

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

Кафедра МОЭВМ

ОТЧЕТ
лабораторная работа №1
по дисциплине «Операционные системы»
Тема: Исследование структур загрузочных модулей

Студент гр. 9381

Николаев А.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.

Основные теоретические положения.

Тип IBM PC хранится в байте по адресу 0F000:0FFFEh, в предпоследнем байте ROM BIOS. Соответствие кода и типа представлены в табл.1.

Таблица 1 – Соответствие кода и типа PC

PC	FF
PC/XT	FE, FB
AT	FC
PS2 модель 30	
FA PS2 модель 50 или 60	
FC PS2 модель 80	
F8	
PCjr	FD
PC Convertible	F9

Для определения версии MS DOS следует воспользоваться функцией 30H прерывания 21H. Входным параметром является номер функции в AH. Выходными параметрами являются:

AL - номер основной версии. Если 0, то < 2.0

AH - номер модификации

BH - серийный номер OEM (Original Equipment Manufacturer)

BL:CH - 24-битовый серийный номер пользователя.

Ход работы.

Был написан код исходного .COM модуля, на основе теории, данной в методических указаниях, который определяет и выводит в консоль необходимую по заданию информацию о РС. В результате были получены «плохой» .EXE модуль и, «хороший» .COM модуль.

Ниже представлены скриншоты «хорошего» .COM модуля и «плохого» .EXE соответственно.

```
C:\>lab1com.COM
PC: AT
SYSTEM VERSION: 5.0
DEM: 0
SERIAL USER NUMBER: 0
C:\>
```

Стоит отметить, что при запуске «плохого» .EXE вывелось предупреждение об отсутствии стека.

Был написан текст «хорошего» .EXE модуля, который выполняет ту же функцию, что и .COM модуль. Ниже представлен скриншот «хорошего» .EXE.

Ответы на контрольные вопросы:

1. Сколько сегментов должна содержать COM-программа?

COM-программа должна содержать только один сегмент, в котором находится код и данные.

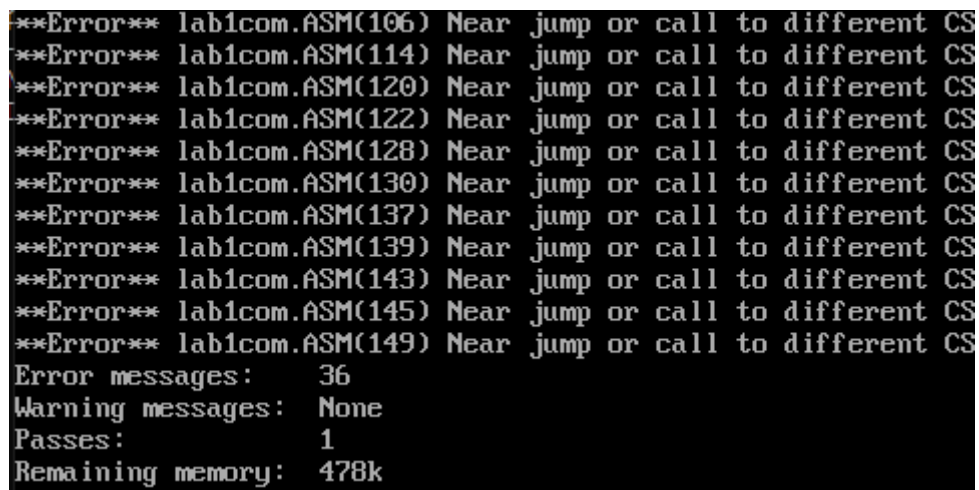
2. EXE-программа?

EXE-программа может содержать несколько сегментов. В программах описываются сегменты стека, данных, кода.

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

При запуске COM-программы первые 100h байт необходимо зарезервировать для префикса программного сегмента (PSP).

