

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

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

Студент гр. 8382

Кобенко В.П.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Исследование различий в структурах исходных текстов модулей .COM и .EXE, структур файлов загрузочных модулей и способов их загрузки в основную память.

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.

Постановка задачи.

Требуется реализовать текст исходного .COM модуля, который определяет тип PC и версию системы. Ассемблерная программа должна

читать содержимое предпоследнего байта ROM BIOS, по таблице, сравнивая коды, определять тип PC и выводить строку с названием модели. Если код не совпадает ни с одним значением, то двоичный код переводиться 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.

Процедуры используемые в программе.

TETR_TO_HEX – Используется для перевода половины байта в шестнадцатеричную систему счисления.

BYTE_TO_HEX – Используется для перевода байта регистра AL в шестнадцатеричную систему счисления, помещая результат в AX.

WRD_TO_HEX – Используется для перевода двух байт регистра AX в шестнадцатеричную систему счисления, помещая результат в регистр DI.

BYTE_TO_DEC – Используется для перевода байта регистра AL в десятичную систему счисления, помещая результат в SI.

TYPE_IBM_PC – Определяет тип IBM PC.

VERS_DOS – Определяет версию MS DOS.

OEM_DOS – Определяет серийный номер OEM.

USER_DOS - Определяет серийный номер пользователя.

PRINT – Вывод на экран.

Структуры данных.

Таблица 1 – Структуры данных

Название поля данных	Тип	Назначение
_type	db	Тип IBM PC
_PC	db	PC
_PC_XT	db	PC/XT
_AT	db	AT
_PS2_30	db	PS2 model 30
_PS2_50_60	db	PS2 model 50 or 60
_PS2_80	db	PS2 model 80
_PCjr	db	PCjr
_PC_Conv	db	PC Convertible
_ver	db	Version number MSDOS
_oem	db	Serial number OEM
_user	db	Serial user s number

Ход работы.

Шаг 1. Запуск «хорошего» .COM модуля.



```
C:\>lab1_1.com
TYPE IMB PC
PC/XT
Version number MSDOS:5.0
Serial number OEM: 255
Serial user s number: 000000
C:\>
```

Рисунок 1 – «Хороший» .COM модуль

Запуск «плохого» .EXE модуля.



Рисунок 2 – «Плохой» .EXE модуль

Шаг 2. Запуск «хорошего» .EXE модуля.

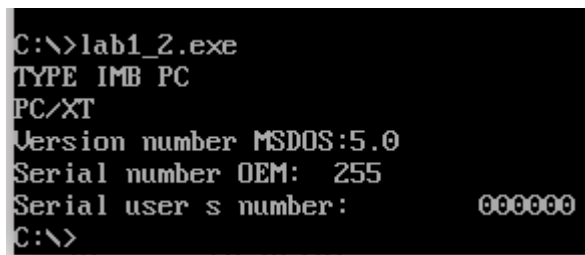


Рисунок 3 – «Хороший» .EXE модуль

Шаг 3. Ответы на контрольные вопросы. Отличия исходных текстов COM и EXE программ.

1) **Сколько сегментов должна содержать COM-программа?**

Один сегмент.

2) **EXE программа?**

EXE программа может содержать больше одного сегмента.

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

Директива `ORG 100h` (смещение `100h`), так как при загрузке COM-файла в память DOS занимает первые 256 байт (`100h`) блоком данных PSP и располагает код программы только после этого блока. Директива `ASSUME`, ставящая в соответствие начало программы сегментам кода и данных.

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

Нет, не все, так как в отличие от EXE-программы, в которой существует таблица настроек (таблица разметки), называемая Relocation Table, COM-программа ею не располагает. Адреса сегментов определяются загрузчиком в момент запуска программы на основе информации о местоположении полей адресов в файле из Relocation Table. Следовательно, в связи с отсутствием этой таблицы в COM-программах, команды вида `mov [регистр], seg [сегмент]` недопустимы.

