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. 0382 _____ Азаров М.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Исследование различий в структурах исходных текстовых 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 PC 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 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 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.

Ход работы

1. Создаем исходник для будущего COM файла.
   1.1. Берем за основу исходник для COM из методички.
   1.2. Сохраняем в программе все сообщения, которые она может вывести.

```
;types PC
PC db 'Type IBM PC: PC',0DH,0AH,'$' ;FF
PCXT db 'Type IBM PC: PC/XT',0DH,0AH,'$' ;FE, FB
AT db 'Type IBM PC: AT',0DH,0AH,'$' ;FC
PS2_30 db 'Type IBM PC: PS model 30',0DH,0AH,'$' ;FA
PS2_80 db 'Type IBM PC: PC model 80',0DH,0AH,'$' ;F8
PCJR db 'Type IBM PC: PCjr',0DH,0AH,'$' ;FD
PCC db 'Type IBM PC: PC Convertible',0DH,0AH,'$' ;F9

VERSION db 'MS DOS version: 01.00 ',0DH,0AH,'$'
OEM_MES db 'OEM:     ',0DH,0AH,'$'
USER db 'User:       H',0DH,0AH,'$'
```

1.3. Создаем макрос **WRITE_MES** для вывода сообщений.

```
WRITE_MES MACRO mes
    mov DX, offset mes
    mov AH, 09h
    int 21h
ENDM
```

1.4. Создаем макрос для удобного определения типа системы.

```
CHECK_TYPE_PC MACRO val, pctype
    cmp AL, val
    jne @f
    WRITE_MES pctype
    jmp DOS_VESION
@@:
ENDM
```

val – значение байта которому соответствует определённая система.

1.5. Выполняем первое задание по определению типа системы.

```
BEGIN:
    mov BX, 0F000h
    mov ES, BX
    mov AL, ES:[0FFFEh]

    CHECK_TYPE_PC 0FFh, PC
    CHECK_TYPE_PC 0FEh, PCXT
    CHECK_TYPE_PC 0FBh, PCXT
    CHECK_TYPE_PC 0FDh, PCJR
    CHECK_TYPE_PC 0FCh, AT
    CHECK_TYPE_PC 0FAh, PS2_30
    CHECK_TYPE_PC 0F8h, PS2_80
    CHECK_TYPE_PC 0F9h, PCC

UNKNOWN_TYPE_PC:
    call BYTE_TO_HEX
    mov BH, AH
    mov DL, AL
    mov AH, 06h
    int 21h
    mov DL, BH
    int 21h
```

Раздел **UNKNOWN_TYPE_PC** нужен для случаев, когда определяемого типа системы нет в известном списке.

1.6. Определяем версию системы:

```
DOS_VESION:
    mov AH, 30h
    int 21h
    mov SI, offset VERSION
    add SI, 17
    cmp AL, 00h
    je MODIFICATION
    mov DH, AH
    call BYTE_TO_DEC ; AL -> VERSION[17]  (= SI)
    mov AL, DH

MODIFICATION:
    add SI, 3
    call BYTE_TO_DEC ; AL -> VERSION[20]   (= SI)
    WRITE MES VERSION
```

1.7. Определяем серийный номер OEM:

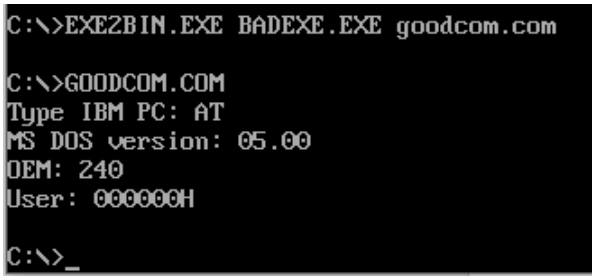
```
OEM:
    mov AL, BH
    mov SI, offset OEM_MES
    add SI, 7
    call BYTE_TO_DEC
    WRITE_MES OEM_MES
```

1.8. Определяем номер пользователя:

```
USER_NUM:
    mov SI, offset USER
    add SI, 11
    mov AX, CX
    call WRD_TO_HEX ; AX -> USER[11]  (= SI)
    mov AL, BL
    call BYTE_TO_HEX ;AL -> junior rank = AH ,senior rank = AL
    sub SI, 2
    mov [SI], AX
    WRITE_MES USER
```

2. Создаем «плохой» EXE. Результат работы плохого EXE:

3. Создаем COM файл из «плохого» EXE с помощью программы EXE2BIN.EXE:



Программа работает корректно.