Необходима директива ASSUME. При компиляции COM модуля без неё возникают ошибки.



```

**Error** lab1com.ASM(106) Near jump or call to different CS
**Error** lab1com.ASM(114) Near jump or call to different CS
**Error** lab1com.ASM(120) Near jump or call to different CS
**Error** lab1com.ASM(122) Near jump or call to different CS
**Error** lab1com.ASM(128) Near jump or call to different CS
**Error** lab1com.ASM(130) Near jump or call to different CS
**Error** lab1com.ASM(137) Near jump or call to different CS
**Error** lab1com.ASM(139) Near jump or call to different CS
**Error** lab1com.ASM(143) Near jump or call to different CS
**Error** lab1com.ASM(145) Near jump or call to different CS
**Error** lab1com.ASM(149) Near jump or call to different CS
Error messages: 36
Warning messages: None
Passes: 1
Remaining memory: 478k

```

Список ошибок при попытке запустить .COM-программу без директивы ASSUME. Как видно на скриншоте, эта директива связывает сегментные регистры и программные сегменты. Без нее не удастся вызвать jump и call.

Директива END - директива завершения программы.

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

Нет, не все. В .COM файле отсутствует таблица настройки с информацией о типе адресов и их местоположении в коде. Поэтому нельзя использовать команды, связанные с адресом сегмента, так как адрес сегмента неизвестен вплоть до загрузки этого сегмента в память. Загрузчику необходима информация о местоположении в файле загрузочного модуля полей адресов.

Отличия форматов файлов COM и EXE модулей.

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

COM-файл состоит из команд, процедур и данных, используемых в программе. Код начинается с нулевого адреса, это видно на скриншоте. Код, данные и стек находятся в одном сегменте.

00000000	E9 C1 00 50 43 3A 20 24 50 43 0D 0A 24 50 43 2F	...PC:.\$PC..\$PC/
0000010	58 54 0D 0A 24 41 54 24 0D 0A 24 50 43 20 28 33	XT..\$AT\$..\$PC.(3
0000020	30 20 6D 6F 64 65 6C 29 0D 0A 24 50 43 20 28 35	0.model)..\$PC.(5
0000030	30 20 6F 72 20 36 30 20 6D 6F 64 65 6C 29 0D 0A	0.or.60.model)..
0000040	24 50 43 20 28 38 30 20 6D 6F 64 65 6C 29 0D 0A	\$PC.(80.model)..
0000050	24 50 43 20 6A 72 0D 0A 24 50 43 20 43 6F 6E 76	\$PC.jr..\$PC.Conv
0000060	65 72 74 69 62 6C 65 0D 0A 24 2E 24 0D 0A 53 59	ertible..\$.\$..SY
0000070	53 54 45 4D 20 56 45 52 53 49 4F 4E 3A 20 24 0D	STEM.VERSION:.\$.
0000080	0A 4F 45 4D 3A 20 24 0D 0A 53 45 52 49 61 6C 20	.OEM:.\$..SERIAL.
0000090	55 53 45 52 20 4E 55 4D 42 45 52 3A 20 24 B4 09	USER.NUMBER:.\$..
00000A0	CD 21 C3 33 C9 8B DA 33 D2 F7 F3 52 41 3B C0 75	.!.3...3...RA;u
00000B0	F6 B4 02 5A 80 FA 09 76 03 80 C2 07 80 C2 30 CD	...Z...v.....0.
00000C0	21 E2 F0 C3 BA 03 01 E8 D4 FF B8 00 F0 8E C0 B8	!.....
00000D0	00 00 26 A0 FE FF 3C FF 74 23 3C FE 74 25 3C FB	..&...<.t#<.t%<.
00000E0	74 21 3C FC 74 23 3C FA 74 25 3C FC 74 27 3C F8	t!<.t#<.t%<.t'<.
00000F0	74 29 3C FD 74 2B 3C F9 74 2D EB 31 90 BA 08 01	t)<.t+<.t-.1....
0000100	EB 34 90 BA 0D 01 EB 2E 90 BA 15 01 EB 28 90 BA	.4.....(..
0000110	1B 01 EB 22 90 BA 2B 01 EB 1C 90 BA 41 01 EB 16	..."..+.....A...
0000120	90 BA 51 01 EB 10 90 BA 59 01 EB 0A 90 BA 10 00	..Q.....Y.....
0000130	E8 70 FF EB 04 90 E8 65 FF B4 30 CD 21 51 53 50	.p.....e..0.!QSP
0000140	BA 6C 01 E8 58 FF B8 00 00 58 50 B4 00 BA 0A 00	.l..X....XP.....
0000150	E8 50 FF BA 6A 01 E8 45 FF 58 8A E5 B4 00 8A C5	.P..j..E.X.....
0000160	BA 0A 00 E8 3D FF BA 7F 01 E8 32 FF 58 50 8A E5=.. ..2.XP..
0000170	B4 00 8A C5 BA 0A 00 E8 29 FF BA 87 01 E8 1E FF).....
0000180	58 B4 00 3C 00 74 06 BA 0A 00 E8 16 FF 58 BA 0A	X..<.t.....X..
0000190	00 E8 0F FF 32 C0 B4 4C CD 212..L.!

