

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

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

Студент гр. 7383

Левкович Д.В.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2019

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.

Необходимые сведения для составления программы:

Тип IBM PC хранится в байте по адресу 0F000:0FFFE, в предпоследнем байте ROM BIOS. Соответствие кода и типа в таблице:

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:

MOV AH,30h

INT 21h

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

Описание функций, которые используются в программе:

1. TYPE_OS – печатает тип ОС.

2. VERSION_OS – печатает версию ОС, серийный номер EOM и серийный номер пользователя.
3. PRINT – печатает строку.
4. TETR_TO_HEX - вспомогательная функция для BYTE_TO_HEX.
5. BYTE_TO_HEX – байт в AL переводится в два символа шестн. Числа в AX.
6. WRD_TO_HEX – перевод в 16 с/с 16-ти разрядного числа. В AX – число, DI – адрес последнего символа.
7. BYTE_TO_DEC – перевод в 10 с/с, SI – адрес поля младшей цифры.

Программа читает содержимое предпоследнего байта ROM BIOS, по таблице, сравнивая коды, определяет тип PC и выводит строку с названием модели.

На рис. 1 – 3 приведены результаты работы программ для различных модулей.

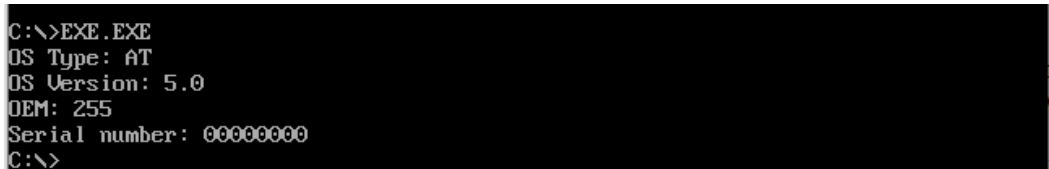
```
C:\>1.com
OS Type: AT
OS Version: 5.0
OEM: 255
Serial number: 00000000
C:\>_
```

Рисунок 1 - Результат работы "хорошего" .COM модуля.

```
C:\>1.exe

OS Type:
OS Type: 5 0
OS Type: 255
OS Type:
OS Type: 000000
OS Type: 00
OS Type:
```

Рисунок 2 - Результат работы "плохого" .EXE модуля.



```
C:\>EXE.EXE
OS Type: AT
OS Version: 5.0
OEM: 255
Serial number: 00000000
C:\>
```

Рисунок 3 - Результат работы "хорошего" .EXE модуля.

Выводы.

В ходе лабораторной работы были разработаны программы на языке ассемблера, определяющие тип PC, версию MS DOS, серийный номер EOM и серийный номер пользователя. Исследованы различия исходных текстов COM и EXE программ, отличие форматов файлов COM и EXE модулей.

Ответы на контрольные вопросы.

Отличия исходных текстов COM и EXE программ

1. Сколько сегментов должна содержать COM-программа?

Ответ: COM-программа содержит 1 сегмент.

2. EXE-программа?

Ответ: Содержит хотя бы 1 сегмент.

3. Какие директивы обязательно должны быть в COM-программе?

Ответ: Обязательным является наличие директивы `ORG 100h`, которая резервирует 256 байт для PSP, выполнение программы начинается со следующей ячейки памяти. Директива `ASSUME`, которая указывает ассемблеру, с каким сегментом связан тот или иной сегментный регистр, без данной директивы программа не скомпилируется, т.к не будет знать, где начинается сегмент кода.

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

Ответ: Нет. Команды, использующие адреса сегментов, запрещены, т.к. отсутствует таблица настроек (Relocation Table). Также нельзя использовать команды, использующие дальнюю адресацию, т.к. для этих команд нужна таблица настройки, в которой содержатся адреса сегментов.

Таблица состоит из элементов, число которых записано в байтах 06-07. Элемент таблицы настройки состоит из двух полей: 2-байтного смещения и 2-байтного сегмента, и указывает слова в загрузочном модуле, содержащее адрес, который должен быть настроен на место памяти, в которое загружается задача.

