

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. 9382

Русинов Д.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.

Задание.

Шаг 1. Напишите текст исходного .COM модуля, который определяет тип РС и версию системы. Это довольно простая задача и для тех, кто уже имеет опыт программирования на ассемблере, это будет небольшой разминкой. Для тех, кто раньше не сталкивался с программированием на ассемблере, это неплохая задача для первого опыта.

За основу возьмите шаблон, приведенный в разделе «Основные сведения». Необходимые сведения о том, как извлечь требуемую информацию, представлены в следующем разделе.

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 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 и серийным номером пользователя. Полученные строки выводятся на экран.

Отладьте полученный исходный модуль.

Результатом выполнения этого шага будет «хороший» .COM модуль, а также необходимо построить «плохой» .EXE, полученный из исходного текста для .COM модуля.

Шаг 2. Напишите текст исходного .EXE модуля, который выполняет те же функции, что и модуль в Шаге 1 и постройте и отладьте его. Таким образом, будет получен «хороший» .EXE.

Шаг 3. Сравните исходные тексты для .COM и .EXE модулей. Ответьте на контрольные вопросы «Отличия исходных текстов COM и EXE программ».

Шаг 4. Запустите FAR и откройте (F3/F4) файл загрузочного модуля .COM и файл «плохого» .EXE в шестнадцатеричном виде. Затем откройте (F3/F4) файл загрузочного модуля «хорошего» .EXE и сравните его с предыдущими файлами. Ответьте на контрольные вопросы «Отличия форматов файлов COM и EXE модулей».

Шаг 5. Откройте отладчик TD.EXE и загрузите .COM. Ответьте на контрольные вопросы «Загрузка COM модуля в основную память». Представьте в отчете план загрузки модуля .COM в основную память.

Шаг 6. Откройте отладчик TD.EXE и загрузите «хороший» .EXE. Ответьте на контрольные вопросы «Загрузка «хорошего» EXE модуля в основную память».

Шаг 7. Оформление отчета в соответствии с требованиями. В отчете необходимо привести скриншоты. Для файлов их вид в шестнадцатеричном виде, для загрузочных модулей – в отладчике.

Необходимые сведения для составления программы

Тип IBM PC хранится в байте по адресу 0F000:0FFFEh, 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

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.

Контрольные вопросы по лабораторной работе No1

Отличия исходных текстов COM и EXE программ

- 1) Сколько сегментов должна содержать COM-программа?
- 2) EXE-программа?
- 3) Какие директивы должны обязательно быть в тексте COM-программы?
- 4) Все ли форматы команд можно использовать в COM-программе?

Отличия форматов файлов COM и EXE модулей

- 1) Какова структура файла COM? С какого адреса располагается код?
- 2) Какова структура файла «плохого» EXE? С какого адреса располагается код? Что располагается с адреса 0?
- 3) Какова структура файла «хорошего» EXE? Чем он отличается от файла «плохого» EXE?

Загрузка COM модуля в основную память

- 1) Какой формат загрузки модуля COM? С какого адреса располагается код?
- 2) Что располагается с адреса 0?
- 3) Какие значения имеют сегментные регистры? На какие области памяти они указывают?

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

Загрузка «хорошего» EXE модуля в основную память

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

2) На что указывают регистры DS и ES?


3) Как определяется стек?

4) Как определяется точка входа?

Выполнение работы.

Были объявлены константные строки для вывода информации. Была определена процедура для определения типа PC – DEFINE_PC в соответствии с таблицей в задании. А также функция для определения ОС и прочих данных – DEFINE_OS.

В результате выполнения были получены следующие результаты:

Модуль	Результат
Хороший .EXE	Type: AT Version MS-DOS: 5.0 Serial number OEM: 0 User serial number: 000000H
Плохой .EXE	
Хороший .COM	Type: AT Version MS-DOS: 5.0 Serial number OEM: 0 User serial number: 000000H

Выводы.

Были исследованы модули .COM и .EXE, рассмотрены из различия в исходных текстах и различия готовых модулей. Также были рассмотрены способы загрузки модулей в основную память.

ПРИЛОЖЕНИЕ А

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

Отличия исходных текстов COM и EXE программ:

1. Сколько сегментов должна содержать COM-программа?

Один сегмент. Стек генерируется автоматически, а код с данными располагаются в одном сегменте.

2. EXE-программа?

Не менее одного сегмента, при этом сегменты стека, кода и данных отдельны друг от друга.

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

В программе должна быть обязательно директива `org 100h`. Она позволяет сместить всю адресацию на 256 байт. Это необходимо, поскольку первые 256 байт занимает блок PSP, а все сегментные регистры при загрузке указывают именно на него. Также необходимо привязать сегмент данных и сегмент кода на один общий сегмент с помощью `ASSUME`.