4. Создаем исходник для будущего EXE файла.
5. Создаем загрузочный модуль хорошего EXE. Проверяем корректность:

Программа работает корректно.

Ответы на контрольные вопросы.

Отличия исходных текстов . COM и . EXE программ:

1) Сколько сегментов должна содержать .COM программа?

Ответ: Программа для COM файла должна содержать только один сегмент – он используется, как сегмент кода, так и сегмент данных. Сегмент стека создается автоматически в этом же сегменте.

2) .EXE программа?

Ответ: От одного в который можно заключить и код и данные и стек, до четырех – сегмент кода , сегмент данных , стек и дополнительный сегмент.

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

Ответ: Необходимо директива ORG 100h, для смещения сегмента, чтобы не попасть в область PSP. Также необходимо использовать ASSUME (CS:TESTPC, DS:TESTPC, etc), чтобы связать сегмент с сегментными регистрами.

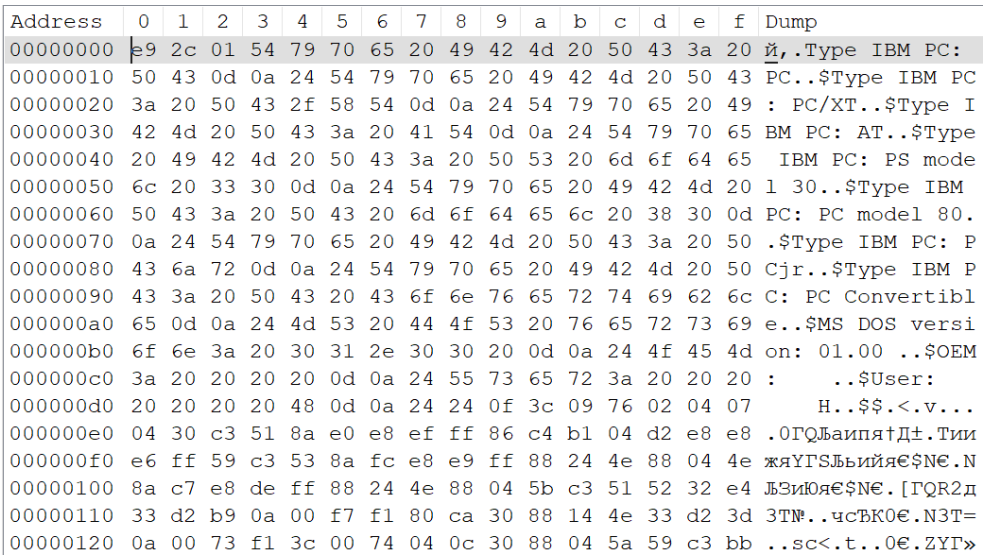
4) Все ли форматы команд можно использовать в .COM программе?

Ответ: Нет, команды с указанием сегментов не могут быть выполнены. В момент ассемблирования и редактирования связей сегментное значение для сегмента неизвестно. Оно определяется только при загрузке программы. Поскольку файл типа .COM не может предоставить загрузчику перечня всех сегментных ссылок (информация для перемещения), то в данном случае программа будет выполняться неправильно.

Отличия форматов файлов . COM и . EXE модулей

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

Ответ: COM файл состоит из одного сегмента в котором находятся и код и данные. Также в этом сегменте автоматически создается стек. Код располагается с адреса 0h , но из-за директивы ORG 100h будет загружен в ОП со смещением начиная с адреса 100h.



Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
00000000	e9	2c	01	54	79	70	65	20	49	42	4d	20	50	43	3a	20	й,.Type IBM PC:
00000010	50	43	0d	0a	24	54	79	70	65	20	49	42	4d	20	50	43	PC..$Type IBM PC
00000020	3a	20	50	43	2f	58	54	0d	0a	24	54	79	70	65	20	49	: PC/XT..$Type I
00000030	42	4d	20	50	43	3a	20	41	54	0d	0a	24	54	79	70	65	BM PC: AT..$Type
00000040	20	49	42	4d	20	50	43	3a	20	50	53	20	6d	6f	64	65	IBM PC: PS mode
00000050	6c	20	33	30	0d	0a	24	54	79	70	65	20	49	42	4d	20	l 30..$Type IBM
00000060	50	43	3a	20	50	43	20	6d	6f	64	65	6c	20	38	30	0d	PC: PC model 80.
00000070	0a	24	54	79	70	65	20	49	42	4d	20	50	43	3a	20	50	.$Type IBM PC: P
00000080	43	6a	72	0d	0a	24	54	79	70	65	20	49	42	4d	20	50	Cjr..$Type IBM P
00000090	43	3a	20	50	43	20	43	6f	6e	76	65	72	74	69	62	6c	C: PC Convertibl
000000a0	65	0d	0a	24	4d	53	20	44	4f	53	20	76	65	72	73	69	e..$MS DOS versi
000000b0	6f	6e	3a	20	30	31	2e	30	30	20	0d	0a	24	4f	45	4d	on: 01.00 ..$OEM
000000c0	3a	20	20	20	20	0d	0a	24	55	73	65	72	3a	20	20	20	: ..$User:
000000d0	20	20	20	20	48	0d	0a	24	24	0f	3c	09	76	02	04	07	H..$$.<.v...
000000e0	04	30	c3	51	8a	e0	e8	ef	ff	86	c4	b1	04	d2	e8	e8	.0ГQЉаипя†Д±.Тии
000000f0	e6	ff	59	c3	53	8a	fc	e8	e9	ff	88	24	4e	88	04	4e	жяYГSЉьийя€$N€.N
00000100	8a	c7	e8	de	ff	88	24	4e	88	04	5b	c3	51	52	32	e4	ЉЗиЮя€$N€.[ГQR2д
00000110	33	d2	b9	0a	00	f7	f1	80	ca	30	88	14	4e	33	d2	3d	3Т№..чсЂК0€.N3Т=
00000120	0a	00	73	f1	3c	00	74	04	0c	30	88	04	5a	59	c3	bb	..sc<.t..0€.ZYГ»

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

Ответ: Плохой EXE содержит только один сегмент. Причем сначала располагается заголовок, потом смещение в 100h из-за директивы org 100h и потом основной сегмент начиная с 300h. Как раз-таки с адреса 0h располагается заголовок EXE файла.