Настройка производится следующим образом:

В области памяти после резидентной части выполняющей загрузку программы строится префикс программного сегмента (PSP);

- 1. Стандартная часть заголовка считывается в память;*
- 2. Определяется длина тела загрузочного модуля (разность длины файла 04-07 и длины заголовка 08-09 плюс число байт в последнем блоке 02-03). В зависимости от признака, указывающего загружать задачу в конец памяти или в начало, определяется сегментный адрес для загрузки. Этот сегмент называется начальным сегментом;*
- 3. Загрузочный модуль считывается в начальный сегмент;*
- 4. Таблица настройки порциями считывается в рабочую память;*
- 5. Для каждого элемента таблицы настройки к полю сегмента прибавляется сегментный адрес начального сегмента. В результате элемент таблицы указывает на слово в памяти, к которому прибавляется сегментный адрес начального сегмента;*
- 6. Когда таблица настройки адресов обработана, в регистры SS и SP записываются значения, указанные в заголовке, а к SS прибавляется сегментный адрес начального сегмента. В ES и DS записывается*

сегментный адрес начала PSP. Управление передается по адресу, указанному в заголовке (байты 14-17).

Отличие форматов COM и EXE модулей

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

Ответ: Программа содержит единственный сегмент, содержащий и данные, и команды. Код располагается с 0h адреса, как видно из рис. 4.

C:\TASM\TASM\BIN\1.COM																h 1252	
0000000000:	E9	BD	01	4F	53	20	54	79	70	65	3A	20	24	4F	53	20	é%00S Type: \$0S
0000000010:	56	65	72	73	69	6F	6E	3A	20	20	2E	20	20	0D	0A	24	Version: . 0\$
0000000020:	4F	45	4D	3A	20	20	20	20	0D	0A	24	53	65	72	69	61	OEM: 0\$Seria
0000000030:	6C	20	6E	75	6D	62	65	72	3A	20	24	09	24	0D	0A	24	l number: \$0\$0\$
0000000040:	50	43	0D	0A	24	50	43	2F	58	54	0D	0A	24	41	54	0D	PC0\$PC/XT0\$AT0
0000000050:	0A	24	50	53	32	20	6D	6F	64	65	6C	20	33	30	0D	0A	\$PS2 model 300\$
0000000060:	24	50	53	32	20	6D	6F	64	65	6C	20	35	30	20	6F	72	\$PS2 model 50 or
0000000070:	20	36	30	0D	0A	24	50	53	32	20	6D	6F	64	65	6C	20	600\$PS2 model
0000000080:	38	30	0D	0A	24	50	43	6A	72	0D	0A	24	50	43	20	43	800\$PCjr0\$PC C
0000000090:	6F	6E	76	65	72	74	69	62	6C	65	0D	0A	24	24	0F	3C	onvertible0\$0\$<
00000000A0:	09	76	02	04	07	04	30	C3	51	8A	E0	E8	EF	FF	86	C4	ove0\$0\$0\$Q\$SàèiÿtÄ
00000000B0:	B1	04	D2	E8	E8	E6	FF	59	C3	53	8A	FC	E8	E9	FF	88	±0\$èèæÿYÄS\$üèÿ~
00000000C0:	25	4F	88	05	4F	8A	C7	E8	DE	FF	88	25	4F	88	05	5B	%0^0\$Çèÿ^%0^0\$[
00000000D0:	C3	51	52	32	E4	33	D2	B9	0A	00	F7	F1	80	CA	30	88	ÄQR2ä3D01\$ ÷ñ€Ê0^
00000000E0:	14	4E	33	D2	3D	0A	00	73	F1	3C	00	74	04	0C	30	88	ŲN3D= \$ sñ< t0\$0^
00000000F0:	04	5A	59	C3	B4	09	CD	21	C3	BA	03	01	E8	F5	FF	B8	♦ZYÄ´oÍ!Ä\$♥0èöÿ.
0000000100:	00	F0	8E	C0	26	A1	FE	FF	3C	FF	74	20	3C	FE	74	22	ðŽÄ&ipÿ<ÿt <ÿt"
0000000110:	3C	FB	74	1E	3C	FC	74	20	3C	FA	74	22	3C	F8	74	24	<ÿt▲<ÿt <ÿt"<ÿt\$
0000000120:	3C	FC	74	26	3C	FD	74	28	3C	F9	74	2A	BA	40	01	EB	<ÿt&<ÿt (<ÿt*\$00è
0000000130:	2B	90	BA	45	01	EB	25	90	BA	4D	01	EB	1F	90	BA	52	+0\$E0è%0\$M0èv0\$R
0000000140:	01	EB	19	90	BA	61	01	EB	13	90	BA	76	01	EB	0D	90	0è↓0\$a0è!!0\$voè0\$0
0000000150:	BA	85	01	EB	07	90	BA	8C	01	EB	01	90	E8	95	FF	C3	\$...0è•0\$(0è00è•ÿÄ
0000000160:	B8	00	00	B4	30	CD	21	BE	0D	01	83	C6	0C	50	E8	60	.´0Í!%0\$0f0P0è`
0000000170:	FF	58	8A	C4	83	C6	03	E8	57	FF	BA	0D	01	E8	74	FF	ÿX\$Äf0\$èWÿ0\$0èÿÿ
0000000180:	BE	20	01	83	C6	07	8A	C7	E8	46	FF	BA	20	01	E8	63	% 0f0\$•\$ÇèFÿ0 0èc
0000000190:	FF	BA	2B	01	E8	5D	FF	8A	C3	E8	0C	FF	8B	D8	8A	D3	ÿ0+0è]ÿ\$Äè0ÿ<0\$Ó
00000001A0:	B4	02	CD	21	8A	D7	CD	21	BF	3B	01	83	C7	03	8B	C1	´0Í!\$×Í!¿;0fC♥<Ä
00000001B0:	E8	06	FF	BA	3B	01	E8	3B	FF	BA	3D	01	E8	35	FF	C3	è0ÿ0;0è;ÿ0=0è5ÿÄ