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

В таком EXE-файле код начинается с адреса 300h. С нулевого адреса располагается управляющая таблица для загрузчика, которая содержит заголовок, таблицу настройки адресов и показывает, что данный файл - EXE.

```
00000000 4D 5A 9A 00 03 00 00 00 20 00 00 00 FF FF 00 00  MZ.....
00000100 00 00 81 C1 00 01 00 00 1E 00 00 00 01 00 00 00  .....
00000200 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00000300 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00000400 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00000500 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00000600 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00000700 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00000800 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00000900 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00000A00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00000B00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00000C00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00000D00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00000E00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00000F00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00001000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00001100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00001200 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00001300 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00001400 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00001500 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00001600 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00001700 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00001800 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00001900 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00001A00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00001B00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00001C00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00001D00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00001E00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00001F00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00002000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00002100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00002200 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00002300 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00002400 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00002500 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00002600 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00002700 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00002800 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00002900 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00002A00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00002B00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00002C00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00002D00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00002E00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00002F00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00003000 E9 C1 00 50 43 3A 20 24 50 43 0D 0A 24 50 43 2F  ..PC!.$PC!.$PC/
00003100 58 54 0D 0A 24 41 54 24 0D 0A 24 50 43 20 28 33  XT!.$AT!.$PC.(3
00003200 30 20 6D 6F 64 65 6C 29 0D 0A 24 50 43 20 28 35  g.model!.$PC.(5
00003300 30 20 6F 72 20 36 30 20 6D 6F 64 65 6C 29 0D 0A  g.or.60.model!..
00003400 24 50 43 20 28 38 30 20 6D 6F 64 65 6C 29 0D 0A  $PC.(80.model)..
00003500 24 50 43 20 6A 72 0D 0A 24 50 43 20 43 6F 6E 76  $PC.jr!.$PC.Conv
00003600 65 72 74 69 62 6C 65 0D 0A 24 2E 24 0D 0A 53 59  ertible!.$$.SY
00003700 53 54 45 4D 20 56 45 52 53 49 4F 4E 3A 20 24 0D  STEM.VERSION!.$
00003800 0A 4F 45 4D 3A 20 24 0D 0A 53 45 52 49 61 6C 20  .DEM!.$$.SERIAL.
00003900 55 53 45 52 20 4E 55 4D 42 45 52 3A 20 24 0D 09  USER.NUMBER!.$
00003A00 CD 21 C3 33 C9 8B 0A 33 02 F7 F3 52 41 3B C0 75  .1.3...3...RA;u
00003B00 F6 B4 02 5A 00 FA 09 76 03 88 C2 07 80 C2 30 CD  ...Z...V.....0.
00003C00 21 E2 F0 C3 BA 03 01 E8 04 FF 8B 00 F0 BE C0 B8  l.....
00003D00 00 00 26 A0 FE FF 3C FF 74 23 3C FE 74 25 3C F8  ..&...<.t#<.t#<.
00003E00 74 21 3C FC 74 23 3C FA 74 25 3C FC 74 27 3C F8  t!<.t#<.t#<.t'<.
00003F00 74 29 3C F0 74 28 3C F9 74 2D EB 31 90 BA 08 01  t)<.t+<.t-..1....
00004000 EB 34 90 BA 0D 01 EB 2E 90 BA 15 01 EB 28 90 BA  .4...t+<.t-..1....
00004100 1B 01 EB 22 90 BA 2B 01 EB 1C 90 BA 41 01 EB 16  ...".+.....A....
00004200 90 BA 51 01 EB 19 90 BA 59 01 EB 0A 90 BA 10 00  ..Q...Y.....
00004300 E8 70 FF EB 04 90 EB 65 FF B4 30 CD 21 51 53 50  .p....0...0.IQSP
00004400 BA 6C 01 E8 58 FF 8B 00 00 58 50 B4 00 BA 0A 00  .l...X....XP....
00004500 E8 50 FF BA 6A 01 E8 45 FF 58 8A E5 04 00 8A C5  .P..j...E.X.....
00004600 BA 0A 00 E8 3D FF BA 7F 01 E8 32 FF 58 50 8A E5  ....=...2.XP...
00004700 B4 00 8A C5 BA 0A 00 E8 29 FF BA 87 01 E8 1E FF  .......).
00004800 58 B4 00 3C 00 74 06 BA 0A 00 E8 16 FF 58 BA 0A  X...<.t...X...
00004900 00 E8 0F FF 32 C0 B4 4C CD 21  ....2...L..l
```

