

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. 0381

Ефимов Н.Д.

Преподаватель

Губкин А.Ф.

Санкт-Петербург

2022

Цель работы.

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

Задание.

Требуется написать 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

Основные теоретические положения.

Изначально были даны несколько процедур, выполняющих некоторые операции:

TETR_TO_HEX - Перевод десятичной цифры в код символа

BYTE_TO_HEX - Перевод байта в 16-ной с/с в символьный код

WRD_TO_HEX - Перевод слова в 16-ной с/с в символьный код

BYTE_TO_DEC - Перевод байта в 16-ной с/с в символьный код в 10-ной

с/с Выполнение работы.

WRITESTRING – Вывод строки на экран

Выполнение работы.

Для выполнения задания нужно написать две процедуры: первая – для определения типа PC, а вторая – для определения характеристик OS. Эти процедуры называются PCTYPE и OSVERSION.

1 шаг – написание файла для .COM модуля. Название файла: lb1_com.asm. Взяв за основу код из методических указаний, был написан остов программы. Дальше следовало добвать ряд переменных, которые будут хранить в себе нужную информацию для работы кода. Эти переменные приведены в разделе «Данные». Дальше были написаны нужные процедуры: процедура PCTYPE работает с информацией по адресу 0F000:0FFFEh. После взятия данных происходи их распределение согласно таблице, приложенной в руководстве к выполнению лабораторной работы. Вторая процедура, OSVERSION, работает с функцией 30h прерывания 21h, а затем, получив нужные данные, помещает их в отведенные переменные в начале кода. Дальше идет запуск программы в DOSBOX. После выполнения программы выводится следующее:

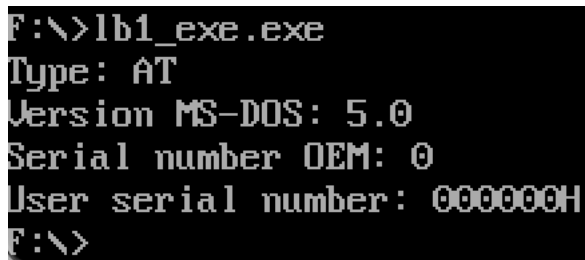
```
F:\>lb1_com.com
Type: AT
Version MS-DOS: 5.0
Serial number OEM: 0
User serial number: 000000H
F:\>
```

Если попробовать запустить код, но не переводить его из EXE, то получится следующее:

```
F:\>lb1_com.exe
Type: PC
5 0
Type: PC
0
000000
Type: PC
Type: PC
F:\>
```

Это и будет «плохой» .EXE модуль.

2 шаг – написание программы для «хорошего» модуля .EXE. Для этого нужно немного исправить файл lb1_com.asm. В этом файле не прописывается стек, поэтому и код, и данные после работы компилятора будут лежать в одной ячейке. Для этого нужно разделить сегмент кода и данных. Добавлением разделения сегмента стека, сегмента данных, сегмента кода получается файл lb1_exe.asm. После сборки в DOSBOX получится следующее:



```
F:\>lb1_exe.exe
Type: AT
Version MS-DOS: 5.0
Serial number OEM: 0
User serial number: 0000000H
F:\>
```

3 шаг – сравнение двух исходных текстов модулей .COM и .EXE. В прошлом пункте были описаны отличия, а ответы на вопросы к лабораторной работе будут даны в А.

4,5,6 шаг – ответы на вопросы будут даны в Приложении А.

Выводы.

Было проведено исследование различий в структурах исходных текстов модулей типов .COM и .EXE, структур файлов загрузочных модулей и способов их загрузки в основную память.

ПРИЛОЖЕНИЕ А

ОТВЕТЫ НА ВОПРОСЫ

Отличия исходных текстов COM и EXE программ

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

Ответ: COM-программа должна содержать только один сегмент. Код и данные находятся в одном сегменте, а стек генерируется автоматически.

