

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

Камакин Д.В.

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.

TETR_TO_HEX — перевод значения 4-ёх младших битов в регистре **AL** в цифру 16-ой **CC** (остаётся в регистре **AL**).

BYTE_TO_HEX — перевод значения байта из регистра **AL** в число 16-ой **CC**. Результат записывается в **AL** и **AH**.

WRD_TO_HEX — перевод значения слова в регистре **AX** в число 16-ой **CC** и запись в виде 4 символов по адресу, записанному в регистре **DI**.

BYTE_TO_DEC — перевод байта в регистре **AL** в 10-ую **CC**. Символы записываются по адресу, записанному в регистре **SI**.

OUTPUT — вызов функции 09h прерывания **int 21h** (посылает строку на стандартный вывод).

Последовательность действий.

Вызываем функцию 30h прерывания **int 21h** для получения текущего номера версии **DOS**. Затем переводим результат работы функции в 16/10 **CC** и записываем в соответствующие строки. Для этого используем функции **BYTE_TO_DEC** и **BYTE_TO_HEX**, а также **WRD_TO_HEX**.

Для вывода информации о типе **PC** читаем содержимое предпоследнего байта **ROM BIOS** и, сравнивая его значение с табличными, получаем строковое представление названия модели.

Выводим все полученные строки при помощи функции **OUTPUT** и завершаем программу.

Результаты исследования проблем.

Отличия исходных текстов COM и EXE программ

1) Сколько сегментов должна содержать COM-программа?

Только один.

2) EXE-программа?

Не менее одного.

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

ORG 100H, поскольку в начале COM-программы определён 256-байтовый PSP (префикс программного сегмента), а значит необходимо обеспечить смещение на 100h байт от начала.

Директива ASSUME необходима для проверки допустимости каждого обращения к именованной ячейке памяти с учётом значения текущего сегментного регистра. Кроме того, требуется использования директивы END для завершения программы.

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

Нельзя использовать seg из-за отсутствия таблицы настройки.

Отличия форматов файлов COM и EXE модулей

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

```

