram...

AX 0000	SI 0000	CS 1A0A	IP 0000	Stack +0 4350	Flags 7202
BX 0000	DI 0000	DS 19F5		+2 5420	
CX 013B	BP 0000	ES 19F5	HS 19F5	+4 7079	OF DF IF SF ZF AF PF CF
DX 0000	SP 0000	SS 1A05	FS 19F5	+6 3A65	0 0 1 0 0 0 0 0

CMD >

0000 EB62	JMP	0064
0002 90	NOP	
0003 B409	MOV	AX,09
0005 CD21	INT	21
0007 C3	RET	
000B 240F	AND	AL,0F
000A 3C09	CMP	AL,09
000C 7602	JNA	0010

DS:0000	CD 20 FF 9F 00 EA F0 FE	AD DE 1B 05 C5 06 00 00	0 1 2 3 4 5 6 7
DS:0008	AD DE 1B 05 C5 06 00 00	01 01 01 00 02 FF FF FF	
DS:0010	18 01 10 01 18 01 92 01	FF FF FF FF FF FF FF FF	
DS:0018	01 01 01 00 02 FF FF FF	FF FF FF FF EB 19 C0 11	
DS:0020	FF FF FF FF FF FF FF FF	A2 01 14 00 18 00 F5 19	
DS:0028	FF FF FF FF FF FF FF FF	FF FF FF FF 00 00 00 00	
DS:0030	A2 01 14 00 18 00 F5 19	05 00 00 00 00 00 00 00	
DS:0038	FF FF FF FF FF FF FF FF		
DS:0040	05 00 00 00 00 00 00 00		
DS:0048	00 00 00 00 00 00 00 00		

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 ↑ 8 ↓ 9 ← 10 →

2. На что указывают регистры DS и ES?

Изначально регистры DS и ES указывают на начало сегмента PSP.

3. Как определяется стек?

Стек определяется с помощью директивы .stack с указанием размера стека. SS указывает на начало сегмента стека, а SP – на конец.

4. Как определяется точка входа?

Точка входа определяется директивой END.

Вывод.

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 программ.

Файл: com.asm

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

; ДАННЫЕ
PC_Type      db      'PC Type: ', 0dh, 0ah, '$'
Mod_numb     db      'Modification number: . ', 0dh, 0ah, '$'
OEM          db      'OEM: ', 0dh, 0ah, '$'
S_numb       db      'Serial Number: ', 0dh, 0ah, '$'

; ПРОЦЕДУРЫ
; печать строки
PRINT_STRING PROC near
    mov     ah, 09h
    int     21h
    ret
PRINT_STRING ENDP

; перевод десятичной цифры в код символа
TETR_TO_HEX  PROC     near
    and     al, 0fh ;логическое умножение всех пар
битов
    cmp     al, 09
    jbe     NEXT ;Переход если ниже или равно
    add     al, 07
NEXT:  add     al, 30h
    ret
TETR_TO_HEX  ENDP
; байт в AL переводится в два символа шестнадцетиричного числа в AX
BYTE_TO_HEX  PROC     near
    push    cx
    mov     al, ah
    call    TETR_TO_HEX
    xchg    al, ah
    mov     cl, 4
    shr     al, cl ;логический сдвиг вправо
    call    TETR_TO_HEX
    pop     cx
    ret
BYTE_TO_HEX  ENDP

; в AX - число, DI - адрес последнего символа
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
```

```

        xor     ah, ah
        call    BYTE_TO_HEX
        mov     [di], ah
        dec     di
        mov     [di], al
        pop     bx
        ret

WRD_TO_HEX      ENDP

```

;перевод байта 16 с.с в символный код 10 с.с

;si - адрес поля младшей цифры

```

BYTE_TO_DEC     PROC      near
        push    cx
        push    dx
        push    ax
        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     ax, 00h
        jbe     end_1
        or      al, 30h
        mov     [si], al
end_1:          pop     ax
        pop     dx
        pop     cx
        ret

BYTE_TO_DEC     ENDP

```

BEGIN:

```

;PC_Type
push     es
push     bx
push     ax
mov     bx, 0F000h
mov     es, bx
mov     ax, es:[0FFFEh]
mov     ah, al
call    BYTE_TO_HEX
lea     bx, PC_Type
mov     [bx + 9], ax; смещение на количество символов
pop     ax
pop     bx
pop     es

;Mod_number
mov     ah, 30h
int     21h
push    ax
push    si
lea     si, Mod_numb
add     si, 21

```