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

В «хорошем» EXE - стек располагается перед кодом. При отсутствии стека, код будет располагаться с адреса 200h. 200h перед ним займет управляющая

информация для загрузчика. Соответственно, адрес начала программы равен 200h + размер стека h.

```
0000000 4D 5A 70 00 03 00 01 00 20 00 00 00 FF FF 00 00 MZp.....
0000010 C8 00 DE 1F 26 00 17 00 1E 00 00 00 01 00 2B 00 ....&.....+.
0000020 17 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00001A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00001B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00001C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00001D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00001E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00001F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000200 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 .....
0000210 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 .....
0000220 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 .....
0000230 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 .....
0000240 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 .....
0000250 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 .....
0000260 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 .....
0000270 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 .....
0000280 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 .....
0000290 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 .....
00002A0 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 .....
00002B0 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 .....
00002C0 01 00 01 00 01 00 01 00 00 00 00 00 00 00 00 00 .....
00002D0 50 43 3A 20 24 50 43 0D 0A 24 50 43 2F 58 54 0D PC:.$PC..$PC/XT.
00002E0 0A 24 41 54 24 0D 0A 24 50 53 32 20 28 33 30 20 .SATS..$PS2.(30.
00002F0 6D 6F 64 65 6C 29 0D 0A 24 50 53 32 20 28 35 30 model)..$PS2.(50
0000300 20 6F 72 20 36 30 20 6D 6F 64 65 6C 29 0D 0A 24 .or.60.model)..$
0000310 50 53 32 20 28 38 30 20 6D 6F 64 65 6C 29 0D 0A PS2.(80.model)..
0000320 24 50 43 20 6A 72 0D 0A 24 50 43 20 43 6F 6E 76 $PC.jr..$PC.Conv
0000330 65 72 74 69 62 6C 65 0D 0A 24 2E 24 0D 0A 53 59 ertible..$.SY
0000340 53 54 45 4D 20 56 45 52 53 49 4F 4E 3A 20 24 0D STEM.VERSION:$.
0000350 0A 4F 45 4D 3A 20 24 0D 0A 53 45 52 49 41 4C 20 .OEM:$.SERIAL.
0000360 55 53 45 52 20 4E 55 4D 42 45 52 3A 20 24 00 00 USER.NUMBER:$.
0000370 B4 09 CD 21 C3 33 C9 8B DA 33 D2 F7 F3 52 41 3B ...i.3...3...RA;
0000380 C0 75 F6 B4 02 5A 80 FA 09 76 03 80 C2 07 80 C2 .u...Z...v.....
0000390 30 CD 21 E2 F0 C3 1E 2B C0 50 B8 0D 00 8E D8 BA 0.i....+.P.....
00003A0 00 00 E8 CB FF B8 00 F0 8E C0 B8 00 00 26 A0 FE .....&..
00003B0 FF 3C FF 74 23 3C FE 74 25 3C FB 74 21 3C FC 74 .<.t#<.t%<.t!<.t
00003C0 23 3C FA 74 25 3C FC 74 27 3C F8 74 29 3C FD 74 #<.t%<.t'<.t)<.t
00003D0 2B 3C F9 74 2D EB 31 90 BA 05 00 EB 34 90 BA 0A +<.t-.1....4...
00003E0 00 EB 2E 90 BA 12 00 EB 28 90 BA 18 00 EB 22 90 .....(.....".
00003F0 BA 29 00 EB 1C 90 BA 40 00 EB 16 90 BA 51 00 EB .).....@.....Q..
0000400 10 90 BA 59 00 EB 0A 90 BA 10 00 E8 67 FF EB 04 ...Y.....g...
```