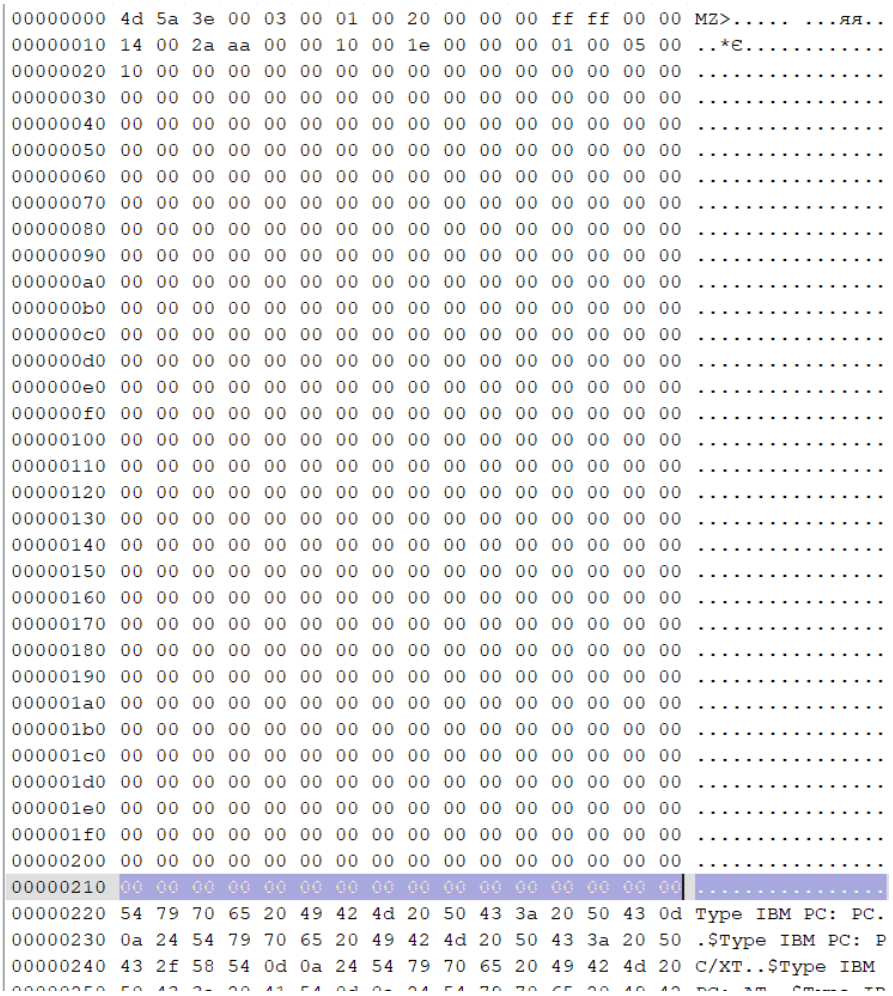
```
00000000 4d 5a 0d 01 03 00 00 00 20 00 00 00 ff ff 00 00 MZ..... ...яя..
00000010 00 00 11 a4 00 01 00 00 1e 00 00 00 01 00 00 00 ...¤............
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

000002c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000002d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000002e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000002f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000300 e9 2c 01 54 79 70 65 20 49 42 4d 20 50 43 3a 20 й,.Type IBM PC:
00000310 50 43 0d 0a 24 54 79 70 65 20 49 42 4d 20 50 43 PC..$Type IBM PC
00000320 3a 20 50 43 2f 58 54 0d 0a 24 54 79 70 65 20 49 : PC/XT..$Type I
00000330 42 4d 20 50 43 3a 20 41 54 0d 0a 24 54 79 70 65 BM PC: AT..$Type
00000340 20 49 42 4d 20 50 43 3a 20 50 53 20 6d 6f 64 65  IBM PC: PS mode
```

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

Ответ: В хорошем EXE сначала располагается заголовок. Заголовок содержит необходимую информацию для загрузки программы в память и специальную таблицу, необходимую для настройки ссылок на дальние сегменты программы. Затем идут сегменты в очерёдности их объявления в исходнике.

Оба начинаются с заголовка, но в «плохом» после заголовка идет просто смещение на 100h и затем единственный сегмент, а в «хорошем» после заголовка идет сегмент стека (с 210h), затем сегмент данных и потом сегмент кода.



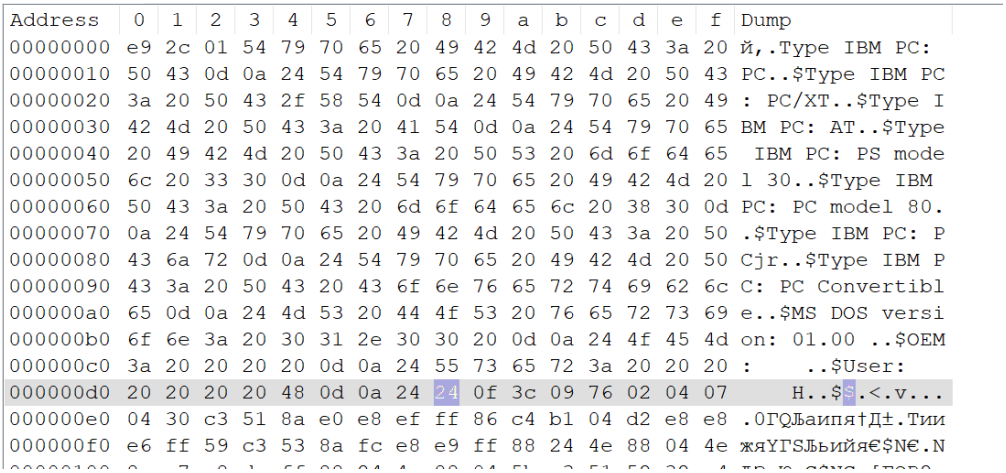
4) Как определяется стек? Какую область памяти он занимает? Какие адреса?

