

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

Студентка гр. 0381

Шиняева 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.

Постановка задачи

Требуется написать текст исходного .COM модуля, который определяет тип РС и версию системы. Ассемблерная программа должна читать содержимое предпоследнего байта ROM BIOS, по таблице, сравнивая коды, определять тип РС и выводить строку с названием модели. Если код не совпадает ни с одним значением, то двоичный код переводится в символьную строку, содержащую запись шестнадцатеричного числа и выводиться на экран в виде соответствующего сообщения.

Затем определяется версия системы. Ассемблерная программа должна по значениям регистров AL и AH формировать текстовую строку в формате xx.yy, где xx – номер основной версии, а yy - номер модификации в десятичной системе счисления, формировать строки с серийным номером OEM (Original Equipment Manufacturer) и серийным номером пользователя. Полученные строки выводятся на экран.

Далее необходимо отладить полученный исходный модуль и получить «хороший» .COM модуль, а также необходимо построить «плохой» .EXE, полученный из исходного текста для .COM модуля. Затем нужно написать текст «хорошего» .EXE модуля, который выполняет те же функции, что и модуль .COM, далее его построить, отладить и сравнить исходные тексты для .COM и .EXE модулей.

Выполнение работы

Были реализованы следующие функции:

PC_TYPE - определяет тип ПК (по содержимому регистра AL) согласно таблице (Рис. 1).

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

Рис. 1 - Соответствие кода предпоследнего байта ROM BIOS и типа PC

Выполнение программы начинается с занесения в регистр AL данных предпоследнего байта ROM BIOS для определения типа PC. Происходит сравнение регистра AL с числами, соответствующих различным типам PC (Рис.1). В случае совпадения происходит вывод на экран соответствующей числу строки, а в противном случае – замена числа на код ASCII символа и вывод его на экран. Все выводы на экран в программе реализованы с помощью функции write, в которой использована функция 09h прерывание 21h.

PC_TYPE - функция OS, определяет номер основной версии MS-DOS, лежащий в регистре AL, номер её модификации в регистре AH, серийный номер OEM в регистре BH и 24-битный серийный номер пользователя в регистрах BL:CH. Программа последовательно переводит числа в соответствующие им строки и выводит на экран. Завершается программа вызовом функции 4Ch прерывания 21h с кодом 0. Были выделены строки для вывода информации:

- T_PC db 'Type: PC',0DH,0AH,'\$' - для типа PC;
- T_PC_XT db 'Type: PC/XT',0DH,0AH,'\$' - для типа PC/XT;
- T_AT db 'Type: AT',0DH,0AH,'\$' - для типа AT;
- T_PS2_M30 db 'Type: PS2 модель 30',0DH,0AH,'\$' - для типа PS2 модель 30;
- T_PS2_M50_60 db 'Type: PS2 модель 50 или 60',0DH,0AH,'\$' - для типа PS2 модель 50 или 60;
- T_PS2_M80 db 'Type: PS2 модель 80',0DH,0AH,'\$' - для типа PS2 модель 80;

- T_PC_JR db 'Type: PCjr',0DH,0AH,'\$' - для типа PC jr;
- T_PC_C db 'Type: PC Convertible',0DH,0AH,'\$' - для типа PC Convertible;
- VERSION db 'Version: . ',0DH,0AH,'\$' - для версии и модификации;
- OEM db 'OEM: ',0DH,0AH,'\$' - для серийного номера OEM;
- USER db 'User: \$' - для серийного номера пользователя;

Результаты работы «хороших» .COM и .EXE модулей представлены на рисунках 2 и 3. Результат работы «плохого» .EXE представлен на рисунке 4.

```
C:\>com.com
Type: AT
Version: 5.0
OEM: 0
User: 000000
```

Рис. 2 - Результат .COM модуля

```
C:\>exe.exe
Type: AT
Version: 5.0
OEM: 0
User: 000000
```

Рис. 3 - Результат .EXE модуля

```
C:\>com.exe
5 0
Type: PC
0
000000
Type: PC
Type: PC
Type: PC
```

Рис. 4 - Результат "плохого" .EXE модуля

Контрольные вопросы

Отличия исходных текстов COM и EXE программ

1. Сколько сегментов должна содержать COM-программа?

COM-программа должна содержать один сегмент, в котором располагаются команды и данные.

2. Сколько сегментов должна содержать EXE-программа?