Рисунок 4 - Структура файла COM.

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

Ответ: «Плохой» EXE содержит таблицу настроек, которая располагается с 0h адреса и код. Код располагается с адреса 300h (см. рис. 6). 200h байт занимает заголовок и 100h дает директива ORG 100h.

```

C:\TASM\TASM\BIN\1.EXE h 1252
00000000: 4D 5A CC 00 03 00 00 00 20 00 00 00 FF FF 00 00 MZÏ ▼ yy
00000001: 00 00 74 32 00 01 00 00 1E 00 00 00 01 00 00 00 t2 @ ▲ @
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
1Help 2Text 3Quit 4Dump 5 6Edit 7Search 8ANSI 9

```

Рисунок 5 - Структура "плохого" EXE

```

C:\TASM\TASM\BIN\1.EXE h 1252
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: E9 BD 01 4F 53 20 54 79 70 65 3A 20 24 4F 53 20 é%0S Type: $0S
00000031: 56 65 72 73 69 6F 6E 3A 20 20 2E 20 20 0D 0A 24 Version: . $
00000032: 4F 45 4D 3A 20 20 20 20 0D 0A 24 53 65 72 69 61 OEM: $Seria
00000033: 6C 20 6E 75 6D 62 65 72 3A 20 24 09 24 0D 0A 24 1 number: $o$ $
1Help 2Text 3Quit 4Dump 5 6Edit 7Search 8ANSI 9

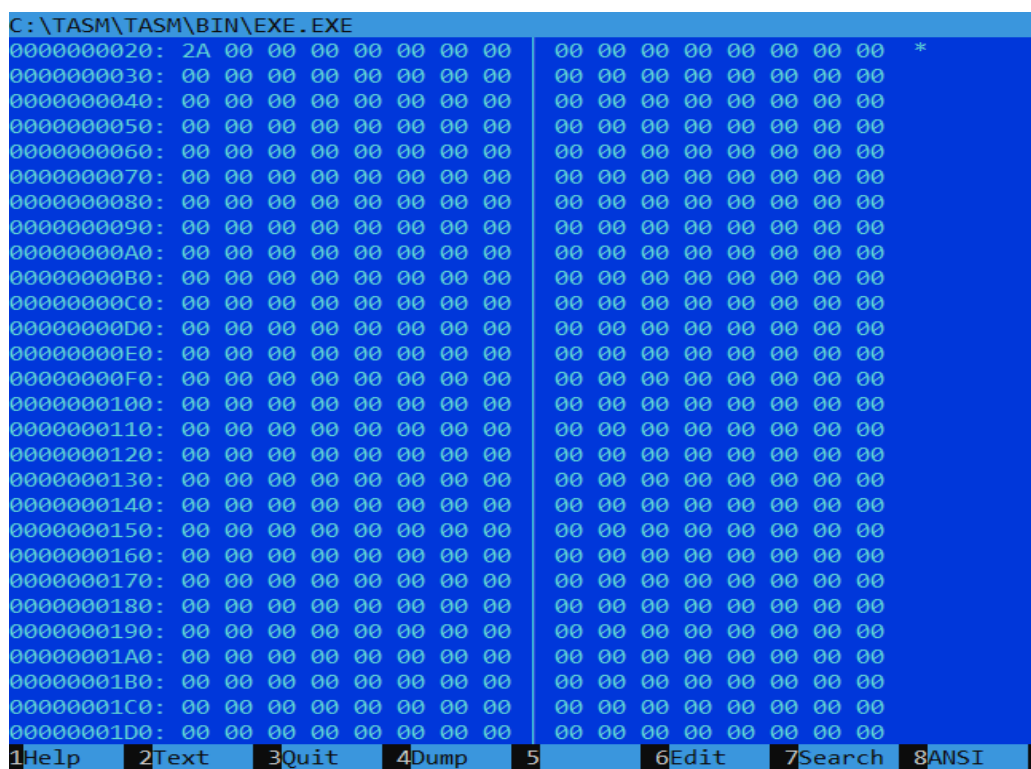
```