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

Так как адрес сегмента до загрузки неизвестен, нельзя использовать, например: `MOV REG, SEG`

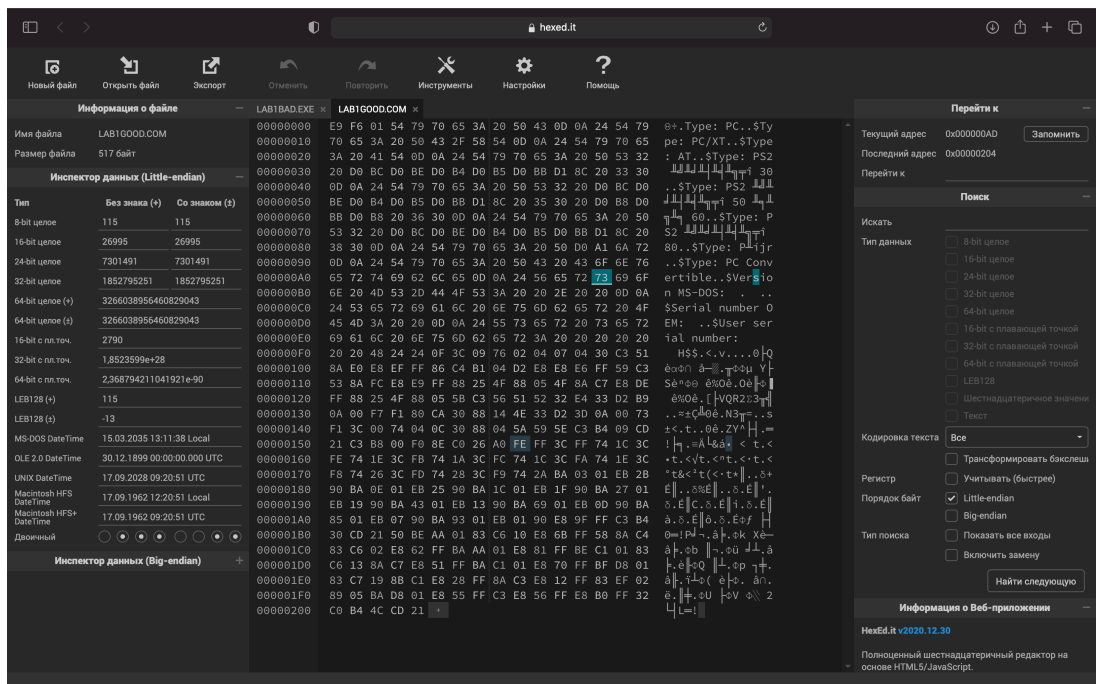
COM-программа не содержит таблицы настройки, которая содержит описание адресов. Адреса в свою очередь зависят от размещения загрузочного модуля в оперативной памяти. Поэтому использование команд, дающих доступ к началу сегментов – недопустимо.

Отличия форматов файлов .COM и .EXE программ:

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

Структура состоит из одного сегмента, в который входит сегмент кода и сегмент данных, стек генерируется автоматически. Также COM-файл имеет ограничение в размере, его максимальный размер – 64 КБ – это максимальный размер одного сегмента.

Сегмент с кодом и данными начинается с 0h. При загрузке модуля устанавливается смещение 256 байт.



DOSBox 0.74-3-2, Cpu speed: 3000 cycles, Frameskip 0, Program: AFDPRO

AX 0000	SI 0000	CS 19F5	IP 0100	Stack +0 0000	Flags 7202
BX 0000	DI 0000	DS 19F5		+2 20CD	
CX 0205	BP 0000	ES 19F5	HS 19F5	+4 9FFF	OF DF IF SF ZF AF PF CF
DX 0000	SP FFFE	SS 19F5	FS 19F5	+6 EA00	0 0 1 0 0 0 0 0

CMD > CD

00FC 0000	ADD	[BX+SI],AL	DS:0000	CD 20 FF 9F 00 EA F0 FE
00FE 0000	ADD	[BX+SI],AL	DS:0008	AD DE 1B 05 C5 06 00 00
0100 E9F601	JMP	02F9	DS:0010	18 01 10 01 18 01 92 01
0103 54	PUSH	SP	DS:0018	01 01 01 00 02 FF FF FF
0104 7970	JNS	0176	DS:0020	FF FF FF FF FF FF FF FF
0106 65	DB	65	DS:0028	FF FF FF FF EB 19 C0 11
0107 3A20	CMP	AH,[BX+SI]	DS:0030	A2 01 14 00 18 00 F5 19
0109 50	PUSH	AX	DS:0038	FF FF FF FF 00 00 00 00
010A 43	INC	BX	DS:0040	05 00 00 00 00 00 00 00
			DS:0048	00 00 00 00 00 00 00 00

