

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

Аухадиев 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.

Функции и структуры данных.

Название	Назначение
TETR_TO_HEX	Перевод 4-х младших битов AL в цифру 16 с/с, представление в виде символа и запись в AL
BYTE_TO_HEX	Байт в AL переводится в два символа шестн. числа в AX
WRD_TO_HEX	Перевод в 16 с/с 16-ти разрядного числа в AX число, DI - адрес последнего символа
BYTE_TO_DEC	Перевод в 10 с/с, SI - адрес поля младшей цифры
PRINT_MESSAGE	Вывод строки на экран (функция 09h)
BEGIN	Поиск всей необходимой информации о компьютере, её обработка и вывод на экран с помощью функций выше

Последовательность действий программы.

Вызов процедуры BEGIN, которая находит и выводит на экран тип PC, серийные номера OEM и пользователя, версию ОС. Для получения необходимых данных используется информация из последнего байта ROM BIOS по адресу 0F000:0FFFEh для получения типа ПК, а также функция 30h прерывания 21h.

Выполнение работы.

1. Написание текста исходного .COM модуля lab_com.asm, его компиляция в "плохой" .EXE модуль lab_com.exe. При помощи EXE2BIN.EXE по плохому .EXE модулю был построен хороший .COM модуль lab_com.com.

2. Создание .EXE модуля lab_exe.asm, его компиляция в хороший .EXE модуль lab_exe.exe.

3. Сравнение исходных текстов lab_com.asm и lab_exe.asm.

4. Сравнение загрузочных модулей с помощью программы hexyl.

5. Исследование загрузочных модулей .COM и .EXE при помощи отладчика AFD.

Ответы на контрольные вопросы.

I. Отличия исходных текстов COM и EXE программ

1) Сколько сегментов должна содержать COM-программа?

Ответ: один сегмент

2) EXE-программа?

Ответ: Один и более сегментов

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

Ответ:

1. Директива ORG 100h, которая задаёт смещение адресации в 256 байт от нулевого адреса для PSP.

2. Директива ASSUME необходима для проверки допустимости каждого обращения к именованной ячейке памяти с учётом значения текущего сегментного регистра.

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

Ответ: Невозможно использование команд вида mov <регистр>, seg <имя сегмента>, так как COM-программа ограничена в 64 КБ, поэтому нельзя использовать команду, которая бы занимала больше или работала бы с 64-битными регистрами.

II. Отличие форматов COM и EXE программ

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

Ответ: COM-файл содержит данные и машинные команды. Код начинается с адреса 0h, но при загрузке модуля устанавливается смещение в 100h.