Рисунок 6 - Расположение кода в "плохом" EXE

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

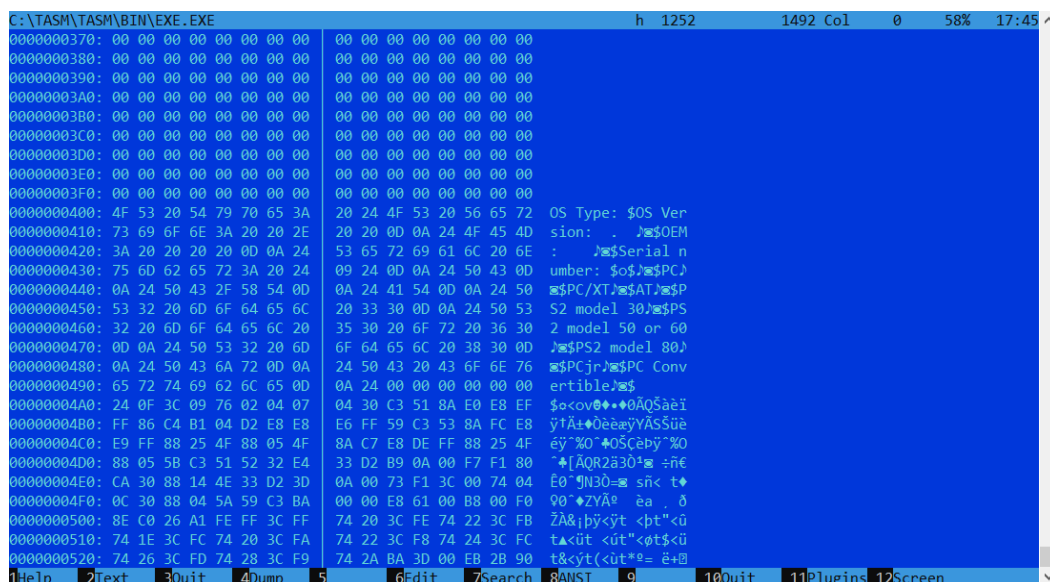
Ответ: В отличие от «плохого» EXE, в «хорошем» есть отдельные сегменты под код, данные и стек. Код программы начинается с адреса 400h, т.к. дополнительно выделено под стек 200 байт.

Проверить длину стека можно посмотрев в самом коде программы, сколько отведено под стек или в карте памяти .MAP.



```
C:\TASM\TASM\BIN\EXE.EXE
00000000: 2A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *
00000001: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
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
```

Рисунок 7 - Структура "хорошего" EXE



```
C:\TASM\TASM\BIN\EXE.EXE
00000037: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000038: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000039: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000003A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000003B: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000003C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000003D: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000003E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000003F: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000040: 4F 53 20 54 79 70 65 3A 20 24 4F 53 20 56 65 72 OS Type: $OS Ver
00000041: 73 69 6F 6E 3A 20 20 2E 20 20 0D 0A 24 4F 45 4D sion: . $OEM
00000042: 3A 20 20 20 20 0D 0A 24 53 65 72 69 61 6C 20 6E : $Serial n
00000043: 75 6D 62 65 72 3A 20 24 09 24 0D 0A 24 50 43 0D umber: $o/$PC)
00000044: 0A 24 50 43 2F 58 54 0D 0A 24 41 54 0D 0A 24 50 $PC/XT/$AT/$P
00000045: 53 32 20 6D 6F 64 65 6C 20 33 30 0D 0A 24 50 53 S2 model 30/$PS
00000046: 32 20 6D 6F 64 65 6C 20 35 30 20 6F 72 20 36 30 2 model 50 or 60
00000047: 0D 0A 24 50 53 32 20 6D 6F 64 65 6C 20 38 30 0D $PS2 model 80)
00000048: 0A 24 50 43 6A 72 0D 0A 24 50 43 20 43 6F 6E 76 $PCjr/$PC Conv
00000049: 65 72 74 69 62 6C 65 0D 0A 24 00 00 00 00 00 00 ertible/$
0000004A: 24 0F 3C 09 76 02 04 07 04 30 C3 51 8A E0 E8 EF $cov0000A0$aei
0000004B: FF 86 C4 B1 04 D2 E8 E8 E6 FF 59 C3 53 8A FC E8 yfA±00eeyVASSu
0000004C: E9 FF 88 25 4F 88 05 4F 8A C7 E8 DE FF 88 25 4F éy"%0*0Scby"%0
0000004D: 88 05 5B C3 51 52 32 E4 33 D2 B9 0A 00 F7 F1 80 "*(AQR2a30"± ñe
0000004E: CA 30 88 14 4E 33 D2 3D 0A 00 73 F1 3C 00 74 04 É0"JN30-± sñ< t+
0000004F: 0C 30 88 04 5A 59 C3 BA 00 00 E8 61 00 B8 00 F0 90*ZYA° ea . d
00000050: 8E C0 26 A1 FE FF 3C FF 74 20 3C FE 74 22 3C FB ZA&|pÿ<yÿt <pt"<ü
00000051: 74 1E 3C FC 74 20 3C FA 74 22 3C F8 74 24 3C FC t<üt <üt"<öt$<ü
00000052: 74 26 3C FD 74 28 3C F9 74 2A BA 3D 00 EB 2B 90 t&<yÿt(<üt*±= Ë+B
```