2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
DS:0100	E9	F6	01	54	79	70	65	3A	20	50	43	0D	0A	24	54	79	
DS:0110	70	65	3A	20	50	43	2F	58	54	0D	0A	24	54	79	70	65	
DS:0120	3A	20	41	54	0D	0A	24	54	79	70	65	3A	20	50	53	32	
DS:0130	20	D0	BC	D0	BE	D0	B4	D0	B5	D0	BB	D1	8C	20	33	30	
DS:0140	0D	0A	24	54	79	70	65	3A	20	50	53	32	20	D0	BC	D0	

0÷.Type: PC..\$Type

pe: PC/X T..\$Type

: AT..\$T ype: PS2

..\$Type: PS2

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 ? ↑ 8 ↓ 9 ←= 10 ⇒

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

Код и данные расположены в одном сегменте. Начинается сегмент с кодом и данными с адреса 300h. С 0h идет заголовок и таблица настроек.

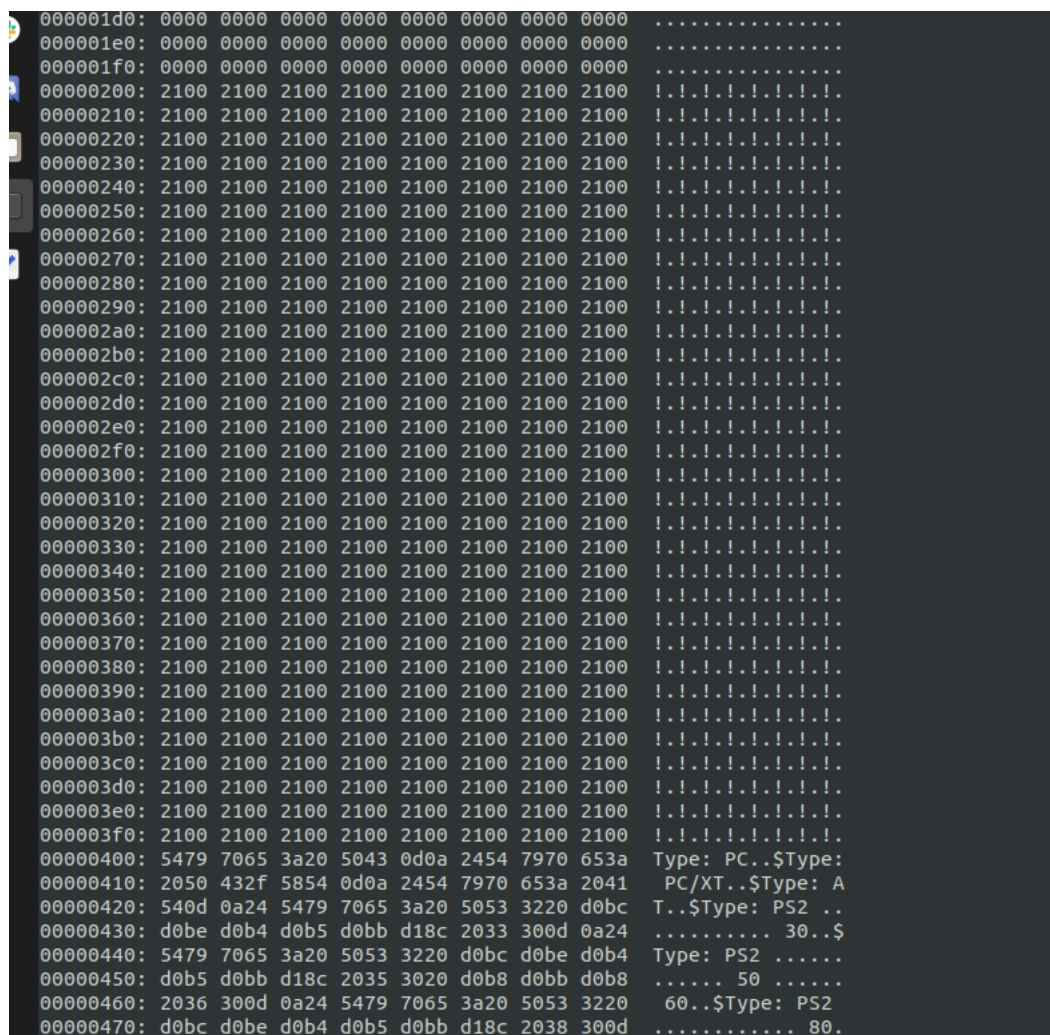
The screenshot shows the HexEd.it v2020.12.30 interface. The main window displays the file 'LAB1BAD.EXE' with a size of 1285 bytes (2 KiB). The 'Инспектор данных (Little-endian)' (Data Inspector) is active, showing the file's metadata and the hex editor view. The hex editor shows the first 256 bytes of the file, starting with the MZ header. The right sidebar contains navigation and search options.

