

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. 9382

Павлов Р.В.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Постановка задачи.

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 и структуры данных:

Название процедуры	Описание процедуры
main	Вызов реализованных процедур, вывод на экран типа ПК
pctype	Формирует строку с указанием типа ПК, помещает в регистр DX её смещение
type_sys_info	Выводит на экран информацию о текущей версии ОС
byte_to_dec	Записывает в строку число размером 1 байт в десятичной системе по указанному в DX адресу
byte_to_hex	Записывает в строку число размером 1 байт в шестнадцатеричной системе по указанному в DX адресу

Последовательность действий:

1. Вызов в main процедуры pctype, получение готовой строки с указанием типа ПК
2. Вывод на экран полученной строки
3. Вызов функции, формирующей строки с информацией об ОС и выводящей их
 - a) Указание целевых адресов (смещение + индекс начала зарезервированных символов строк)
 - b) Побайтовая загрузка информации в AL
 - c) Вызов одной из процедур для перевода числа в из AL в нужную СС и помещения в строки
 - d) Вывод сформированных строк на экран

Ход работы.

- 1) Написан текст исходного .COM модуля, реализованы процедуры формирования строк с типом ПК и версией ОС. Модуль отлажен, созданы «хороший» .COM и «плохой» .EXE загрузочный модули, последний получен из исходного .COM.
- 2) Написан текст исходного .EXE модуля, в котором реализованы идентичные процедуры, но также введено несколько сегментов (помимо CODE есть ещё STACK и DATA). Модуль отлажен, создан загрузочный .EXE модуль.
- 3) Выполнено сравнение исходных текстов .COM и .EXE модулей, выявлены различия.
- 4) Файлы загрузочных модулей .COM и .EXE («хорошего» и «плохого») просмотрены в Far manager в шестнадцатеричном виде, найдены отличия .COM модулей от .EXE.