00000000	e9 15 01 4f 53 20 76 65	72 73 69 6f 6e 3a 20 20	***OS ve	rsion:
00000010	2e 20 0d 0a 24 4f 45 4d	3a 20 20 20 0d 0a 24 53	. __\$OEM:	__ \$\$
00000020	65 72 69 61 6c 20 6e 75	6d 62 65 72 3a 20 20 20	erial nu	mber:
00000030	20 20 20 20 0d 0a 24 50	43 20 74 79 70 65 3a 20	__ \$P	C type:
00000040	24 41 54 0d 0a 24 50 43	0d 0a 24 50 43 2f 58 54	\$AT __ \$PC	__ \$PC/XT
00000050	0d 0a 24 50 53 32 20 6d	6f 64 65 6c 20 33 30 0d	__ \$PS2 m	odel 30
00000060	0a 24 50 53 32 20 6d 6f	64 65 6c 20 35 30 20 6f	__ \$PS2 mo	del 50 o
00000070	72 20 36 30 0d 0a 24 50	53 32 20 6d 6f 64 65 6c	r 60 __ \$P	S2 model
00000080	20 38 30 0d 0a 24 50 43	6a 72 0d 0a 24 50 43 20	80 __ \$PC	jr __ \$PC
00000090	43 6f 6e 76 65 72 74 69	62 6c 65 0d 0a 24 87 ad	Converti	ble __ \$xx
000000a0	a0 e7 a5 ad a8 a5 20 e0	a5 a3 a8 e1 e2 e0 a0 20	xxxxxx x	xxxxxx
000000b0	80 95 3d 20 20 20 20 0d	0a 24 24 0f 3c 09 76 02	xx=	__ \$\$*<_v*
000000c0	04 07 04 30 c3 51 8a e0	e8 ef ff 86 c4 b1 04 d2	***0xQxx	xxxxxx*x
000000d0	e8 e8 e6 ff 59 c3 53 8a	fc e8 e9 ff 88 25 4f 88	xxxxYxSx	xxxxxx%0x
000000e0	05 4f 8a c7 e8 de ff 88	25 4f 88 05 5b c3 51 52	*0xxxxxx	%0x*[xQR
000000f0	32 e4 33 d2 b9 0a 00 f7	f1 80 ca 30 88 14 4e 33	2x3xx _0x	xxx0x*N3
00000100	d2 3d 0a 00 73 f1 3c 00	74 04 0c 30 88 04 5a 59	x=_0sx<0	t*_0x*ZY
00000110	c3 50 b4 09 cd 21 58 c3	b8 00 f0 8e c0 26 a0 fe	xPx _x!Xx	x0xxx&xx
00000120	ff ba 37 01 e8 ea ff 3c	ff 74 20 3c fe 74 22 3c	xx7*xxx<	xt <xt"<
00000130	fb 74 1e 3c fc 74 20 3c	fa 74 22 3c fc 74 24 3c	xt*<xt <	xt"<xt\$<
00000140	f8 74 26 3c fd 74 28 3c	f9 74 2a ba 46 01 eb 28	xt&<xt(<	xt*xF*x(<
00000150	90 ba 4b 01 eb 22 90 ba	41 01 eb 1c 90 ba 53 01	xxK*x"xx	A*x*xxS*
00000160	eb 16 90 ba 62 01 eb 10	90 ba 77 01 eb 0a 90 ba	***b*x*	xxw*_x_xx
00000170	86 01 eb 04 90 ba 8d 01	e8 96 ff b4 30 cd 21 50	***xxxx*	xxxx0x!P
00000180	56 8d 36 03 01 83 c6 0c	e8 63 ff 83 c6 03 8a c4	Vx6*xx	xCxxxxxx
00000190	e8 5b ff ba 03 01 e8 78	ff 5e 58 8a c7 8d 36 15	x[xx*xx	x^Xxxx6*
000001a0	01 83 c6 05 e8 47 ff ba	15 01 e8 64 ff 8a c3 e8	***xGxx	***dxxxx
000001b0	13 ff 8d 3e 1f 01 83 c7	0f 89 05 8b c1 8d 3e 1f	***>*xx	***xxx>*
000001c0	01 83 c7 14 e8 0f ff ba	1f 01 e8 44 ff c3	***xxxx	***Dxx

2) Какова структура файла "плохого" EXE? С какого адреса располагается код? Что располагается с адреса 0?

Ответ: В "плохом" EXE файле данные и код содержатся в одном сегменте.

С адреса 0h идёт таблица настроек. Код располагается с адреса 300h.