00000000 E9 1B 01 4D 61 63 68 69 6E 65 20 74 79 70 65 3A 20 24 53 79 73 ..Machine type: $Sys
00000015 74 65 6D 20 76 65 72 73 69 6F 6E 3A 20 20 2E 20 20 0D 0A 24 4F tem version: . . . $0
0000002A 45 4D 3A 20 20 0D 0A 24 55 73 65 72 20 73 65 72 69 61 6C 20 6E EM: ..$User serial n
0000003F 75 6D 62 65 72 3A 20 20 20 20 20 20 20 0D 0A 24 41 54 0D 0A 24 umber: ..$AT..$
00000054 50 43 0D 0A 24 50 43 2F 58 54 0D 0A 24 50 53 32 20 6D 6F 64 65 PC..$PC/XT..$PS2 mode
00000069 6C 20 33 30 0D 0A 24 50 53 32 20 6D 6F 64 65 6C 20 35 30 20 6F l 30..$PS2 model 50 o
0000007E 72 20 36 30 0D 0A 24 50 53 32 20 6D 6F 64 65 6C 20 38 30 0D 0A r 60..$PS2 model 80..
00000093 24 50 43 6A 72 0D 0A 24 50 43 20 43 6F 6E 76 65 72 74 69 62 6C $PCjr..$PC Convertibl
000000A8 65 0D 0A 24 20 20 20 20 28 75 6E 6B 6F 77 6E 20 74 79 70 65 29 e..$ (unkown type)
000000BD 0D 0A 24 24 0F 3C 09 76 02 04 07 04 30 C3 51 8A E0 E8 EF FF 86 ..$.<.v....0.Q.....
000000D2 C4 B1 04 D2 E8 E8 E6 FF 59 C3 53 8A FC E8 E9 FF 88 25 4F 88 05 .....Y.S.....%0..
000000E7 4F 8A C7 E8 DE FF 88 25 4F 88 05 5B C3 51 52 32 E4 33 D2 B9 0A 0.....%0...[.QR2.3...
000000FC 00 F7 F1 80 CA 30 88 14 4E 33 D2 3D 0A 00 73 F1 3C 00 74 04 0C .....0..N3.=..s.<.t..
00000111 30 88 04 5A 59 C3 50 B4 09 CD 21 58 C3 B4 30 CD 21 8A F4 BF 12 0..ZY.P...!X..0.!....
00000126 01 E8 CA FF AD 88 65 10 8A C6 E8 C1 FF AD 88 65 12 8A C7 E8 8F .....e.....e.....
0000013B FF BF 29 01 88 45 05 88 65 06 BF 32 01 8A C3 E8 7E FF 88 45 15 ..)E..e..2....~..E.
00000150 88 65 16 8B C1 83 C7 1A E8 81 FF BA 03 01 E8 B6 FF B8 00 F0 8E .e.....
00000165 C0 26 A0 FE FF 3C FF 74 23 3C FE 74 2B 3C FB 74 27 3C FC 74 1D .&...<.t#<.t+<.t'<.t.
0000017A 3C FA 74 25 3C FC 74 27 3C F8 74 29 3C FD 74 2B 3C F9 74 2D EB <.t%<.t'<.t)<.t+<.t-.
0000018F 31 90 BA 54 01 EB 39 90 BA 4F 01 EB 33 90 BA 59 01 EB 2D 90 BA 1..T..9..0..3..Y..-..
000001A4 61 01 EB 27 90 BA 70 01 EB 21 90 BA 85 01 EB 1B 90 BA 94 01 EB a...'..p...!.....
000001B9 15 90 BA 9B 01 E9 56 FF E8 07 FF BF AC 01 88 05 88 65 01 BA AC .....V.....e...
000001CE 01 E8 45 FF BA 12 01 E8 3F FF BA 29 01 E8 39 FF BA 32 01 E8 33 ..E.....?...)..9..2..3
000001E3 FF 32 C0 B4 4C CD 21 .2..L.!