Шаг 4. .COM модуль в шестнадцатеричном виде.

vlad@vlad-GL62M-7RDX:~/OS\$ hexyl LAB1_1.COM			
00000000	e9 7d 01 54 59 50 45 20	49 4d 42 20 50 43 20 0d	x)*TYPE IMB PC _
00000010	0a 24 50 43 0d 0a 24 50	43 2f 58 54 0d 0a 24 41	_SPC__SP C/XT__SA
00000020	54 0d 0a 24 50 53 32 20	6d 6f 64 65 6c 20 33 30	T__SPS2 model 30
00000030	0d 0a 24 50 53 32 20 6d	6f 64 65 6c 20 35 30 20	_SPS2 m odel 50
00000040	6f 72 20 36 30 0d 0a 24	50 53 32 20 6d 6f 64 65	or 60__SPS2 mode
00000050	6c 20 38 30 0d 0a 24 50	43 6a 72 0d 0a 24 50 43	l 80__SPCjr__SPC
00000060	20 43 6f 6e 76 65 72 74	69 62 6c 65 0d 0a 24 56	Convert ible__SV
00000070	65 72 73 69 6f 6e 20 6e	75 6d 62 65 72 20 4d 53	ersion n umber MS
00000080	44 4f 53 3a 20 2e 20 20	20 20 0d 0a 24 53 65 72	DOS: . __Ser
00000090	69 61 6c 20 6e 75 6d 62	65 72 20 4f 45 4d 3a 20	ial numb er OEM:
000000a0	20 20 20 0d 0a 24 53 65	72 69 61 6c 20 75 73 65	__Se rial use
000000b0	72 20 73 20 6e 75 6d 62	65 72 3a 20 20 20 20 20	r s numb er:
000000c0	20 20 20 20 20 20 20 0d	0a 24 24 0f 3c 09 76 02	__SP*<_v*
000000d0	04 07 04 30 c3 51 8a e0	e8 ef ff 86 c4 b1 04 d2	***0xQxx xxxxxx*
000000e0	e8 e8 e6 ff 59 c3 53 8a	fc e8 e9 ff 88 25 4f 88	xxxxYx\$ xxxxxx%0x
000000f0	05 4f 8a c7 e8 de ff 88	25 4f 88 05 5b c3 51 52	*0xxxxxx %0x*[xQR
00000100	32 e4 33 d2 b9 0a 00 f7	f1 80 ca 30 88 14 4e 33	2x3xx_0x xxx0x*N3
00000110	d2 3d 0a 00 73 f1 3c 00	74 04 0c 30 88 04 5a 59	x=_0sx<0 t*_0x*ZY
00000120	c3 1e b8 00 f0 8e d8 2b	db b7 fe 1f c3 50 56 be	x*x0xxxx+ xxx*xPVx
00000130	6f 01 83 c6 15 e8 c6 ff	be 6f 01 83 c6 17 8a c4	0*xxxxxx x0*xxxxx
00000140	e8 bb ff 5e 58 c3 50 53	56 be 8d 01 83 c6 16 8a	xxx^XxPS Vxxx*xx*
00000150	c7 e8 aa ff 5e 5b 58 c3	53 51 57 50 bf a6 01 83	xxxx^[Xx SQWPxxx*
00000160	c7 22 8b c1 e8 7f ff 8a	c3 e8 69 ff bf a6 01 83	x"xxxxxx xxlxxx*
00000170	c7 1d 89 05 58 5f 59 5b	c3 50 b4 09 cd 21 58 c3	xxx*X_V[xPx_!Xx
00000180	e8 9e ff ba 03 01 e8 f0	ff ba 12 01 80 ff ff 74	xxxx*xxx xx*xxxxt
00000190	47 ba 17 01 80 ff fe 74	3f ba 17 01 80 ff fb 74	Gx*xxxxt ?x*xxxxt
000001a0	37 ba 1f 01 80 ff fc 74	2f ba 24 01 80 ff fa 74	7x*xxxxt /x\$*xxxxt
000001b0	27 ba 33 01 80 ff fc 74	1f ba 48 01 80 ff f8 74	'x3*xxxxt *xH*xxxxt
000001c0	17 ba 57 01 80 ff fd 74	0f ba 5e 01 80 ff f9 74	*xW*xxxxt *x^*xxxxt
000001d0	07 8a c7 e8 ff fe 8b d0	e8 9e ff b4 30 cd 21 e8	*xxxxxxx xxxx0x!x
000001e0	4b ff e8 61 ff e8 70 ff	ba 6f 01 e8 8b ff ba 8d	Kxxaxpx xo*xxxxx
000001f0	01 e8 85 ff ba a6 01 e8	7f ff 32 c0 b4 4c cd 21	*xxxxxxx *x2xLx!