00000000	4d 5a ce 00 03 00 00 00	20 00 00 00 ff ff 00 00	MZx0*000	000xx00
00000010	00 00 a2 ba 00 01 00 00	1e 00 00 00 01 00 00 00	00xx0*00	*000*000
00000020	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00000000	00000000
*				
00000300	e9 15 01 4f 53 20 76 65	72 73 69 6f 6e 3a 20 20	***OS ve	rsion:
00000310	2e 20 0d 0a 24 4f 45 4d	3a 20 20 20 0d 0a 24 53	. __\$OEM:	__ \$\$
00000320	65 72 69 61 6c 20 6e 75	6d 62 65 72 3a 20 20 20	erial nu	mber:
00000330	20 20 20 20 0d 0a 24 50	43 20 74 79 70 65 3a 20	__ \$P	C type:
00000340	24 41 54 0d 0a 24 50 43	0d 0a 24 50 43 2f 58 54	\$AT __ \$PC	__ \$PC/XT
00000350	0d 0a 24 50 53 32 20 6d	6f 64 65 6c 20 33 30 0d	__ \$PS2 m	odel 30
00000360	0a 24 50 53 32 20 6d 6f	64 65 6c 20 35 30 20 6f	__ \$PS2 mo	del 50 o
00000370	72 20 36 30 0d 0a 24 50	53 32 20 6d 6f 64 65 6c	r 60 __ \$P	S2 model
00000380	20 38 30 0d 0a 24 50 43	6a 72 0d 0a 24 50 43 20	80 __ \$PC	jr __ \$PC
00000390	43 6f 6e 76 65 72 74 69	62 6c 65 0d 0a 24 87 ad	Converti	ble __ \$xx
000003a0	a0 e7 a5 ad a8 a5 20 e0	a5 a3 a8 e1 e2 e0 a0 20	xxxxxx x	xxxxxx
000003b0	80 95 3d 20 20 20 20 0d	0a 24 24 0f 3c 09 76 02	xx=	__ \$\$*<_v*
000003c0	04 07 04 30 c3 51 8a e0	e8 ef ff 86 c4 b1 04 d2	***0xQxx	xxxxxx*x
000003d0	e8 e8 e6 ff 59 c3 53 8a	fc e8 e9 ff 88 25 4f 88	xxxxYxSx	xxxxxx%0x
000003e0	05 4f 8a c7 e8 de ff 88	25 4f 88 05 5b c3 51 52	*0xxxxxx	%0x*[xQR
000003f0	32 e4 33 d2 b9 0a 00 f7	f1 80 ca 30 88 14 4e 33	2x3xx _0x	xxx0x*N3
00000400	d2 3d 0a 00 73 f1 3c 00	74 04 0c 30 88 04 5a 59	x=_0sx<0	t*_0x*ZY
00000410	c3 50 b4 09 cd 21 58 c3	b8 00 f0 8e c0 26 a0 fe	xPx _x!Xx	x0xxx&xx
00000420	ff ba 37 01 e8 ea ff 3c	ff 74 20 3c fe 74 22 3c	xx7*xxx<	xt <xt"<
00000430	fb 74 1e 3c fc 74 20 3c	fa 74 22 3c fc 74 24 3c	xt*<xt <	xt"<xt\$<
00000440	f8 74 26 3c fd 74 28 3c	f9 74 2a ba 46 01 eb 28	xt&<xt(<	xt*xF*x(<
00000450	90 ba 4b 01 eb 22 90 ba	41 01 eb 1c 90 ba 53 01	xxK*x"xx	A*x*xxS*
00000460	eb 16 90 ba 62 01 eb 10	90 ba 77 01 eb 0a 90 ba	***b*x*	xxw*_x_xx
00000470	86 01 eb 04 90 ba 8d 01	e8 96 ff b4 30 cd 21 50	***xxxx*	xxxx0x!P
00000480	56 8d 36 03 01 83 c6 0c	e8 63 ff 83 c6 03 8a c4	Vx6*xx	xCxxxxxx
00000490	e8 5b ff ba 03 01 e8 78	ff 5e 58 8a c7 8d 36 15	x[xx*xx	x^Xxxx6*
000004a0	01 83 c6 05 e8 47 ff ba	15 01 e8 64 ff 8a c3 e8	***xGxx	***dxxxx
000004b0	13 ff 8d 3e 1f 01 83 c7	0f 89 05 8b c1 8d 3e 1f	***>*xx	***xxx>*
000004c0	01 83 c7 14 e8 0f ff ba	1f 01 e8 44 ff c3	***xxxx	***Dxx

3) Какова структура "хорошего" EXE? Чем он отличается от файла "плохого" EXE?

Ответ: В "хорошем" EXE данные, стек и код находятся в разных сегментах.