```

Рис. 1. COM файл в шестнадцатеричной форме.

В файле COM содержится только код и данные программы. Код располагается с 0 адреса.

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

Код и данные расположены в одном сегменте, начиная с адреса 300h. С адреса 0 расположена таблица настроек и заголовок.

dkamakin@dkamakin:~/Documents/Lab05/MASM\$ hexyl SRC_COM.EXE

00000000	4d 5a ea 00 03 00 00 00	20 00 00 00 ff ff 00 00	MZx0*000	000xx00
00000010	00 00 a2 12 00 01 00 00	1e 00 00 00 01 00 00 00	00x0*00	*000*000
00000020	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00000000	00000000
*				
00000300	e9 1b 01 4d 61 63 68 69	6e 65 20 74 79 70 65 3a	x**Machi	ne type:
00000310	20 24 53 79 73 74 65 6d	20 76 65 72 73 69 6f 6e	\$System	version
00000320	3a 20 20 2e 20 20 0d 0a	24 4f 45 4d 3a 20 20 0d	: . __	\$OEM: _
00000330	0a 24 55 73 65 72 20 73	65 72 69 61 6c 20 6e 75	_\$User s	erial nu
00000340	6d 62 65 72 3a 20 20 20	20 20 20 20 0d 0a 24 41	mber: _	\$A
00000350	54 0d 0a 24 50 43 0d 0a	24 50 43 2f 58 54 0d 0a	T_\$PC__	\$PC/XT__
00000360	24 50 53 32 20 6d 6f 64	65 6c 20 33 30 0d 0a 24	\$PS2 mod	el 30__\$
00000370	50 53 32 20 6d 6f 64 65	6c 20 35 30 20 6f 72 20	PS2 mode	l 50 or
00000380	36 30 0d 0a 24 50 53 32	20 6d 6f 64 65 6c 20 38	60__\$PS2	model 8
00000390	30 0d 0a 24 50 43 6a 72	0d 0a 24 50 43 20 43 6f	0__\$PCjr	__\$PC Co
000003a0	6e 76 65 72 74 69 62 6c	65 0d 0a 24 20 20 20 20	nvertibl	e__\$
000003b0	28 75 6e 6b 6f 77 6e 20	74 79 70 65 29 0d 0a 24	(unkown	type)__\$
000003c0	24 0f 3c 09 76 02 04 07	04 30 c3 51 8a e0 e8 ef	\$*<_v***	*0xQxxxx
000003d0	ff 86 c4 b1 04 d2 e8 e8	e6 ff 59 c3 53 8a fc e8	xxxx*xxx	xxYxSxxx
000003e0	e9 ff 88 25 4f 88 05 4f	8a c7 e8 de ff 88 25 4f	xxx%0*x*0	xxxxxx%0
000003f0	88 05 5b c3 51 52 32 e4	33 d2 b9 0a 00 f7 f1 80	x*[xQR2x	3xx_0xxx
00000400	ca 30 88 14 4e 33 d2 3d	0a 00 73 f1 3c 00 74 04	x0*x*N3x=	_0sx<0t*
00000410	0c 30 88 04 5a 59 c3 50	b4 09 cd 21 58 c3 b4 30	_0x*ZYxP	x_x!Xxx0
00000420	cd 21 8a f4 bf 12 01 e8	ca ff ad 88 65 10 8a c6	x!xxxx*x	xxxxe*xx
00000430	e8 c1 ff ad 88 65 12 8a	c7 e8 8f ff bf 29 01 88	xxxxxe*x	xxxxx)*x
00000440	45 05 88 65 06 bf 32 01	8a c3 e8 7e ff 88 45 15	E*x*e*x2*	xxx~xxE*
00000450	88 65 16 8b c1 83 c7 1a	e8 81 ff ba 03 01 e8 b6	xe*xxxx*	xxxx*x*xx
00000460	ff b8 00 f0 8e c0 26 a0	fe ff 3c ff 74 23 3c fe	xx0xxx&x	xx<xt#<x
00000470	74 2b 3c fb 74 27 3c fc	74 1d 3c fa 74 25 3c fc	t+<xt'<x	t*<xt%<x
00000480	74 27 3c f8 74 29 3c fd	74 2b 3c f9 74 2d eb 31	t'<xt)<x	t+<xt-x1
00000490	90 ba 54 01 eb 39 90 ba	4f 01 eb 33 90 ba 59 01	xxT*x9xx	0*x3xxY*
000004a0	eb 2d 90 ba 61 01 eb 27	90 ba 70 01 eb 21 90 ba	x-xxa*x'	xxp*x!xx
000004b0	85 01 eb 1b 90 ba 94 01	eb 15 90 ba 9b 01 e9 56	x*x*xxxx*	xxxx*xV
000004c0	ff e8 07 ff bf ac 01 88	05 88 65 01 ba ac 01 e8	xx*xxxx*	*xe*xxx*
000004d0	45 ff ba 12 01 e8 3f ff	ba 29 01 e8 39 ff ba 32	Exx*xx?x	x)*x9xx2
000004e0	01 e8 33 ff 32 c0 b4 4c	cd 21	*x3x2xxL	x!

Рис. 2. «Плохой» EXE файл в шестнадцатеричной форме.