EXE-программа может содержать от одного до трёх сегментов: сегмент команд, данных и стека (последние два могут отсутствовать).

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

Должна быть обязательна директива `ORG 100h`, так как при загрузке модуля все сегментные регистры содержат адрес префикса программного сегмента (PSP), который является 256-байтовым(100H) блоком, поэтому адресация имеет смещение в 256 байт от нулевого адреса. Также необходима процедура `ASSUME` для того, чтобы сегмент данных и сегмент кода указывали на один общий сегмент. (`ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING`)

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

Не все форматы команд можно использовать в COM-программе. Так, например, ввиду отсутствия таблицы настроек нельзя использовать команды вида `mov <register>, seg <segment>`. Таблица настроек 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.

Отличия форматов файлов .COM и .EXE модулей

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

Файл .COM состоит из программного сегмента PSP размером 256 байт, одного сегмента, содержащего команды и данные, а также стека, генерируемого автоматически. Структура файла представлена на рисунке 5. Код располагается с адреса 100h (256).

00000000	E9	D0	01	54	79	70	65	3A	20	50	43	0D	0A	24	54	79	31.Type: PC.\$Type
00000010	70	65	3A	20	50	43	2F	58	54	0D	0A	24	54	79	70	65	pe: PC/XT.\$Type
00000020	3A	20	41	54	0D	0A	24	54	79	70	65	3A	20	50	53	32	: AT..\$Type: PS2
00000030	20	D0	BC	D0	BE	D0	B4	D0	B5	D0	BB	D1	8C	20	53	30	..\$Type: PS2
00000040	0D	0A	24	54	79	70	65	3A	20	50	53	32	20	D0	BC	D0	..\$Type: PS2
00000050	BE	D0	B4	D0	B5	D0	BB	D1	8C	20	35	30	20	D0	B8	D0	..\$Type: PS2
00000060	BB	D0	B8	20	36	30	0D	0A	24	54	79	70	65	3A	20	50	..\$Type: PC
00000070	53	32	20	D0	BC	D0	BE	D0	B4	D0	B5	D0	BB	D1	8C	20	S2
00000080	38	30	0D	0A	24	54	79	70	65	3A	20	50	D0	A1	6A	72	80..\$Type: P
00000090	0D	0A	24	54	79	70	65	3A	20	50	43	20	43	6F	6E	76	..\$Type: PC
000000A0	65	72	74	69	62	6C	65	0D	0A	24	56	45	72	73	69	6F	ertible..\$Version
000000B0	6E	3A	20	20	2E	20	20	0D	0A	24	4F	45	4D	3A	20	20	n: . .\$.OEM:
000000C0	0D	0A	24	55	73	65	72	3A	20	20	20	20	20	20	20	20	..\$User:
000000D0	24	24	0F	C3	09	76	02	04	07	04	30	C3	51	8A	E0	E8	\$.<..v...Y\$E
000000E0	EF	FF	86	C4	B1	04	D2	E8	E8	E6	FF	59	C3	53	8A	FC	n a-..t...Y\$E
000000F0	E8	E9	FF	88	25	4F	88	05	4F	8A	C7	E8	DE	FF	88	25	00 e%0.e.0e.0e
00000100	4F	88	05	5B	C3	51	52	32	E4	33	D2	B9	0A	00	F7	F1	0E.[0R23Z3]..=
00000110	80	CA	30	88	14	4E	33	D2	30	0A	00	73	F1	3C	00	74	C0e..N...=s<t<
00000120	04	0C	30	88	04	5A	59	C3	B8	00	F0	E8	C0	26	0A	FE	..0e.ZY...=A&A
00000130	FF	3C	FF	74	1C	3C	FE	74	1E	3C	FB	74	1A	3C	FC	74	< t.<t.<t.<t
00000140	1C	3C	FA	1E	3C	F8	74	26	3C	FB	74	28	3C	F9	74	74	< t.<.*&t<t<t
00000150	2A	BA	03	01	EB	2B	90	BA	0E	01	EB	25	90	BA	1C	01	* .0.eE .0.eE
00000160	EB	1F	90	BA	27	01	EB	19	90	BA	43	01	EB	13	90	BA	0.EE .0.E.C.0.E
00000170	69	01	EB	0D	90	BA	85	01	EB	07	90	BA	93	01	EB	01	0.EE .0.E.C.0.E
00000180	90	BA	09	CD	21	C3	B4	30	CD	21	50	BE	AA	01	83	C6	E ..! H0P ..A
00000190	09	E8	71	FF	58	8A	C4	83	C6	03	E8	68	FF	BA	AA	01	0.q xE-..0h ..
000001A0	B4	09	CD	21	BE	BA	01	83	C6	05	8A	C7	F8	56	FF	BA0.e.0e.0e
000001B0	BA	01	B4	09	CD	21	BF	C3	01	83	C7	08	0B	C1	E8	2C	.. = ..0.e.0e
000001C0	FF	8A	C3	E8	16	FF	83	EF	02	89	05	BA	C3	01	BA	09	e.0. an.e .0.e
000001D0	CD	21	C3	E8	52	FF	E8	AD	FF	32	C0	B4	4C	CD	21		= 0R qf 2 ..