00000000	4d 5a cd 01 02 00 01 00	20 00 11 00 ff ff 1d 00	MZx**0*0	0*0xx*0
00000010	00 01 6d 65 10 00 00 00	1e 00 00 00 01 00 72 00	0*me*000	*000*0r0
00000020	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00000000	00000000
*				
00000210	eb 5f 90 24 0f 3c 09 76	02 04 07 04 30 c3 51 8a	x_x\$*<_v	***0xQx
00000220	e0 e8 ef ff 86 c4 b1 04	d2 e8 e8 e6 ff 59 c3 53	xxxxxxxx*	xxxxxYxS
00000230	8a fc e8 e9 ff 88 25 4f	88 05 4f 8a c7 e8 de ff	xxxxxx%0	x*0xxxxx
00000240	88 25 4f 88 05 5b c3 51	52 32 e4 33 d2 b9 0a 00	x%0x*[xQ	R2x3xx_0
00000250	f7 f1 80 ca 30 88 14 4e	33 d2 3d 0a 00 73 f1 3c	xxxx0x*N	3x=_0sx<
00000260	00 74 04 0c 30 88 04 5a	59 c3 50 b4 09 cd 21 58	0t* 0x*Z	YxPx_!X
00000270	c3 b8 13 00 8e d8 b8 00	f0 8e c0 26 a0 fe ff ba	xx*0xxx0	xxx&xxxx
00000280	36 00 e8 e5 ff 3c ff 74	20 3c fe 74 22 3c fb 74	60xxx<xt	<xt"<xt
00000290	1e 3c fc 74 20 3c fa 74	22 3c fc 74 24 3c f8 74	*<xt <xt	'<xt\$<xt
000002a0	26 3c fd 74 28 3c f9 74	2a ba 45 00 eb 28 90 ba	&<xt(<xt	*xE0x(xx
000002b0	4a 00 eb 22 90 ba 40 00	eb 1c 90 ba 52 00 eb 16	J0x"xx@0	x*xxR0x*
000002c0	90 ba 61 00 eb 10 90 ba	76 00 eb 0a 90 ba 85 00	xxa0x*xx	v0x_ xxx0
000002d0	eb 04 90 ba 8c 00 e8 91	ff b4 30 cd 21 50 56 8d	x*xxx0xx	xx0!PVx
000002e0	36 02 00 83 c6 0c e8 5e	ff 83 c6 03 8a c4 e8 56	6*0xx_x^	xxx*xxxV
000002f0	ff ba 02 00 e8 73 ff 5e	58 8a c7 8d 36 14 00 83	xx*0sx^	Xxxx6*0x
00000300	c6 05 e8 42 ff ba 14 00	e8 5f ff 8a c3 e8 0e ff	x*xBxx*0	x_ xxxxx
00000310	8d 3e 1e 00 83 c7 0f 89	05 8b c1 8d 3e 1e 00 83	x>*0xxx*	xxxx>*0x
00000320	c7 14 e8 0a ff ba 1e 00	e8 3f ff 32 c0 b4 4c cd	x*x_ xx*0	x?*2xxLx
00000330	21 c3 4f 53 20 76 65 72	73 69 6f 6e 3a 20 20 2e	!x0S ver	sion: .
00000340	20 0d 0a 24 4f 45 4d 3a	20 20 20 0d 0a 24 53 65	__\$0EM:	__\$Se
00000350	72 69 61 6c 20 6e 75 6d	62 65 72 3a 20 20 20 20	rial num	ber:
00000360	20 20 20 0d 0a 24 50 43	20 74 79 70 65 3a 20 24	__\$PC	type: \$
00000370	41 54 0d 0a 24 50 43 0d	0a 24 50 43 2f 58 54 0d	AT__\$PC_	__\$PC/XT_
00000380	0a 24 50 53 32 20 6d 6f	64 65 6c 20 33 30 0d 0a	__\$PS2 mo	del 30_
00000390	24 50 53 32 20 6d 6f 64	65 6c 20 35 30 20 6f 72	__\$PS2 mod	el 50 or
000003a0	20 36 30 0d 0a 24 50 53	32 20 6d 6f 64 65 6c 20	60___\$PS	2 model
000003b0	38 30 0d 0a 24 50 43 6a	72 0d 0a 24 50 43 20 43	80___\$PCj	r___\$PC C
000003c0	6f 6e 76 65 72 74 69 62	6c 65 0d 0a 24	onvertib	le__\$