Загрузка COM модуля в основную память.

AX 0000	SI 0000	CS 119C	IP 0100	Stack +0 0000	FLAGS 0200							
BX 0000	DI 0000	DS 119C		+2 0000								
CX 019A	BP 0000	ES 119C	HS 119C	+4 0000	OF DF IF SF ZF AF PF CF	0	0	1	0	0	0	0
DX 0000	SP FFF5	SS 119C	FS 119C	+6 0000		0	0	1	0	0	0	0
CMD >				1		0	1	2	3	4	5	6
				DS:0000		CD	20	27	58	00	EA	FD
				DS:0008		AD	DE	ED	04	92	01	00
0100 E9C100				JMP 01C4		DS:0010	18	01	10	01	18	01
0103 50				PUSH AX		DS:0018	03	FF	FF	FF	FF	FF
0104 43				INC BX		DS:0020	FF	FF	FF	FF	FF	FF
0105 3A20				CMP AH,[BX+SI]		DS:0028	FF	FF	FF	FF	96	11
0107 2450				AND AL,50		DS:0030	92	01	14	00	18	00
0109 43				INC BX		DS:0038	FF	FF	FF	FF	00	00
010A 0D0A24				OR AX,240A		DS:0040	05	00	00	00	00	00
010D 50				PUSH AX		DS:0048	00	00	00	00	00	00
2						0	1	2	3	4	5	6
DS:0000						CD	20	27	58	00	EA	FD
DS:0010						18	01	10	01	18	01	92
DS:0020						FF	FF	FF	FF	FF	FF	FF
DS:0030						92	01	14	00	18	00	9C
DS:0040						05	00	00	00	00	00	00
1 Step				2 StepProc	3 Retrieve	4 Help	5 Set BRK	6	7 up	8 dn	9 le	0 ri

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

После загрузки COM модуля в память, сегментные регистры указывают на начало PSP. Код начинается с адреса 100h.

2. Что располагается с адреса 0?

С адреса 0 располагается сегмент PSP.

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

Сегментные регистры указывают на PSP и поэтому они равны.

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

В COM модуле стек нельзя объявить, он объявляется автоматически. Указатель SP указывает на FFFFh. SS указывает на начало PSP 0h. Стек лежит между SS и SP и адреса меняются от больших к меньшим. От FFFFh до 0000h.

Загрузка «хорошего» EXE. модуля в основную память.