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

```

000002F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000300 E9 D0 01 54 79 70 65 3A 20 50 43 0D 0A 24 54 70 01_Type: PC...$Type
00000310 70 65 3A 20 50 43 2F 58 54 0D 0A 24 54 79 70 65 pe: PC/XT...$Type
00000320 3A 20 41 54 0D 0A 24 54 79 70 65 3A 20 50 53 32 ..AT...$Type: PS2
00000330 20 D0 BC D0 BE D0 B4 D0 B5 D0 BB D1 8C 20 53 33 30 20 D0 BB D1 8C 20 53 33 30 ..$Type: PS2
00000340 0D 0A 24 54 79 70 65 3A 20 50 53 32 20 D0 BC D0 ..$Type: PS2
00000350 BE D0 B4 D0 B5 D0 BB D1 8C 20 35 30 20 D0 B8 D0 ..$Type: PS2
00000360 BB D0 B8 20 B6 30 0D 0A 24 54 79 70 65 3A 20 50 53 32 20 D0 B8 D0 ..$Type: PS2
00000370 53 32 20 D0 BC D0 BE D0 B4 D0 B5 D0 BB D1 8C 20 53 32 20 D0 B8 D0 ..$Type: PS2
00000380 38 30 0D 0A 24 54 79 70 65 3A 20 50 50 D1 6A 72 80...$Type: PC-ijr
00000390 0D 0A 24 54 79 70 65 3A 20 50 43 0D 43 6F 6E 76 ..$Type: PC Convertible..$Version:
000003A0 65 72 74 69 62 6C 65 0D 0A 24 56 65 72 73 69 6F n: ..$ OEM:
000003B0 6E 3A 20 20 2E 20 20 0D 0A 24 4F 45 4D 3A 20 20 20 ..$User:
000003C0 0D 0A 24 55 73 65 72 3A 20 20 20 20 20 20 20 20 ..$$.<v...0
000003D0 24 2A 0F 3C 09 76 02 04 07 04 3C 31 5A 8E 0E E8 ..$$.<v...0
000003E0 EF FF 8C 04 B1 04 D2 E8 E8 E6 FF 59 C3 53 8A FC n ..$$.<v...0
000003F0 E8 E9 FF 88 25 4F 88 05 4F 8A C7 E8 DE FF 88 25 ..$$.<v...0
00000400 4F 88 05 5B C3 51 52 32 E4 33 D2 B9 0A 00 F7 F1 0E ..$$.<v...0
00000410 80 CA 30 88 14 4E 33 D2 3D 0A 00 73 F1 3C 00 74 ..$$.<v...0
00000420 04 0C 30 88 04 5A 59 C3 B8 00 F0 8E 0C 26 A0 FE ..$$.<v...0
00000430 FF 3C FF 74 1C 3C FE 74 1E 3C FB 74 1A 3C FC 74 ..$$.<v...0
00000440 1C 3C FA 74 1C 3C FE 74 26 3C FD 74 28 3C F9 74 ..$$.<v...0
00000450 2A BA 03 01 EB 2B 90 BA 0E 01 EB 25 90 BA 1C 01 ..$$.<v...0
00000460 EF 1F 90 BA 27 01 EB 19 90 BA 43 01 EB 13 90 BA ..$$.<v...0
00000470 69 01 EB 0D 90 BA 85 01 EB 07 90 BA 93 01 EB 01 ..$$.<v...0
00000480 B4 09 CD 21 C3 B4 30 CD 21 50 BE AA 01 83 C6 ..$$.<v...0
00000490 09 E8 71 FF 58 8A C3 C6 03 E8 68 FF BA AA 01 ..$$.<v...0
000004A0 B4 09 CD 21 BE BA 01 83 C6 05 8A C7 E8 56 FF BA ..$$.<v...0
000004B0 BA 01 B4 09 CD 21 BF C3 01 83 C7 0B 8B C1 E8 2C ..$$.<v...0
000004C0 FF 8A C3 E8 16 FF 83 EF 02 89 05 C3 C3 01 B4 09 ..$$.<v...0
000004D0 CD 21 C3 E8 52 FF E8 AD FF 32 C0 B4 C4 CD 21 ..$$.<v...0