III. Загрузка COM модуля в основную память

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

Ответ:

1. Определяется сегментный адрес участка ОП, у которого достаточно места для загрузки программы

2. Создаётся блок памяти для PSP и программы

3. Загружается COM-файл с адреса 100h

4. Сегментные регистры CS, DS, ES, SS устанавливаются на начало PSP (0h)

5. Регистр SP устанавливается на конец PSP (FFh)

6. В стек записывается значение 0000

7. В регистр IP записывается значение 100h. С такого же адреса и начинается код.

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

AX 0000	SI 0000	CS 119C	IP 0100	Stack +0 0000	FLAGS 0200
BX 0000	DI 0000	DS 119C		+2 0000	
CX 01CE	BP 0000	ES 119C	HS 119C	+4 0000	OF DF IF SF ZF AF PF CF
DX 0000	SP FFF5	SS 119C	FS 119C	+6 0000	0 0 1 0 0 0 0 0

0100 E91501 JMP 0218

0103 4F DEC DI

0104 53 PUSH BX

0105 207665 AND [BP+65],DH

0108 7273 JC 017D

010A 696F6E3A20 IMUL BP,[BX+6E],203A

010F 20E200D AND [0D20],CH

0113 0A24 OR AH,[SI]

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
DS:0000	CD	20	9C	11	00	EA	FD	FF	AD	DE	ED	04	92	01	00	00
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

2) Что располагается с адреса 0?

Ответ: Сегмент PSP

3) Какие значения имеют сегментные регистры? На какие области памяти ни указывают?

Ответ: При загрузке программы они указывают на начало PSP

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

Ответ: Стек расположен между адресами SS:0000h и SS:FFFFh

IV. Загрузка "хорошего" EXE модуля в основную память

1) Как загружается "хороший" EXE? Какие значения имеют сегментные регистры?

Ответ: Регистры DS и ES указывают на начало блока PSP, регистр CS указывает на начало сегмента кода, а регистр SS - на начало сегмента стека.

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

AX 0000 SI 0000 CS 11AC IP 0010 Stack +0 0000 FLAGS 0200
 BX 0000 DI 0000 DS 119C +2 0000
 CX 01CD BP 0000 ES 119C HS 119C +4 0000 OF DF IF SF ZF AF PF CF
 DX 0000 SP 0100 SS 11C9 FS 119C +6 0000 0 0 1 0 0 0 0 0

CMD >

Address	Disassembly	Comment	Hex
0010 EB5F	JMP	0071	CD 20 38 23 00 EA FD FF
0012 90	NOP		AD DE ED 04 92 01 00 00
0013 240F	AND	AL, 0F	18 01 10 01 18 01 92 01
0015 3C09	CMP	AL, 09	04 FF FF FF FF FF FF FF
0017 7602	JNA	001B	FF FF FF FF 96 11 C4 FF
0019 0407	ADD	AL, 07	92 01 14 00 18 00 9C 11
001B 0430	ADD	AL, 30	FF FF FF FF 00 00 00 00
001D C3	RET		05 00 00 00 00 00 00 00

Memory dump (DS:0000):

Address	Hex	ASCII
DS:0000	CD 20 38 23 00 EA FD FF	AD DE ED 04 92 01 00 00
DS:0010	18 01 10 01 18 01 92 01	04 FF FF FF FF FF FF FF
DS:0020	04 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	FF FF FF FF 00 00 00 00	05 00 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

2) На что указывают регистры DS и ES?

Ответ: На начало блока PSP

3) Как определяется стек?

Ответ: SS - начало сегмента, SS:SP - конец

4) Как определяется точка входа?

Ответ: параметром после директивы END, в качестве которого нужно передать метку, с которой программа начнёт выполнение команд.

Заключение.

Были изучены различия в структурах исходных текстов модулей .COM и .EXE, структур файлов загрузочных модулей и способов их загрузки в основную память.