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

В «хорошем» EXE код и данные разделены по сегментам, таблица настроек корректна (начинается с адреса 0). У «хорошего» и «плохого» EXE файла код начинается с разных смещений, поскольку у первого нет директивы ORG 100h.

```

dkamakin@dkamakin:~/Documents/LabOS/MASM$ hexyl SRC_EXE.EXE
00000000  4d 5a ff 01 02 00 01 00  20 00 11 00 ff ff 20 00  MZ***0*0  0*0*x 0
00000010  00 01 00 23 10 00 00 00  1e 00 00 00 01 00 72 00  0*0#*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  xxxxxxx*  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!X
00000270  c3 b8 14 00 8e d8 ba 02  00 e8 ee ff b4 30 cd 21  xx*0xxx*  0xxxx0x!
00000280  8a f4 bf 11 00 e8 bf ff  ad 88 65 10 8a c6 e8 b6  xxx*0xxx  xxe*xxxx
00000290  ff ad 88 65 12 8a c7 e8  84 ff bf 28 00 88 45 05  xxxe*xxx  xxx(0xE*
000002a0  88 65 06 bf 31 00 8a c3  e8 73 ff 88 45 15 88 65  xe*x10xx  sxxxExe
000002b0  16 8b c1 83 c7 1a e8 76  ff b8 00 f0 8e c0 26 a0  *xxxx*v  xx0xxx&x
000002c0  fe ff 3c ff 74 23 3c fe  74 2b 3c fb 74 27 3c fc  xx<xt#<x  t+<xt'<x
000002d0  74 1d 3c fa 74 25 3c fc  74 27 3c f8 74 29 3c fd  t*<xt%<x  t'<xt)<x
000002e0  74 2b 3c f9 74 2d eb 31  90 ba 53 00 eb 39 90 ba  t+<xt-x1  xxS0x9xx
000002f0  4e 00 eb 33 90 ba 58 00  eb 2d 90 ba 60 00 eb 27  N0x3xxX0  x-xx'0x'
00000300  90 ba 6f 00 eb 21 90 ba  84 00 eb 1b 90 ba 93 00  xx00x!xx  x0x*xxx0
00000310  eb 15 90 ba 9a 00 e9 51  ff e8 02 ff bf ab 00 88  x*xxx0xQ  xx*xxx0x
00000320  05 88 65 01 ba ab 00 e8  40 ff ba 11 00 e8 3a ff  *xe*xxx0  @xx*0x:x
00000330  ba 28 00 e8 34 ff ba 31  00 e8 2e ff 32 c0 b4 4c  x(0x4xx1  0x.x2xxL
00000340  cd 21 4d 61 63 68 69 6e  65 20 74 79 70 65 3a 20  x!Machin  e type:
00000350  24 53 79 73 74 65 6d 20  76 65 72 73 69 6f 6e 3a  $System  version:
00000360  20 20 2e 20 20 0d 0a 24  4f 45 4d 3a 20 20 0d 0a  . __$ OEM: __
00000370  24 55 73 65 72 20 73 65  72 69 61 6c 20 6e 75 6d  $User se  rial num
00000380  62 65 72 3a 20 20 20 20  20 20 20 0d 0a 24 41 54  ber: __$AT
00000390  0d 0a 24 50 43 0d 0a 24  50 43 2f 58 54 0d 0a 24  __$PC__$  PC/XT__$
000003a0  50 53 32 20 6d 6f 64 65  6c 20 33 30 0d 0a 24 50  PS2 mode  l 30__$P
000003b0  53 32 20 6d 6f 64 65 6c  20 35 30 20 6f 72 20 36  S2 model  50 or 6
000003c0  30 0d 0a 24 50 53 32 20  6d 6f 64 65 6c 20 38 30  0__$PS2  model 80
000003d0  0d 0a 24 50 43 6a 72 0d  0a 24 50 43 20 43 6f 6e  __$PCjr_  _$PC Con
000003e0  76 65 72 74 69 62 6c 65  0d 0a 24 20 20 20 20 28  vertible  __$ (
000003f0  75 6e 6b 6f 77 6e 20 74  79 70 65 29 0d 0a 24  unkown t  ype)__$

```

Рис. 3. «Хороший» EXE файл в шестнадцатеричной форме.