Ответ: Стек находится между заголовком и сегментом данных и занимает 210h-21Fh

Загрузка .COM модуля в основную память

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

Ответ: COM-файла считывается с диска и помещается в память, начиная с PSP:0100h. После загрузки сегментные регистры CS, DS, ES и SS указывают на PSP, SP указывает на конец сегмента PSP, IP содержит 100H в результате команды JMP PSP:100H.



```
Address  0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  Dump
00000000 e9 2c 01 54 79 70 65 20 49 42 4d 20 50 43 3a 20 й,.Type IBM PC:
00000010 50 43 0d 0a 24 54 79 70 65 20 49 42 4d 20 50 43 PC..$Type IBM PC
00000020 3a 20 50 43 2f 58 54 0d 0a 24 54 79 70 65 20 49 : PC/XT..$Type I
00000030 42 4d 20 50 43 3a 20 41 54 0d 0a 24 54 79 70 65 BM PC: AT..$Type
00000040 20 49 42 4d 20 50 43 3a 20 50 53 20 6d 6f 64 65  IBM PC: PS mode
00000050 6c 20 33 30 0d 0a 24 54 79 70 65 20 49 42 4d 20 l 30..$Type IBM
00000060 50 43 3a 20 50 43 20 6d 6f 64 65 6c 20 38 30 0d PC: PC model 80.
00000070 0a 24 54 79 70 65 20 49 42 4d 20 50 43 3a 20 50 .$Type IBM PC: P
00000080 43 6a 72 0d 0a 24 54 79 70 65 20 49 42 4d 20 50 Cjr..$Type IBM P
00000090 43 3a 20 50 43 20 43 6f 6e 76 65 72 74 69 62 6c C: PC Convertibl
000000a0 65 0d 0a 24 4d 53 20 44 4f 53 20 76 65 72 73 69 e..$MS DOS versi
000000b0 6f 6e 3a 20 30 31 2e 30 30 20 0d 0a 24 4f 45 4d on: 01.00 ..$OEM
000000c0 3a 20 20 20 20 0d 0a 24 55 73 65 72 3a 20 20 20 :    ..$User:
000000d0 20 20 20 20 48 0d 0a 24 24 0f 3c 09 76 02 04 07     H..$$.<.v...
000000e0 04 30 c3 51 8a e0 e8 ef ff 86 c4 b1 04 d2 e8 e8 .0ГQЉаипя†Д±.Тии
000000f0 e6 ff 59 c3 53 8a fc e8 e9 ff 88 24 4e 88 04 4e жяYГSЉьийя€$N€.N
```

2) Что располагается с адреса 0?

Ответ: Если имеется ввиду ОП то PSP, если же загрузочный модуль то сегмент.

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

Ответ: Сегментные регистры CS, DS, ES и SS указывают на сегмент кода, сегмент данных, дополнительные данные и стек соответственно. В начале выполнения программы регистры указывают на начало PSP

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

Ответ: Стек генерируется автоматически при создании .COM программы в конце сегмента. Адреса стека расположены в диапазоне FFFEh – 0h.

Загрузка «хорошего» EXE модуля в основную память

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

Ответ: Загрузочный модуль считывается в начальный сегмент. Таблица настройки порциями считывается в рабочую память. Для каждого

элемента таблицы настройки к полю сегмента прибавляется сегментный адрес начального сегмента. В результате элемент таблицы указывает на слово в памяти, к которому прибавляется сегментный адрес начального сегмента. Когда таблица настройки адресов обработана, в регистры SS и SP записываются значения, указанные в заголовке, а к SS прибавляется сегментный адрес начального сегмента. В ES и DS записывается сегментный адрес начала PSP. Управление передается по адресу, указанному в заголовке

2)На что указывают регистры DS и ES?

Ответ: В начале выполнения программы они указывают на PSP.

3) Как определяется стек?

Ответ: Стек определяется с помощью директивы .stack, или вручную с помощью директивы segment.

3) Как определяется точка входа?

Ответ: Директивой END.

Вывод.

В ходе работы были изучены различия в структурах исходных текстовых модулей типов **.COM** и **.EXE**, а также различия в загрузочных модулей этих типов и способов их загрузки в основную память.