Рисунок 4 - .COM модуль в шестнадцатеричном виде

«Плохой» .EXE модуль в шестнадцатеричном виде.

vlad@vlad-GL62M-7RDX:~/OS\$ hexyl LAB1_1.EXE																		
00000000	4d	5a	00	01	03	00	00	00	20	00	00	00	ff	ff	00	00	MZ0**000	000**00
00000010	00	00	d5	5f	00	01	00	00	1e	00	00	00	01	00	00	00	00x_0*00	*000*000
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00000000	00000000
*																		
00000300	e9	7d	01	54	59	50	45	20	49	4d	42	20	50	43	20	0d	x}*TYPE	IMB PC
00000310	0a	24	50	43	0d	0a	24	50	43	2f	58	54	0d	0a	24	41	__SPC__SP	C/XT__SA
00000320	54	0d	0a	24	50	53	32	20	6d	6d	6f	64	65	6c	20	33	T__SPS2	model 30
00000330	0d	0a	24	50	53	32	20	6d	6f	64	65	6c	20	35	30	20	__SPS2 m	odel 50
00000340	6f	72	20	36	30	0d	0a	24	50	53	32	20	6d	6f	64	65	or 60_\$	PS2 mode
00000350	6c	20	38	30	0d	0a	24	50	43	6a	72	0d	0a	24	50	43	l 80_\$P	Cjr__SPC
00000360	20	43	6f	6e	76	65	72	74	69	62	6c	65	0d	0a	24	56	Convert	ible_\$V
00000370	65	72	73	69	6f	6e	20	6e	75	6d	62	65	72	20	4d	53	ersion n	umber MS
00000380	44	4f	53	3a	20	2e	20	20	20	20	0d	0a	24	53	65	72	DOS: .	__\$Ser
00000390	69	61	6c	20	6e	75	6d	62	65	72	20	4f	45	4d	3a	20	ial numb	er OEM:
000003a0	20	20	20	0d	0a	24	53	65	72	69	61	6c	20	75	73	65	__\$Se	rial use
000003b0	72	20	73	20	6e	75	6d	62	65	72	3a	20	20	20	20	20	r s numb	er:
000003c0	20	20	20	20	20	20	20	0d	0a	24	24	0f	3c	09	76	02	__\$*<	_v*
000003d0	04	07	04	30	c3	51	8a	e0	e8	ef	ff	86	c4	b1	04	d2	***0xQx	xxxxxxx
000003e0	e8	e8	e6	ff	59	c3	53	8a	fc	e8	e9	ff	88	25	4f	88	xxxxYxS	xxxxx%0x
000003f0	05	4f	8a	c7	e8	de	ff	88	25	4f	88	05	5b	c3	51	52	*0xxxxxx	%0x*[xQR
00000400	32	e4	33	d2	b9	0a	00	f7	f1	80	ca	30	88	14	4e	33	2x3xx_0x	xxx0x*N3
00000410	d2	3d	0a	00	73	f1	3c	00	74	04	0c	30	88	04	5a	59	x=_0sx<0	t* 0x*ZY
00000420	c3	1e	b8	00	f0	8e	d8	2b	db	b7	fe	1f	c3	50	56	be	*x0xxx+	xxx*xPVx
00000430	6f	01	83	c6	15	e8	c6	ff	be	6f	01	83	c6	17	8a	c4	o*x*x*x*x	xo*x*x*x*x
00000440	e8	bb	ff	5e	58	c3	50	53	56	be	8d	01	83	c6	16	8a	xxx^XxPS	Vxxx*x*x*x
00000450	c7	e8	aa	ff	5e	5b	58	c3	53	51	57	50	bf	a6	01	83	xxxx^[Xx	SQWPxxx
00000460	c7	22	8b	c1	e8	7f	ff	8a	c3	e8	69	ff	bf	a6	01	83	x"xxxxxx	xxixxxx
00000470	c7	1d	89	05	58	5f	59	5b	c3	50	b4	09	cd	21	58	c3	*x*x_X_Y[xPx_x!Xx
00000480	e8	9e	ff	ba	03	01	e8	f0	ff	ba	12	01	80	ff	ff	74	xxxx*x*x	xx*xxxxt
00000490	47	ba	17	01	80	ff	fe	74	3f	ba	17	01	80	ff	fb	74	Gx*xxxxt	?x*xxxxt
000004a0	37	ba	1f	01	80	ff	fc	74	2f	ba	24	01	80	ff	fa	74	7x*xxxxt	/x\$*xxxxt
000004b0	27	ba	33	01	80	ff	fc	74	1f	ba	48	01	80	ff	f8	74	'x3*xxxxt	*xH*xxxxt
000004c0	17	ba	57	01	80	ff	fd	74	0f	ba	5e	01	80	ff	f9	74	*xW*xxxxt	*x^*xxxxt
000004d0	07	8a	c7	e8	ff	fe	8b	d0	e8	9e	ff	b4	30	cd	21	e8	*xxxxxxx	xxxx0x!x
000004e0	4b	ff	e8	61	ff	e8	70	ff	ba	6f	01	e8	8b	ff	ba	8d	Kxxaxpx	xo*xxxxxx
000004f0	01	e8	85	ff	ba	a6	01	e8	7f	ff	32	c0	b4	4c	cd	21	*xxxxxx	*x2xxLx!