AX 0000	SI 0000	CS 11C3	IP 0026	Stack +0 0000	FLAGS 0200																									
BX 0000	DI 0000	DS 119C		+2 0000																										
CX 0270	BP 0000	ES 119C	HS 119C	+4 0000	OF	DF	IF	SF	ZF	AF	PF	CF																		
DX 0000	SP 00C8	SS 11AC	FS 119C	+6 0000	0	0	1	0	0	0	0	0																		
CMD >																														
				1	0	1	2	3	4	5	6	7																		
				DS:0000	CD	20	CC	46	00	EA	FD	FF																		
				DS:0008	AD	DE	ED	04	92	01	00	00																		
0026 1E	PUSH DS			DS:0010	18	01	10	01	18	01	92	01																		
0027 2BC0	SUB AX,AX			DS:0018	03	FF	FF	FF	FF	FF	FF	FF																		
0029 50	PUSH AX			DS:0020	FF	FF	FF	FF	FF	FF	FF	FF																		
002A B8B911	MOV AX,11B9			DS:0028	FF	FF	FF	FF	96	11	C4	FF																		
002D 8ED8	MOV DS,AX			DS:0030	92	01	14	00	18	00	9C	11																		
002F BA0000	MOV DX,0000			DS:0038	FF	FF	FF	FF	00	00	00	00																		
0032 E8CBFF	CALL 0000			DS:0040	05	00	00	00	00	00	00	00																		
0035 B800F0	MOV AX,F000			DS:0048	00	00	00	00	00	00	00	00																		
2																														
				0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F											
DS:0000					CD	20	CC	46	00	EA	FD	FF	AD	DE	ED	04	92	01	00	00	. .F....									
DS:0010					18	01	10	01	18	01	92	01	03	FF	FF	FF	FF	FF	FF	FF									
DS:0020					FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	96	11	C4	FF									
DS:0030					92	01	14	00	18	00	9C	11	FF	FF	FF	FF	00	00	00	00									
DS:0040					05	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00									
1 Step													2StepProc		3Retrieve		4 Help		5Set BRK		6		7 up		8 dn		9 le		0 ri	

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

Создается PSP. После запуска программы DS и ES указывают на начало PSP. CS – на начало сегмента команд. а SS – на начало сегмента стека.

2. На что указывают регистры DS и ES?

Изначально регистры DS и ES указывают на начало сегмента PSP.

3. Как определяется стек?

Стек может быть объявлен с помощью директивы STACK. Если стек не объявлять, то он будет создан автоматически таким же образом, как в COM-модуле.

4. Как определяется точка входа?

При помощи директивы END. Смещение точки входа в программу загружается в указатель команд IP.

Вывод.

Были изучены .COM и .EXE модули. Были выведены их различия и получены плохой и хороший EXE. Изучена разница между ними.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ. lab1com.asm

TestPC SEGMENT

ASSUME CS:TestPC, DS:TestPC, es:NOTHING, SS:NOTHING

ORG 100H

START: JMP BEGIN

TYPEPC db 'PC: \$'

PC db 'PC', 13,10,'\$'

PC_TYPE_XT db 'PC/XT', 13,10,'\$'

PC_TYPE_AT db 'AT\$', 13,10,'\$'

PC_TYPE_30 db 'PC (30 model)', 13,10,'\$'

PC_TYPE_5060 db 'PC (50 or 60 model)', 13,10,'\$'

PC_TYPE_80 db 'PC (80 model)', 13,10,'\$'

PC_TYPE_JR db 'PC jr', 13,10,'\$'

PC_TYPE_CONVERTIBLE db 'PC Convertible', 13,10,'\$'

DOT db '.\$'

VERSION db 13, 10, 'SYSTEM VERSION: \$'

OEM db 13, 10, 'OEM: \$'

USER db 13, 10, 'SERIAL USER NUMBER: \$'

PRINT PROC NEAR

mov ah,9

int 21h

ret

PRINT ENDP

OutInt proc near

xor cx, cx

mov bx, dx

OutInt2:

xor dx,dx

div bx

```

        push    dx
        inc     cx
        cmp     ax, ax
        jnz     OutInt2
        mov     ah, 02h
OutInt3:
        pop     dx
        cmp     dl, 9
        jbe     OutInt4
        add     dl, 7
OutInt4:
        add     dl, '0'
        int     21h
        loop    OutInt3
        ret
OutInt endp

```

BEGIN:

```

; Определяем весию ПК
mov     dx, OFFSET TYPEPC
call    PRINT
mov     ax, 0F000h
mov     es, ax
mov     ax, 0
mov     al, es:[0FFFEh]
cmp     al, 0FFh
jz      PCLABEL
cmp     al, 0FEh
jz      PC_TYPE_XTLABEL
cmp     al, 0FBh
jz      PC_TYPE_XTLABEL
cmp     al, 0FCh
jz      PC_TYPE_ATLABEL
cmp     al, 0FAh
jz      PC_TYPE_30LABEL