2. **EXE-программа?**

Такой тип программы должен содержать не менее одного сегмента. Сегменты данных, стека и кода описываются отдельно. Если сегмент стека не описан, то будет использоваться стек DOS

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:LABONE, DS:LABONE, ES:NOTHING, SS:NOTHING`)

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

Не все форматы поддерживаются. Нельзя использовать команды вида `mov <регистр>, seg <имя сегмента>`, так как в .com-программе отсутствует таблица настроек (содержит описание адресов, которые зависят от размещения загрузочного модуля в ОП).

Отличия форматов файлов COM и EXE модулей

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

COM-файл состоит из одного сегмента, состоящего из сегмент кода и сегмент данных, сегмент стека генерируется автоматически при создании COM-программы. COM-файл ограничен размером одного сегмента и не превышает 64 Кб

Код начинается с адреса 0h, но при загрузке модуля устанавливается смещение в 100h.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
0	E9	F5	01	54	79	70	65	3A	20	50	43	0D	0A	24	54	79	éø Type: PC \$Ty
16	70	65	3A	20	50	43	2F	58	54	0D	0A	24	54	79	70	65	pe: PC\XT \$Type
32	3A	20	41	54	0D	0A	24	54	79	70	65	3A	20	50	53	32	: AT \$Type: PS2
48	20	D0	BC	D0	BE	D0	B4	D0	B5	D0	BB	D1	8C	20	33	30	Ð¼Ð»Ñ 30
64	0D	0A	24	54	79	70	65	3A	20	50	53	32	20	D0	BC	D0	\$Type: PS2 Ð¼Ð
80	BE	D0	B4	D0	B5	D0	BB	D1	8C	20	35	30	20	D0	B8	D0	¼Ð'Ð¼Ð»Ñ 50 Ð.Ð
96	BB	D0	B8	20	36	30	0D	0A	24	54	79	70	65	3A	20	50	»Ð. 60 \$Type: P
112	53	32	20	D0	BC	D0	BE	D0	B4	D0	B5	D0	BB	D1	8C	20	S2 Ð¼Ð¼Ð'Ð¼Ð»Ñ
128	38	30	0D	0A	24	54	79	70	65	3A	20	50	D0	A1	6A	72	80 \$Type: PDjr
144	0D	0A	24	54	79	70	65	3A	20	50	43	20	43	6F	6E	76	\$Type: PC Conv
160	65	72	74	69	62	6C	65	0D	0A	24	56	65	72	73	69	6F	ertible \$Versio
176	6E	20	4D	53	2D	44	4F	53	3A	20	20	2E	20	20	0D	0A	n MS-DOS: .
192	24	53	65	72	69	61	6C	20	6E	75	6D	62	65	72	20	4F	\$Serial number O
208	45	4D	3A	20	20	0D	0A	24	55	73	65	72	20	73	65	72	EM: \$User ser
224	69	61	6C	20	6E	75	6D	62	65	72	3A	20	20	20	20	20	ial number:
240	20	20	48	20	24	24	0F	3C	09	76	02	04	07	04	30	C3	H \$ \$ < v 0Ã
256	51	8A	E0	E8	EF	FF	86	C4	B1	04	D2	E8	E8	E6	FF	59	Qàèìÿ± ÒèèÿY
272	C3	53	8A	FC	E8	E9	FF	88	25	4F	88	05	4F	8A	C7	E8	ÃSüèèÿ%O OÇè
288	DE	FF	88	25	4F	88	05	5B	C3	51	52	32	E4	33	D2	B9	þÿ%O [ÃQR2ã3Ö'
304	0A	00	F7	F1	80	CA	30	88	14	4E	33	D2	3D	0A	00	73	+ñÉO N3O= s
320	F1	3C	00	74	04	0C	30	88	04	5A	59	C3	B4	09	CD	21	ñ<t 0 ZYÁ' Í!
336	C3	B8	00	F0	8E	C0	26	A0	FE	FF	3C	FF	74	1C	3C	FE	Ã, ðÃ& þÿ<ÿt <þ
352	74	1E	3C	FB	74	1A	3C	FC	74	1C	3C	FA	74	1E	3C	F8	t <Út <Út <Út <ø
368	74	26	3C	FD	74	28	3C	F9	74	2A	BA	03	01	EB	2B	90	t&<ÿt(<Út** è+
384	BA	0E	01	EB	25	90	BA	1C	01	EB	1F	90	BA	27	01	EB	° è%° è ° è
400	19	90	BA	43	01	EB	13	90	BA	69	01	EB	0D	90	BA	85	°C è ° è °
416	01	EB	07	90	BA	93	01	EB	01	90	E8	9F	FF	C3	B4	30	è ° è èÿÃ' 0
432	CD	21	50	BE	AA	01	83	C6	10	E8	6D	FF	58	8A	C4	83	ÍP¼* Æ èmÿXÃ
448	C6	03	E8	64	FF	BA	AA	01	E8	81	FF	BE	C1	01	83	C6	Æ èdy** èÿ¼Ã Æ
464	13	8A	C7	E8	53	FF	BA	C1	01	E8	70	FF	BF	D8	01	83	ÇèSÿ*Ã èpÿ¿Ø
480	C7	19	8B	C1	E8	2A	FF	8A	C3	E8	14	FF	83	EF	02	89	Ç Áè*ÿÃè ÿí
496	05	BA	D8	01	E8	55	FF	C3	E8	56	FF	E8	B0	FF	32	C0	°Ø èUÿÃèÿÿè*ÿ2Ã
512	B4	4C	CD	21													°LÍ!