Загрузка СОМ модуля в основную память

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

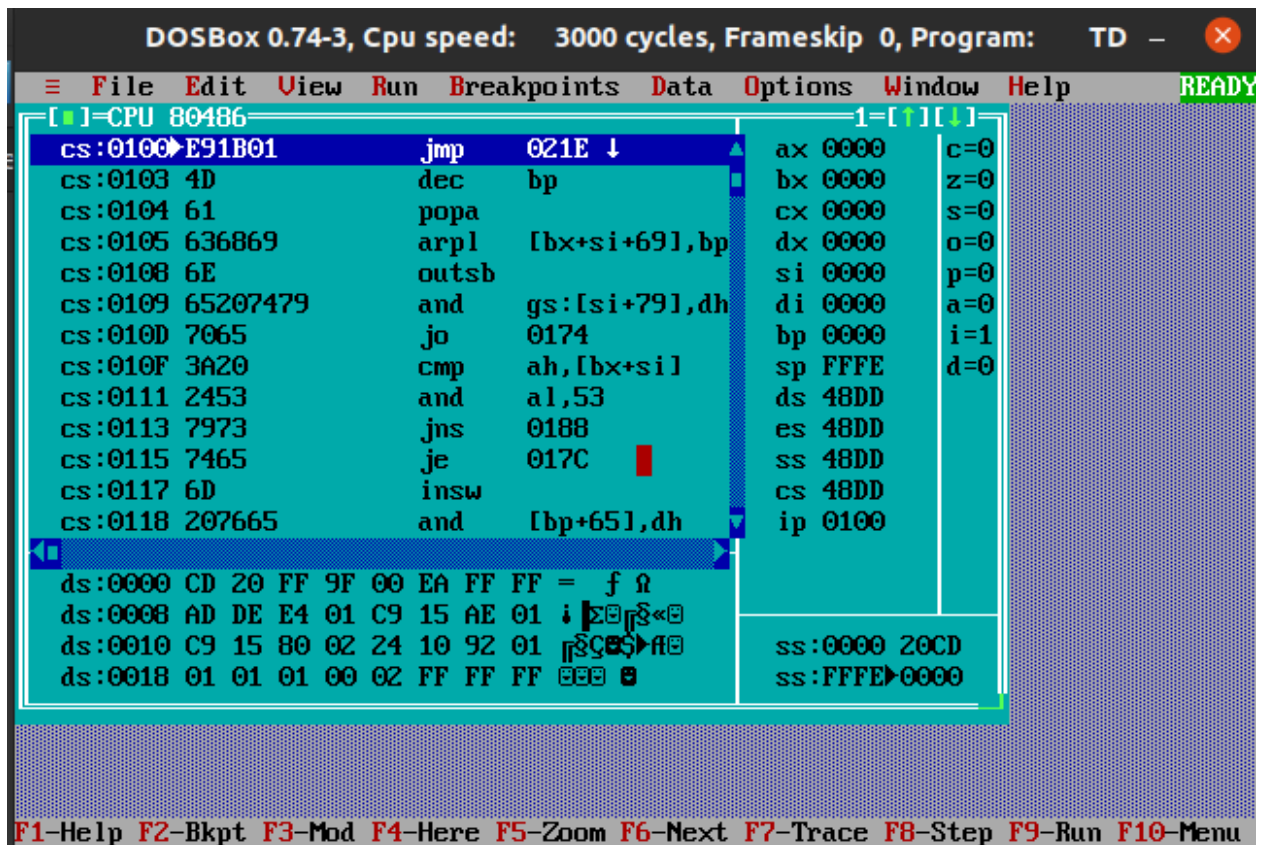


Рис. 4. Результат работы программы TD для COM файла.

Сначала помещается PSP в подходящее по размеру место, а затем сам код (значит он расположен с адреса 100h).

2) Что располагается с адреса 0?

PSP

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

DS,ES,CS и SS указывают на PSP

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

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

Загрузка «хорошего» EXE модуля в основную память

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

Считывается информация из заголовков EXE, в IP загружается смещение точки входа. CS — начало сегмента кода, DS и ES — начало блока PSP, SS — начало стека.

2) На что указывают регистры DS и ES?

На начало блока PSP

3) Как определяется стек?

Директивой .stack, SS будет указывать на начало, SS:SP на конец.

4) Как определяется точка входа?

Директивой END (метка указывает на точку входа)

План загрузки модуля .COM в основную память

1. В основной памяти ищется свободный сегмент, в который будет загружена программа.
2. В начало сегмента загружается блок PSP (100h), после которого идёт COM модуль.

Заключение

Исследованы различия в структурах исходных текстов модулей типов .COM и .EXE, структуры файлов загрузочных модулей и способы их загрузки в основную память.