The screenshot shows the HexEd.it v2020.12.30 interface. The main window displays the file 'LAB1BAD.EXE' with a size of 1285 bytes (2 KiB). The 'Инспектор данных (Little-endian)' (Data Inspector) is active, showing the file's metadata and the hex editor view. The hex editor shows the first 256 bytes of the file, starting with the MZ header. The right sidebar contains navigation and search options.

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

EXE-файл имеет поделенные на сегменты стек, данные и код, может иметь неограниченный размер, имеет в начале заголовок, который используется при загрузке этого модуля. Заголовок содержит данные и сигнатуру, таблицу для настройки адресов.

Плохой EXE-файл не имеет разделения сегментов кода и данных. Плохой EXE-файл имеет смещение 300h, поскольку изначально смещение 100h, а при создании EXE появляется смещение 200h для модуля PSP. У хорошего EXE выделяется сегмент под стек, поэтому итоговое смещение у хорошего EXE в данной программе – 400h.



000001d0:	0000	0000	0000	0000	0000	0000	0000	0000
000001e0:	0000	0000	0000	0000	0000	0000	0000	0000
000001f0:	0000	0000	0000	0000	0000	0000	0000	0000
00000200:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
00000210:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
00000220:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
00000230:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
00000240:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
00000250:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
00000260:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
00000270:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
00000280:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
00000290:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
000002a0:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
000002b0:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
000002c0:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
000002d0:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
000002e0:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
000002f0:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
00000300:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
00000310:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
00000320:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
00000330:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
00000340:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
00000350:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
00000360:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
00000370:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
00000380:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
00000390:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
000003a0:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
000003b0:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
000003c0:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
000003d0:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
000003e0:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
000003f0:	2100	2100	2100	2100	2100	2100	2100	2100	! ! ! ! ! ! ! ! ! !
00000400:	5479	7065	3a20	5043	0d0a	2454	7970	653a	Type: PC...\$Type:
00000410:	2050	432f	5854	0d0a	2454	7970	653a	2041	PC/XT...\$Type: A
00000420:	540d	0a24	5479	7065	3a20	5053	3220	d0bc	T...\$Type: PS2 ..
00000430:	d0be	d0b4	d0b5	d0bb	d18c	2033	300d	0a24 30...\$
00000440:	5479	7065	3a20	5053	3220	d0bc	d0be	d0b4	Type: PS2
00000450:	d0b5	d0bb	d18c	2035	3020	d0b8	d0bb	d0b8 50
00000460:	2036	300d	0a24	5479	7065	3a20	5053	3220	60...\$Type: PS2
00000470:	d0bc	d0be	d0b4	d0b5	d0bb	d18c	2038	300d 80.

Загрузка COM модуля в основную память:

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

Сначала определяется адрес сегмента оперативной памяти, в котором достаточно места для загрузки программы, затем считывается COM-файл с диска и помещается в память, начиная с PSP:0100h. Сегментные регистры CS, DS, ES и SS будут указывать на PSP (48DD). SP будет указывать на конец PSP, 00H помещен в стек, IP содержит 100H из-за команды JMP PSP:100h.

2. Что располагается с адреса 0?

Сегмент PSP, размер сегмента 100h.

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

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

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

Стек генерируется автоматически при загрузке COM-программы. SS – на начало – 0h, SP – на конец – FFFEh, адрес расположен в диапазоне 0h – FFFEh.

Загрузка «хорошего» EXE модуля в основную память:

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

Начинается загрузка с адреса PSP:0100h. Считывается информация PSP, выполняется перемещение адресов сегментов, DS и ES устанавливаются на начало PSP, SS устанавливается на начало сегмента стека, CS на начало сегмента кода. В IP загружается смещение до точки входа в программу, оно берется из метки после директивы END.

CS = 490D

SS = 48ED

ES = 48DD

DS = 48DD

2. На что указывают регистры DS и ES?

Регистры DS и ES указывают на начало сегмента PSP.

3. Как определяется стек?

При помощи директивы .STACK, также задается размер стека. Регистр SS будет указывать на начало сегмента стека, а SP на конец.

4. Как определяется точка входа?

Точка входа определяется при помощи директивы END.