◦ .COM модуль:

```
3:\Документы\Операционные системы\pavlov\lab1\LAB1.COM.COM
3\▲Pш\  =I|0=Iшй  PSQ2фн
зУГ|0|0  YeA-0SL,и [N2фтEY[X|PSQ2фн> зУГ|0|0  YeAW
|▼A-•A-0SL,и [N2фтшY[X|▲н  E0+3-н  ▼н°  :+t*Ст-ьPБы°  тУБ|т0Лзы
P||0Гт↓ша  Гь↓||Z0Гт0шо  K-Гт▼шг  Гьс|  =!K||п0Гт0шу  Гь9=!K||л0Гт0шi  K-Гт0ша  K-Гт0шY  Гь!=!PC2 модель 80
$PC Convertible
$PC2 модель 30
$PC/XT (FB)
$AT либо PC2 модель 50 или 60
$PCjr
$PC/XT (FE)
$PC
$Версия DOS: .
$Серийный номер OEM:  HEX
$Серийный номер пользователя:  HEX$Неизвестный номер модели
$ф0i0+0$0"0A0H0U0
```



```

F:\Документы\Операционные системы\pavlov\lab1\LAB1_EXE.EXE
0000000170: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000190: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000001A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000001B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000001C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000001D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000001E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000001F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000200: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000210: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000220: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000230: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000240: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000250: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000260: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000270: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000280: 50 43 32 20 AC AE A4 A5 AB EC 20 38 30 0D 0A 24 PC2 модель 80)
0000000290: 50 43 20 43 6F 6E 76 65 72 74 69 62 6C 65 0D 0A PC Convertible)
00000002A0: 24 50 43 32 20 AC AE A4 A5 AB EC 20 33 30 0D 0A $PC2 модель 30)
00000002B0: 24 50 43 2F 58 54 20 28 46 42 29 0D 0A 24 41 54 $PC/XT (FB)
00000002C0: 20 AB A8 A1 AE 20 50 43 32 20 AC AE A4 A5 AB EC либо PC2 модель
00000002D0: 20 35 30 20 A8 AB A8 20 36 30 0D 0A 24 50 43 6A 50 или 60)
00000002E0: 72 0D 0A 24 50 43 2F 58 54 20 28 46 45 29 0D 0A r)
00000002F0: 24 50 43 0D 0A 24 82 A5 E0 E1 A8 EF 20 44 4F 53 $PC)
0000000300: 3A 20 00 00 2E 00 00 0D 0A 24 91 A5 E0 A8 A9 AD : .
0000000310: EB A9 20 AD AE AC A5 E0 20 4F 45 4D 3A 20 00 00 ый номер OEM:
0000000320: 20 48 45 58 0D 0A 24 91 A5 E0 A8 A9 AD EB A9 20 HEX)
0000000330: AD AE AC A5 E0 20 AF AE AB EC A7 AE A2 A0 E2 A5 номер пользовате
0000000340: AB EF 3A 20 00 00 00 00 00 00 20 48 45 58 24 8D ля:
0000000350: A5 A8 A7 A2 A5 E1 E2 AD EB A9 20 AD AE AC A5 E0 еизвестный номер
0000000360: 20 AC AE A4 A5 AB A8 20 00 00 0D 0A 24 00 00 10 модели
0000000370: 00 21 00 31 00 3E 00 5D 00 64 00 71 00 00 00 00 ! 1 > ] d q
0000000380: 50 53 51 32 E4 BB 0A 00 33 F6 83 C6 01 B9 02 00 PSQ2)
0000000390: F6 F3 80 C4 50 53 8B DA 88 20 5B 4E 32 E4 E2 F0 YeA-05ЛrИ [N2фT
00000003A0: 59 5B 58 C3 50 53 51 32 E4 BB 10 00 33 F6 83 C6 Y[X|PSQ2)
00000003B0: 01 B9 02 00 F6 F3 80 FC 0A 7C 03 80 C4 07 80 C4 0|
00000003C0: 30 53 8B DA 88 20 5B 4E 32 E4 E2 E8 59 5B 58 C3 05ЛrИ [N2фTшY[X|
00000003D0: B8 00 F0 8E D8 33 C0 B8 FE FF BB F8 00 B9 08 00 7 00+3 7 7
00000003E0: 3A D8 74 06 43 E2 F9 EB 15 90 B8 08 00 8E D8 81 :t+Ст.ы$P 0+Б
00000003F0: EB F8 00 D1 E3 81 C3 ED 00 8B 17 EB 14 90 50 B8 ы° 7yБ|э Л$ы$PP7
0000000400: 08 00 8E D8 58 BA CF 00 83 C2 19 E8 96 FF 83 EA 0+X|= ГТ↓шЦ Гь
0000000410: 19 C3 BA 76 00 83 C2 0C E8 65 FF 8A C4 83 C2 03 ↓|v Г-9ше К-ГТ♥
0000000420: E8 5D FF 83 EA 0F B4 09 CD 21 8A C7 BA 8A 00 83 ш] Гьo|o=!K||К Г
0000000430: C2 14 E8 6F FF 83 EA 14 CD 21 8A C3 BA A7 00 83 Т9шо Гь9=!K||э Г
0000000440: C2 1D E8 5F FF 8A C5 83 C2 02 E8 57 FF 8A C1 83 Теш_ К|ГТ0шW К|Г
0000000450: C2 02 E8 4F FF 83 EA 21 CD 21 C3 33 C0 1E 50 E8 Т0ш0 Гь!=!|3!▲Pш
0000000460: 6E FF B4 09 CD 21 B4 30 CD 21 E8 A5 FF CB n |o=!|0=!ше 7

```

5) Посредством отладчика TD выполнена загрузка .COM и .EXE модулей в ОП.

```

E File Edit View Run Breakpoints Data Options Window Help READY
[ ] CPU 80486 1=[ ]
cs:0100 33C0 xor ax,ax ax 0000 c=0
cs:0102 1E push ds bx 0000 z=0
cs:0103 50 push ax cx 0000 s=0
cs:0104 E85C00 call 0163 dx 0000 o=0
cs:0107 B409 mov ah,09 si 0000 p=0
cs:0109 CD21 int 21 di 0000 a=0
cs:010B B430 mov ah,30 bp 0000 i=1
cs:010D CD21 int 21 sp FFFE d=0
cs:010F E88900 call 019B ds 48DD
cs:0112 CB retf es 48DD
cs:0113 50 push ax ss 48DD
cs:0114 53 push bx cs 48DD
cs:0115 51 push cx ip 0100

ds:0000 CD 20 FF 9F 00 EA FF FF = Я Ъ
ds:0008 AD DE E4 01 C9 15 AE 01 H 00 80
ds:0010 C9 15 80 02 24 10 92 01 8A 05 T
ds:0018 01 01 01 00 02 FF FF FF 00 0

ss:0000 20CD
ss:FFFE 0000

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