Рисунок 5 - «Плохой» .EXE модуль в шестнадцатеричном виде

«Хороший» .EXE модуль в шестнадцатеричном виде.

00000000	4d 5a 07 00 03 00 01 00	20 00 21 00 ff ff 21 00	MZ*0*0*0 0!0**!0
00000010	00 02 71 80 00 00 00 00	1e 00 00 00 01 00 ba 00	0*q*0000 *000*0*x0
00000020	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00000000 00000000
*			
00000200	e9 b6 00 24 0f 3c 09 76	02 04 07 04 30 c3 51 8a	**0\$*<_v ****0*xQx
00000210	e0 e8 ef ff 86 c4 b1 04	d2 e8 e8 e6 ff 59 c3 53	*****YxS
00000220	8a fc e8 e9 ff 88 25 4f	88 05 4f 8a c7 e8 de ff	*****%0 x*0*****
00000230	88 25 4f 88 05 5b c3 51	52 32 e4 33 d2 b9 0a 00	%0x*[xQ R2x3xx_0
00000240	f7 f1 80 ca 30 88 14 4e	33 d2 3d 0a 00 73 f1 3c	*****0x*N 3x=_0sx<
00000250	00 74 04 0c 30 88 04 5a	59 c3 1e b8 00 f0 8e d8	0t* 0x*Z Y**x0xxx
00000260	2b db b7 fe 1f c3 50 56	be 6c 00 83 c6 15 e8 c6	+***xPV x!0***xx
00000270	ff be 6c 00 83 c6 17 8a	c4 e8 bb ff 5e 58 c3 50	**!0***x *****^XxP
00000280	53 56 be 8a 00 83 c6 16	8a c7 e8 aa ff 5e 5b 58	SV**0xxx *****^[X
00000290	c3 53 51 57 50 bf a3 00	83 c7 22 8b c1 e8 7f ff	xSQWPxxx **"***x
000002a0	8a c3 e8 69 ff bf a3 00	83 c7 1d 89 05 58 5f 59	***i***0 ***x*_X_Y
000002b0	5b c3 50 b4 09 cd 21 58	c3 b8 14 00 8e d8 8c db	[xPx_*!X ***0****x
000002c0	e8 97 ff ba 00 00 e8 e9	ff ba 0f 00 80 ff ff 74	*****0xxx ***0****t
000002d0	47 ba 14 00 80 ff fe 74	3f ba 14 00 80 ff fb 74	Gx*0****t ?*0****t
000002e0	37 ba 1c 00 80 ff fc 74	2f ba 21 00 80 ff fa 74	7x*0****t /x!0****t
000002f0	27 ba 30 00 80 ff fc 74	1f ba 45 00 80 ff f8 74	'x00****t *xE0****t
00000300	17 ba 54 00 80 ff fd 74	0f ba 5b 00 80 ff f9 74	*xT0****t *x[0****t
00000310	07 8a c7 e8 f8 fe 8b d0	e8 97 ff b4 30 cd 21 e8	*xxxxxxx *****0x!x
00000320	44 ff e8 5a ff e8 69 ff	ba 6c 00 e8 84 ff ba 8a	DxxZxx!x x!0*****
00000330	00 e8 7e ff ba a3 00 e8	78 ff 32 c0 b4 4c cd 21	0x~xxx0x x*2xxLx!
00000340	54 59 50 45 20 49 4d 42	20 50 43 20 0d 0a 24 50	TYPE IMB PC __\$P
00000350	43 0d 0a 24 50 43 2f 58	54 0d 0a 24 41 54 0d 0a	C__\$PC/X T__\$SAT__
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 56 65 72 73	nvertibl e__\$Vers
000003b0	69 6f 6e 20 6e 75 6d 62	65 72 20 4d 53 44 4f 53	ion numb er MSDOS
000003c0	3a 20 2e 20 20 20 20 6d	0a 24 53 65 72 69 61 6c	: .__\$Serial
000003d0	20 6e 75 6d 62 65 72 20	4f 45 4d 3a 20 20 20 20	number __OEM:
000003e0	0d 0a 24 53 65 72 69 61	6c 20 75 73 65 72 20 73	__\$Serial user s
000003f0	20 6e 75 6d 62 65 72 3a	20 20 20 20 20 20 20 20	number:
00000400	20 20 20 20 0d 0a 24		__\$