```

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

В «хорошем» EXE файле уже исправлено объединение всех сегментов, теперь они разнесены по разным: сегмент стека, сегмент данных и сегмент кода, в отличии от «плохого». Также отсутствие обязательной для COM файлов директивы ORG 100h приводит к тому, что память под управляющие структуры (PSP) не выделяется.

000001F0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000200	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000210	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000220	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000230	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000240	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000250	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000260	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000270	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000280	54 79 70 65 3A 20 50 43	0D 0A 24 54 79 70 65 3A	Type: PC..\$Type:
00000290	20 50 43 2F 58 54 0D 0A	24 54 79 70 65 3A 20 41	PC/XT..\$Type: A
000002A0	54 0D 0A 24 54 79 70 65	3A 20 50 53 32 20 D0 BC	T..\$Type: PS2
000002B0	D0 BE D0 B4 D0 B5 D0 B8	D1 8C 20 33 30 D0 0A 24
000002C0	54 79 70 65 3A 20 50 53	32 20 D0 BC D0 BE D0 B4	Type: PS2
000002D0	D0 B5 D0 B8 D1 8C 20 35	30 20 D0 B8 D0 B8 D0 B8
000002E0	20 36 30 D0 0A 24 54 79	70 65 3A 20 50 53 32 20	60..\$Type: PS2
000002F0	D0 BC D0 BE D0 B4 D0 B5	D0 B8 D1 8C 20 38 30 D0
00000300	0A 24 54 79 70 65 3A 20	50 D0 A1 6A 72 D0 0A 24	..\$Type: p-ijr..\$
00000310	54 79 70 65 3A 20 50 43	20 43 6F 6E 76 65 72 74	Type: PC Convert
00000320	69 62 6C 65 D0 0A 24 56	65 72 73 69 6F 6E 3A 20	ible..\$Version:
00000330	20 2E 20 20 D0 0A 24 4F	45 4D 3A 20 20 D0 0A 24	..\$OEM: ..\$
00000340	55 73 65 72 3A 20 20 20	20 20 20 20 20 24 00 00	User: ..\$
00000350	E9 02 01 24 0F 3C 09 76	02 04 07 04 30 C3 51 8A	0..\$.<.v....0Qè
00000360	E0 E8 EF FF 86 C4 B1 04	D2 E8 E8 E6 FF 59 C3 53	0001 à- .T0001 V}S
00000370	8A FC E8 E9 FF 88 25 4F	88 05 4F 8A C7 E8 DE FF	è"00 è%0è.0è00
00000380	88 25 4F 88 05 5B C3 51	52 32 E4 33 D2 B9 0A 00	è%0è.[QR2E3T ..
00000390	F7 F1 80 CA 30 88 14 4E	33 D2 3D 0A 00 73 F1 3C	=±C0è.N3T=..s±<
000003A0	00 74 04 0C 30 88 04 5A	59 C3 B8 00 F0 8E C0 26	.t..0è.ZY .=A è
000003B0	A0 FE FF 3C FF 74 1C 3C	FE 74 1E 3C FB 74 1A 3C	â. < t.<.t.<√t.<
000003C0	FC 74 1C 3C FA 74 1E 3C	F8 74 26 3C FD 74 28 3C	"t.<.t.<"t&<"t(<
000003D0	F9 74 2A BA 00 00 EB 2B	90 BA 0B 00 EB 25 90 BA	.t* ..0+è ..0%è
000003E0	19 00 EB 1F 90 BA 24 00	EB 19 90 BA 40 00 EB 13	..0è.è \$.0è 0.0è.
000003F0	90 BA 66 00 EB 0D 90 BA	82 00 EB 07 90 BA 90 00	è f.è.è è.0è.è è.
00000400	EB 01 90 B4 09 CD 21 C3	B4 30 CD 21 50 BE A7 00	0è.è .=- 0=1P0.
00000410	83 C6 09 E8 71 FF 58 8A	C4 83 C6 03 E8 68 FF BA	â .0q Xè-â .0h
00000420	A7 00 B4 09 CD 21 BE B7	00 83 C6 05 8A C7 E8 56	°. .=- 0.â .è 0V
00000430	FF BA B7 00 B4 09 CD 21	BF C0 00 83 C7 0B 8B C1	0. - = 0.L.è .è
00000440	F8 2C FF BA C3 F8 16 FF	83 FF 02 89 85 BA C0 00	0è.è0è.â0è.è.L.