```

```

        cmp     al, 0FCh
        jz      PC_TYPE_5060LABEL
        cmp     al, 0F8h
        jz      PC_TYPE_80LABEL
        cmp     al, 0FDh
        jz      PC_TYPE_JRLABEL
        cmp     al, 0F9h
        jz      PC_TYPE_CONVERTIBLELABEL
        jmp     UNKNOWN_TYPE_LABEL

PCLABEL:
        mov     dx, OFFSET PC
        jmp     PCTYPE_OUT

PC_TYPE_XTLABEL:
        mov     dx, OFFSET PC_TYPE_XT
        jmp     PCTYPE_OUT

PC_TYPE_ATLABEL:
        mov     dx, OFFSET PC_TYPE_AT
        jmp     PCTYPE_OUT

PC_TYPE_30LABEL:
        mov     dx, OFFSET PC_TYPE_30
        jmp     PCTYPE_OUT

PC_TYPE_5060LABEL:
        mov     dx, OFFSET PC_TYPE_5060
        jmp     PCTYPE_OUT

PC_TYPE_80LABEL:
        mov     dx, OFFSET PC_TYPE_80
        jmp     PCTYPE_OUT

PC_TYPE_JRLABEL:
        mov     dx, OFFSET PC_TYPE_JR
        jmp     PCTYPE_OUT

PC_TYPE_CONVERTIBLELABEL:
        mov     dx, OFFSET PC_TYPE_CONVERTIBLE
        jmp     PCTYPE_OUT

UNKNOWN_TYPE_LABEL:
        mov     dx, 16

```

```

        call OutInt
        jmp      MSDOS_VERSION
PCTYPE_OUT:
        call PRINT
MSDOS_VERSION:
        mov     ah,30h
        int     21h
        push    cx
        push    bx
        push    ax
        mov     dx, OFFSET VERSION
        call PRINT
        mov     ax, 0
        pop     ax
        push    ax
        mov     ah, 0
        mov     dx, 10
        call OutInt
        mov     dx, OFFSET DOT
        call PRINT
        pop     ax
        mov     ah, ch
        mov     ah, 0
        mov     al, ch
        mov     dx, 10
        call OutInt
        mov     dx, OFFSET OEM
        call PRINT
        pop     ax
        push    ax
        mov     ah, ch
        mov     ah, 0
        mov     al, ch
        mov     dx, 10
        call OutInt

```

```

                mov     dx, OFFSET USER
                call    PRINT
                pop     ax
                mov     ah, 0
                cmp     al, 0
                jz      NEXT_NUMBER
                mov     dx, 10
                call    OutInt
NEXT_NUMBER:
                pop     ax
                mov     dx, 10
                call    OutInt
                xor     al, al
                mov     ah, 4CH
                int     21H
TestPC         ENDS
                END     START

```

ИСХОДНЫЙ КОД ПРОГРАММЫ. lab1exe.asm

```

AStack        SEGMENT    STACK
                DW 100 DUP(1)
AStack        ENds

```

```

DATA          SEGMENT

```

```

TYPEPC db 'PC: $'
PC db 'PC', 13,10,'$'
PC_TYPE_XT db 'PC/XT', 13,10,'$'
PC_TYPE_AT db 'AT$', 13,10,'$'
PC_TYPE_30 db 'PS2 (30 model)', 13,10,'$'
PC_TYPE_5060 db 'PS2 (50 or 60 model)', 13,10,'$'
PC_TYPE_80 db 'PS2 (80 model)', 13,10,'$'