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

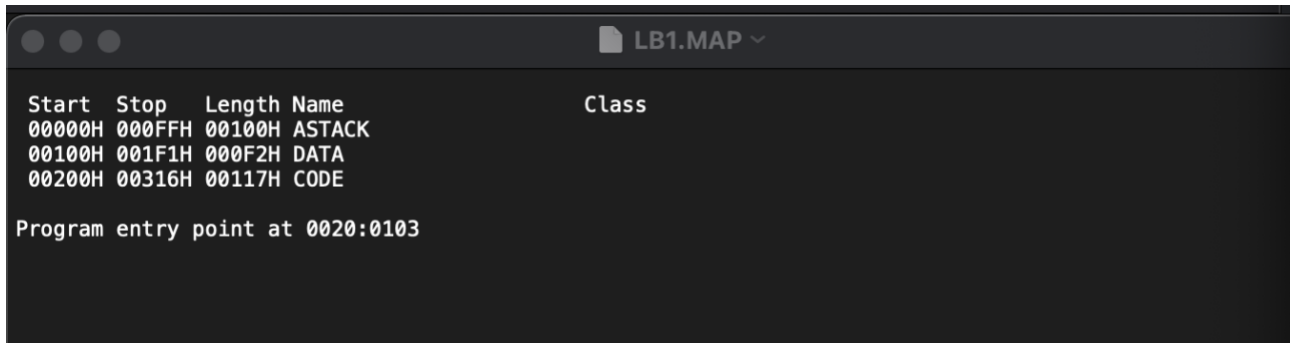
В «плохом» EXE данные и код располагаются в одном сегменте. Это неправильно, так как код и данные должны быть разделены на отдельные сегменты. Код располагается с адреса 300h, а с адреса 0h идёт таблица настроек. Символы MZ указывают на то, что это 16-битный формат исполняемого файла с расширением .EXE. Эти символы берутся из первых двух байтов – 4D и 5A. Данная сигнатура – инициалы Марка Зибовски, одного из создателей MS-DOS.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
0	4D	5A	04	01	03	00	00	00	20	00	00	00	FF	FF	00	00	MZ yy
16	00	00	66	CC	00	01	00	00	1E	00	00	00	01	00	00	00	fi
32	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
48	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
64	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
80	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
96	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
112	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
128	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
144	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
176	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
192	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
208	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
224	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
256	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
272	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
288	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
304	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
320	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
336	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
352	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
368	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
384	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
400	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
416	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
432	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
448	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
464	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
480	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
496	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
512	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
528	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
544	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
560	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
576	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
592	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
608	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
624	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
640	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
656	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
672	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
688	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
704	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
720	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
736	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
752	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
768	E9	F5	01	54	79	70	65	3A	20	50	43	0D	0A	24	54	79	é5 Type: PC \$Ty
784	70	65	3A	20	50	43	2F	58	54	0D	0A	24	54	79	70	65	re: PC/XT \$Type
800	3A	20	41	54	0D	0A	24	54	79	70	65	3A	20	50	53	32	: AT \$Type: PS2
816	20	D0	BC	D0	BE	D0	B4	D0	B5	D0	BB	D1	8C	20	33	30	Ð%Ð%Ð' ÐµÐ»Ñ 30