Рисунок 6 - «Хороший» .EXE модуль в шестнадцатеричном виде

Ответы на контрольные вопросы. Отличия форматов файлов COM и EXE программ.

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

COM файл состоит из одного сегмента и содержит данные и машинные команды. Код начинается с адреса 0h, но при загрузке модуля устанавливается смещение в 100h.

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

В «плохом» EXE файле данные и код содержатся в одном сегменте. Код располагается с адреса 300h. С адреса 0h располагается Relocation Table (таблица разметки).

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

В «хорошем» файле EXE содержится информация для загрузчика, сегмент стека, сегмент данных и сегмент кода (3 сегмента вместо одного в «плохом» .EXE). Код располагается с адреса 200h в отличии от 300h в «плохом» .EXE файле.

Шаг 5. Загрузка COM модуля в основную память.

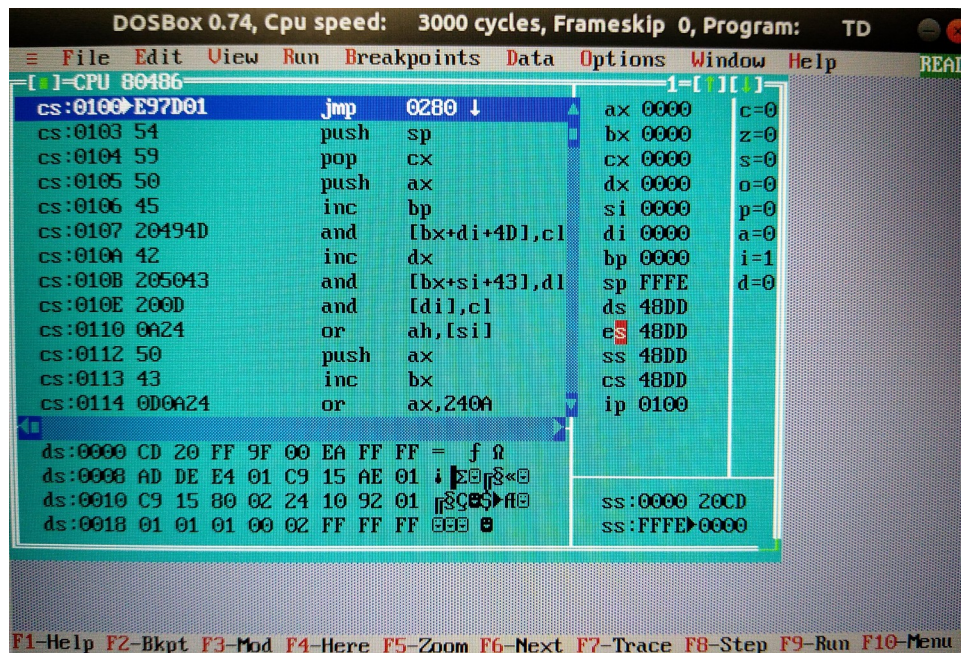


Рисунок 7 – Загрузка COM модуля в основную память

Ответы на контрольные вопросы. Загрузка COM модуля в основную память.