```

```
PC_TYPE_JR db 'PC jr', 13,10,'$'
PC_TYPE_CONVERTIBLE db 'PC Convertible', 13,10,'$'

DOT db '.$'
VERSION db 13, 10, 'SYSTEM VERSION: $'
OEM db 13, 10, 'OEM: $'
USER db 13, 10, 'SERIAL USER NUMBER: $'
```

```
DATA      ENds
```

```
CODE      SEGMENT
          ASSUME CS:CODE, ds:DATA, SS:AStack
```

```
PRINT     PROC    NEAR
           mov     ah,9
           int     21h
           ret
PRINT     ENDP
```

```
OutInt proc near
    xor     cx, cx
    mov     bx, dx
oi2:
    xor     dx,dx
    div     bx
    push    dx
    inc     cx
    cmp     ax, ax
    jnz     oi2
    mov     ah, 02h
oi3:
    pop     dx
    cmp     dl,9
    jbe     oi4
    add     dl,7
```



```

oi4:
    add    dl, '0'
    int    21h
    loop   oi3
    ret

```

```

OutInt endp

```

```

Main      PROC  FAR
            push  ds
            sub   ax,ax
            push  ax
            mov   ax,DATA
            mov   ds,ax

            mov   dx, OFFSET TYPEPC
            call  PRINT
            mov   ax, 0F000h
            mov   ES, ax
            mov   ax, 0
            mov   al, ES:[0FFFEh]
            cmp   al, 0FFh
            jz    PCLABEL
            cmp   al, 0FEh
            jz    PC_TYPE_XTLABEL
            cmp   al, 0FBh
            jz    PC_TYPE_XTLABEL
            cmp   al, 0FCh
            jz    PC_TYPE_ATLABEL
            cmp   al, 0FAh
            jz    PC_TYPE_30LABEL
            cmp   al, 0FCh
            jz    PC_TYPE_5060LABEL
            cmp   al, 0F8h
            jz    PC_TYPE_80LABEL

```

```

        cmp     al, 0FDh
        jz      PC_TYPE_JRLABEL
        cmp     al, 0F9h
        jz      PC_TYPE_CONVERTIBLELABEL
        jmp     UNKNOWN_TYPE_LABEL
PCLABEL:
        mov     dx, OFFSET PC
        jmp     PCTYPE_OUT
PC_TYPE_XTLABEL:
        mov     dx, OFFSET PC_TYPE_XT
        jmp     PCTYPE_OUT
PC_TYPE_ATLABEL:
        mov     dx, OFFSET PC_TYPE_AT
        jmp     PCTYPE_OUT
PC_TYPE_30LABEL:
        mov     dx, OFFSET PC_TYPE_30
        jmp     PCTYPE_OUT
PC_TYPE_5060LABEL:
        mov     dx, OFFSET PC_TYPE_5060
        jmp     PCTYPE_OUT
PC_TYPE_80LABEL:
        mov     dx, OFFSET PC_TYPE_80
        jmp     PCTYPE_OUT
PC_TYPE_JRLABEL:
        mov     dx, OFFSET PC_TYPE_JR
        jmp     PCTYPE_OUT
PC_TYPE_CONVERTIBLELABEL:
        mov     dx, OFFSET PC_TYPE_CONVERTIBLE
        jmp     PCTYPE_OUT
UNKNOWN_TYPE_LABEL:
        mov     dx, 16
        call    OutInt
        jmp     MSDOS_VERSION
PCTYPE_OUT:
        call    PRINT

```

MSDOS_VERSION:

```
    mov     ah, 30h
    int     21h
    push    cx
    push    bx
    push    ax
    mov     dx, OFFSET VERSION
    call    PRINT
    mov     ax, 0
    pop     ax
    push    ax
    mov     ah, 0
    mov     dx, 10
    call    OutInt
    mov     dx, OFFSET DOT
    call    PRINT
    pop     ax
    mov     ah, ch
    mov     ah, 0
    mov     al, ch
    mov     dx, 10
    call    OutInt
    mov     dx, OFFSET OEM
    call    PRINT
    pop     ax
    push    ax
    mov     ah, ch
    mov     ah, 0
    mov     al, ch
    mov     dx, 10
    call    OutInt
    mov     dx, OFFSET USER
    call    PRINT
    pop     ax
    mov     ah, 0
```

```

        cmp    al, 0
        jz     NEXT_NUMBER
        mov    dx, 10
        call   OutInt
NEXT_NUMBER:
        pop    ax
        mov    dx, 10
        call   OutInt
        ret
Main     ENDP
CODE     ENds
        END Main

```