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

В EXE-программе код, данные и стек – отдельные сегменты. EXE-файл имеет заголовок, который используется при его загрузке. Заголовок состоит из форматированной части, содержащей сигнатуру и данные, необходимые для загрузки EXE-файла, и таблицы для настройки адресов. В отличие от «плохого» EXE в «хорошем» EXE присутствуют три сегмента: сегмент кода, сегмент данных и сегмент стека, а «плохой» EXE содержит один сегмент, совмещающий код и данные. Также в «плохом» EXE адресация кода начинается с 300h, так как он получается из .COM файла, в котором изначально сегмент кода смещён на 100h, а при создании «плохого» EXE к этому смещению добавляется размер PSP модуля(200h). А в «хорошем» EXE структура программы начинается с PSP

модуля, поэтому код начинается с 200h, а со 100h – сегмент кода.

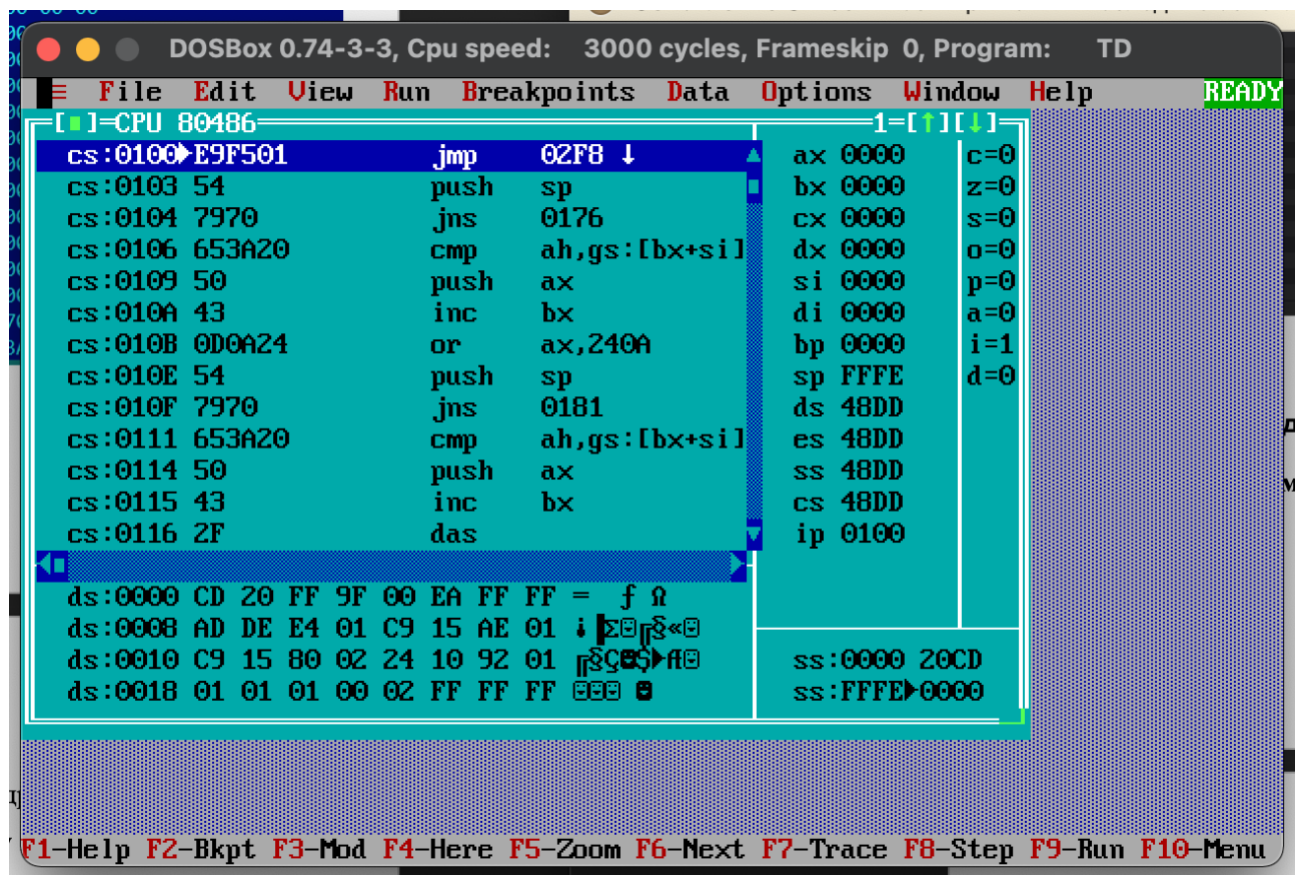


Open New from clipboard Save as... Read / filter Info																		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16		
0	4D	5A	17	01	03	00	01	00	20	00	00	00	FF	FF	00	00	MZ	yy
16	00	01	71	10	03	01	20	00	1E	00	00	00	01	00	07	01	q	
32	20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
48	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
64	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
80	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
96	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
112	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
128	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
144	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
176	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
192	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
208	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
224	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
256	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
272	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
288	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
304	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
320	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
336	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
352	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
368	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
384	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
400	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