Рисунок 8 - Структура "хорошего" EXE. Расположение кода.

Загрузка COM модуля в основную память

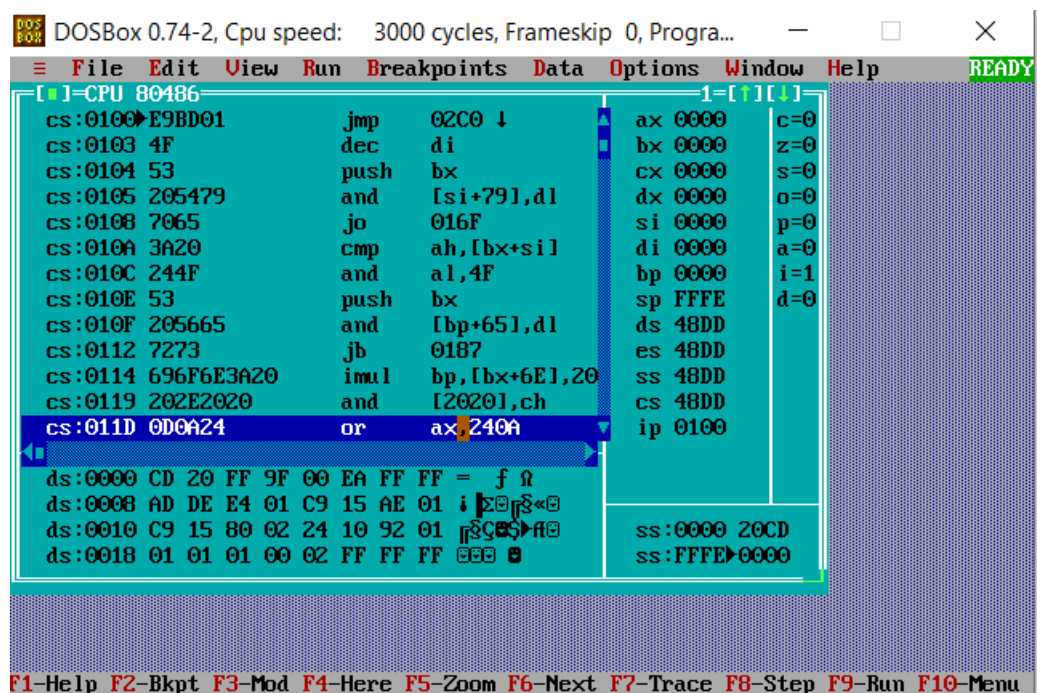


Рисунок 9 - Загрузка COM в основную память

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

Ответ: Сначала выделяется сегмент памяти для модуля. Затем устанавливаются на начало выделенного сегмента памяти. В первые 100h байт располагается PSP. Регистр SP устанавливается в конец сегмента. Код располагается с адреса 48DD.

2. Что располагается с адреса 0?

Ответ: PSP

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

Ответ: Все сегментные регистры указывают на PSP и имеют значения 48DD (см. рис. 9).

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

4. Как определяется точка входа?

Ответ: Смещение точки входа в программу загружается в IP. IP, а именно адрес, с которого начинается выполнение программы определяется операндом директивы END, который называется точкой входа.