Рис. 7 - Структура «хорошего» файла .EXE

Загрузка .COM модуля в основную память

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

После загрузки программы все 4 сегментных регистра указывают на начало единственного сегмента, то есть фактически на начало PSP. Указатель стека SP автоматически инициализируется числом 0FFFFeh. Таким образом, независимо от фактического размера программы, ей выделяется 64 Кбайт адресного пространства, всю нижнюю часть которого занимает стек (рисунок 8). Указатель команд на момент начала программы указывает на адрес 100h, с которого, соответственно, располагается код.

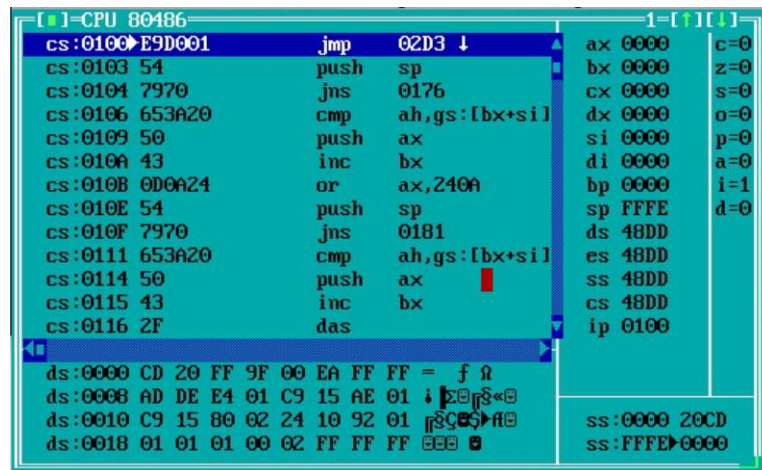


Рис. 8 - Отладка .COM модуля (с помощью TD.EXE)

2. Что располагается с адреса 0?

С адреса 0 располагается программный сегмент PSP, размером 256 байт, зарезервируемый операционной системой.

3. Какие значения имеют сегментные регистры? На какие области памяти они указывают? Сегментные регистры имеют значения 48DDh и указывают на программный сегмент PSP.

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

Стек занимает всю нижнюю часть адресного пространства. При этом сегментный регистр SS в начале программы, как и остальные регистры, указывает на PSP (48DDh), а регистр SP – на конец стека с адресом 0FFFFh.

Загрузка «хорошего» .EXE модуля в основную память

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

Система, загрузив программу в память, инициализирует сегментные регистры так, что регистры DS и ES указывают на начало PSP, CS – на начало сегмента команд, а SS – на начало сегмента стека. В указатель команд IP загружается смещение точки входа в программу, а в указатель стека SP – смещение конца сегмента стека. Таким образом, после загрузки программы в память адресуемыми оказываются все регистры, кроме сегментов данных.



Рис. 9 - Отладка «хорошего» .EXE модуля (с помощью TD.EXE)

2. На что указывают регистры DS и ES?

При запуске программы регистры DS и ES указывают на начало PSP.

3. Как определяется стек?

Оператор segment, начинающий сегмент стека, имеет описатель stack, что приводит к тому, что при загрузке программы в память регистр SS будет настроен на начало сегмента стека, а указатель стека SP —на его конец.

4. Как определяется точка входа?

Точка входа определяется с помощью директивы END.

Вывод

В ходе лабораторной работы были исследованы различия в структурах исходных текстов модулей типов .COM и .EXE, структур файлов загрузочных модулей и способов их загрузки в основную память.