416	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
432	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
448	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
464	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
480	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
496	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
512	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
528	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
544	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
560	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
576	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
592	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
608	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
624	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
640	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
656	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
672	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
688	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
704	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
720	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
736	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
752	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
768	54	79	70	65	3A	20	50	43	0D	0A	24	54	79	70	65	3A	Type: PC \$Type:	
784	20	50	43	2F	58	54	0D	0A	24	54	79	70	65	3A	20	41	PC/XT \$Type: A	
800	54	0D	0A	24	54	79	70	65	3A	20	50	53	32	20	D0	BC	T \$Type: PS2 D¼	
816	D0	BE	D0	B4	D0	B5	D0	BB	D1	8C	20	33	30	0D	0A	24	Ð¼Ð´Ð¼Ð»Ñ 30 \$	

Загрузка COM модуля в основную память

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

Определяется сегментный адрес участка ОП, у которого достаточно места для загрузки программы, образ COM-файла считывается с диска и помещается в память, начиная с PSP:0100h. После загрузки двоичного образа COM-программы сегментные регистры CS, DS, ES и SS указывают на PSP(в данном случае сегментные регистры указывают на 48DD), SP указывает на конец сегмента PSP(обычно FFFE), слово 00H помещено в стек, IP содержит 100H в результате команды JMP PSP:100H.



2. **Что располагается с адреса 0?**

Программный сегмент PSP, размером 256 байт (100h), зарезервированный операционной системой.

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

Сегментные регистры CS, DS, ES и SS указывают на PSP и имеют значения 48DD.