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

После загрузки COM-программы в память сегментные регистры указывают на начало PSP. Код располагается с адреса 100h (ip = 0100h).

2) **Что располагается с 0 адреса?**

Адрес начала PSP.

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

48DDh. Они указывают на начало PSP.

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

Какие адреса?

Стек определяется автоматически, указатель стека устанавливается на конец сегмента. Если для программы размер сегмента в 64КБ является достаточным, то DOS устанавливает в регистре SP адрес конца сегмента – FFFh. Адреса расположены в диапазоне 0000h-FFFh.

Шаг 6. Загрузка «хорошего» EXE модуля в память.

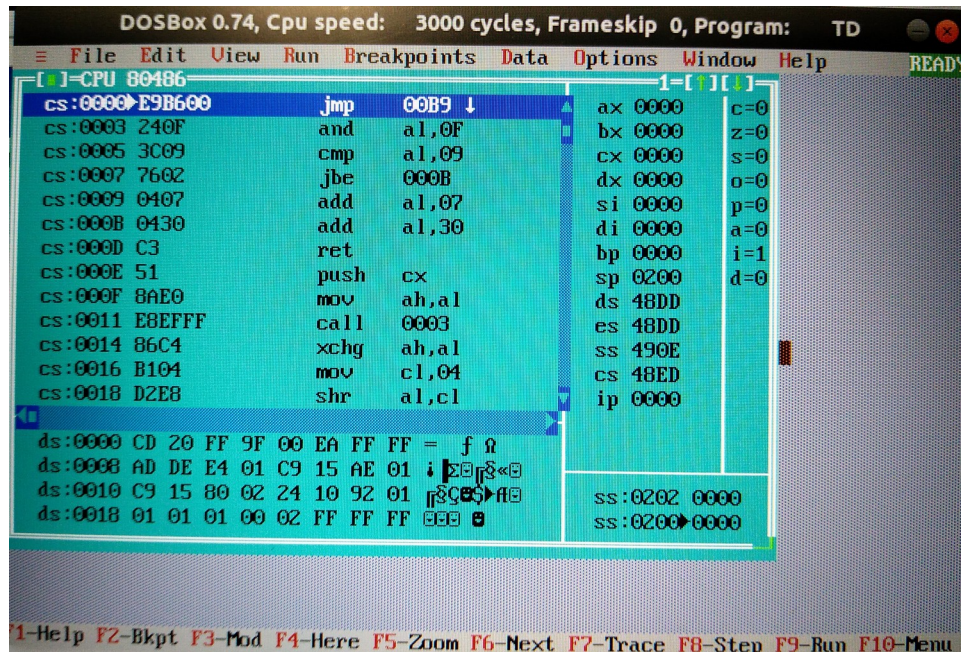


Рисунок 8 – Загрузка «хорошего» EXE модуля в память

Ответы на контрольные вопросы. Загрузка «хорошего» EXE модуля в память.

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

В области памяти строится PSP, стандартная часть заголовка считывается в память, определяется длина тела загрузочного модуля, определяется начальный сегмент, загрузочный модуль считывается в начальный сегмент, таблица настройки считывается в рабочую память, определяются значения сегментных регистров. DS и ES устанавливаются на начало PSP, SS - на начало стека, CS - на начало сегмента кода.

2) На что указывают регистры DS и ES?

DS и ES указывают на начало PSP. После выполнения команд `mov ax, @data` и `mov ds, ax` регистре DS содержит адрес начала сегмента данных.

3) Как определяется стек?

В исходном коде модуля стек определяется при помощи директивы `STACK`, а при исполнении в регистры SS и SP записываются адрес начала сегмента стека и его вершины соответственно.

4) Как определяется точка входа?

При помощи команды `END`.

Вывод.

В ходе работы было проведено исследование различий в структурах исходных текстов модулей `.COM` и `.EXE`, структур файлов загрузочных модулей и способов их загрузки в основную память.