```

        call    BYTE_TO_DEC
        add     si, 3
        mov     al, ah
        call    BYTE_TO_DEC
        pop     si
        pop     ax

;OEM
        mov     al, bh
        lea     si, OEM
        add     si, 7
        call    BYTE_TO_DEC

;S_number
        mov     al, bl
        call    BYTE_TO_HEX
        lea     di, S_num
        add     di, 15
        mov     [di], ax
        mov     ax, cx
        lea     di, S_num
        add     di, 20
        call    WRD_TO_HEX

        lea     dx, PC_Type
        call    PRINT_STRING
        lea     dx, Mod_num
        call    PRINT_STRING
        lea     dx, OEM
        call    PRINT_STRING
        lea     dx, S_num
        call    PRINT_STRING

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

TESTPC  ENDS

        END     START

```

Файл exe.asm:

```

AStack  SEGMENT  STACK
AStack  ENDS

```

DATA SEGMENT

```

PC_Type          db      'PC Type: ', 0dh, 0ah, '$'
Mod_num          db      'Modification number: ', 0dh, 0ah, '$'
OEM              db      'OEM: ', 0dh, 0ah, '$'
S_num            db      'Serial Number: ', 0dh, 0ah, '$'
DATA ENDS

```

CODE SEGMENT

```

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

```

START:

```

        jmp BEGIN
;ПРОЦЕДУРЫ

```

```

;печатать строки
PRINT_STRING PROC near
    mov     ah, 09h
    int     21h
    ret
PRINT_STRING ENDP

;перевод десятичной цифры в код символа
TETR_TO_HEX PROC near
    and     al, 0fh ;логическое умножение всех пар битов
    cmp     al, 09
    jbe     NEXT
    add     al, 07
NEXT:      add     al, 30h
    ret
TETR_TO_HEX ENDP

;байт в AL переводится в 2 символа шестнадцетиричного числа в AX
BYTE_TO_HEX PROC near
    push    cx
    mov     al, ah
    call    TETR_TO_HEX
    xchg    al, ah
    mov     cl, 4
    shr     al, cl ;логический сдвиг вправо
    call    TETR_TO_HEX
    pop     cx
    ret
BYTE_TO_HEX ENDP

;в AX - число, DI - адрес последнего символа
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
    xor     ah, ah
    call    BYTE_TO_HEX
    mov     [di], ah
    dec     di
    mov     [di], al
    pop     bx
    ret
WRD_TO_HEX ENDP

;si - адрес поля младшей цифры
BYTE_TO_DEC PROC near
    push    cx
    push    dx
    push    ax
    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     ax, 00h
                                jbe     end_l
                                or      al, 30h
                                mov     [si], al
end_l:  pop                     ax
                                pop     dx
                                pop     cx
                                ret
BYTE_TO_DEC                    ENDP

BEGIN:
    push    ds
    sub     ax, ax
    push    ax
    mov     ax, DATA
    mov     ds, ax

                                ;PC_Type
                                push     es
                                push     bx
                                push     ax
                                mov     bx, 0F000h
                                mov     es, bx
                                mov     ax, es:[0FFFEh]
                                mov     ah, al
                                call     BYTE_TO_HEX
                                lea      bx, PC_Type
                                mov     [bx + 9], ax ;смещение на количество символов
                                pop      ax
                                pop      bx
                                pop      es

                                ;Mod_number
                                mov     ah, 30h
                                int      21h
                                push     ax
                                push     si
                                lea     si, Mod_numb
                                add     si, 21
                                call     BYTE_TO_DEC
                                add     si, 3
                                mov     al, ah
                                call     BYTE_TO_DEC
                                pop     si
                                pop     ax

                                ;OEM
                                mov     al, bh

```

```

        lea     si, OEM
        add     si, 7
        call    BYTE_TO_DEC

;S_Number
mov     al, bl
call    BYTE_TO_HEX
lea     di, S_numb
add     di, 15
mov     [di], ax
mov     ax, cx
lea     di, S_numb
add     di, 20
call    WRD_TO_HEX

lea     dx, PC_Type
call    PRINT_STRING
lea     dx, Mod_numb
call    PRINT_STRING
lea     dx, OEM
call    PRINT_STRING
lea     dx, S_numb
call    PRINT_STRING

;ВЫХОД В dos
xor     al, al
mov     ah, 4ch
int     21h
ret

CODE     ENDS

END      START

```