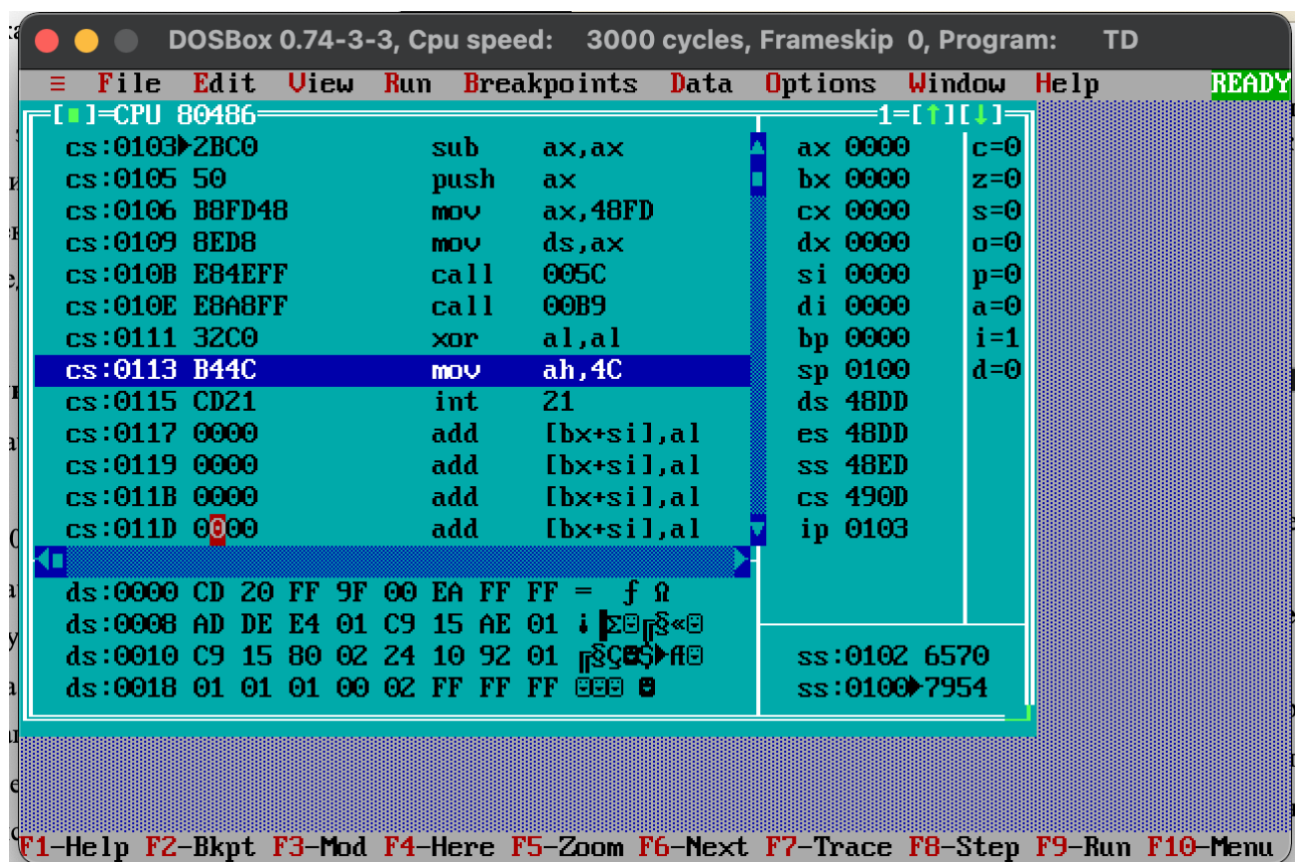
4. **Как определяется стек? Какую область памяти он занимает? Какие адреса?**

Стек генерируется автоматически при создании COM-программы. SS – на начало (0h), регистр SP указывает на конец стека (FFFEh), Адреса стека расположены в диапазоне 0h – FFFЕh (FFFEh, – последний адрес, кратный двум).

Загрузка «хорошего» EXE модуля в основную память

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

EXE-файл загружается с адреса PSP:0100h. В процессе загрузки считывается информация заголовка (PSP) в начале файла и выполняется перемещение адресов сегментов, то есть DS и ES устанавливаются на начало сегмента PSP (DS=ES=48DD), SS (SS=48ED) – на начало сегмента стека, CS (CS=490D) – на начало сегмента команд. В IP загружается смещение точки входа в программу, которая берётся из метки после директивы END. Причём дополнительный программный сегмент (PSP) присутствует в каждом EXE-файле.



2. На что указывают регистры DS и ES?

Они указывают на начало сегмента PSP

3. **Как определяется стек?**

Стек определяется директивой `stack`, после которой задается размер стека. При исполнении регистр `SS` указывает на начало этого стека, а `SP` – на конец стека

4. **Как определяется точка входа?**

При помощи директивы `END`.