```

```

E File Edit View Run Breakpoints Data Options Window Help READY
[ ] CPU 80486 1=[ ]
cs:00DB 33C0 xor ax,ax ax 0000 c=0
cs:00DD 1E push ds bx 0000 z=0
cs:00DE 50 push ax cx 0000 s=0
cs:00DF E86EFF call 0050 dx 0000 o=0
cs:00E2 B409 mov ah,09 si 0000 p=0
cs:00E4 CD21 int 21 di 0000 a=0
cs:00E6 B430 mov ah,30 bp 0000 i=1
cs:00E8 CD21 int 21 sp 0080 d=0
cs:00EA E8A5FF call 0092 ds 48DD
cs:00ED CB retf es 48DD
cs:00EE 0000 add [bx+si],al ss 48ED
cs:00F0 0000 add [bx+si],al cs 4905
cs:00F2 0000 add [bx+si],al ip 00DB

ds:0000 CD 20 FF 9F 00 EA FF FF = Я Ъ
ds:0008 AD DE E4 01 C9 15 AE 01 H 00 80
ds:0010 C9 15 80 02 24 10 92 01 8A 05 T
ds:0018 01 01 01 00 02 FF FF FF 00 0

ss:0082 2032
ss:0080 4350

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

```

Ответы на контрольные вопросы.

Отличия исходных текстов .COM и .EXE программ

1. COM-программа должна содержать один сегмент.
2. EXE-программа может содержать более 1 сегмента, но сегмент кода обязательно должен быть описан.

3. В тексте COM-программы обязательна директива **org 100h**. Она смещает IP на 256 байт вперёд (в начале единственного сегмента располагается PSP - префикс программного сегмента).
4. Запрещены команды работы с 64-битными регистрами, обращение к памяти по адресу, большему, чем 0FFFFh, и т. п. - то есть все действия программы ограничены блоком памяти размером 64 Кбайт.

Отличия форматов файлов COM и EXE модулей

1. COM-файл представляет собой набор машинных команд и данных. Код располагается с нулевого адреса (не учитывая смещения на 100h от начала).
2. «Плохой» EXE-файл был дополнен таблицей настроек, которая располагается с нулевого адреса, при этом машинный код (данные идут за ним) располагается с адреса 300h, что позволяет предположить, что 256₁₀ или 100h байт, начиная с 200h — это те 100h байт, которые были зарезервированы под ранее упомянутый PSP, который при запуске загружается в один с кодом сегмент.
3. У «хорошего» EXE сегменты расположены в порядке их расположения в исходном коде, также сегменты данных и кода, расположенные рядом, расположены чуть ближе к началу, чем в плохом EXE (280h), поскольку под стек вручную выделяется 64 слова, т. е. 128₁₀ или 80h байт, а память под PSP не резервируется.

Загрузка COM модуля в основную память

1. План загрузки:
 - 1) Выделена память объёмом 64 кб
 - 2) Данные программы помещены в выделенную область памяти
 - 3) Сегментных регистры установлены в значение адреса сегмента (48DD), который указывает на PSP
 - 4) SP установлен в FFFE (стек заполняется с конца сегмента)
 - 5) В вершине стека расположено слово 0000
 - 6) IP установлен в 100h
2. С адреса 0 располагается PSP

3. Сегментные регистры указывают на одну область памяти — единственный сегмент программы — и имеют одинаковые значения (48DD).
4. Стек занимает весь сегмент, но заполняется с его конца, ему доступны EFFF адресов (0000 – FFFE), поскольку слово занимает 2 байта.

Загрузка «хорошего» EXE модуля в основную память

1. Выделяется необходимый объём памяти, программа загружается в ОП, после чего пересчитываются ссылки, затем выполняется далёкий переход к CS:IP. Начальные значения сегментных регистров:
 - DS - 48DD
 - ES - 48DD
 - SS - 48EC
 - CS – 4905
2. DS и ES указывают на начало сегмента PSP.
3. Под стек выделяется фиксированное количество слов, на начало этого участка памяти устанавливается регистр SS, а SP указывает на его последнее слово.
4. Директивой END. Значение соответствующего метке, на которую указывает директива, адреса сегмента, заносится в CS, а адреса смещения — в IP.

Выводы.

В результате выполнения лабораторной работы были исследованы структуры .COM и .EXE загрузочных модулей, а также структуры их исходных текстов.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД

- имя файла : lab1_com.asm

```
.model tiny

.code
    org 100h

main proc far
    xor ax, ax
    push ds
    push ax

    call pctype
    mov ah, 9
    int 21h

    mov ah, 30h
    int 21h

    call type_sys_info

    retf
main endp

byte_to_dec proc near
    push ax
    push bx
    push cx

    xor ah, ah
    mov bx, 10
    xor si, si
    add si, 1

    mov cx, 2
c1:
    div bl
    add ah, '0'
    push bx
    mov bx, dx
    mov [bx + si], ah
    pop bx
    dec si
    xor ah, ah
    loop c1

    pop cx
    pop bx
    pop ax
    retn
byte_to_dec endp

byte_to_hex proc near
    push ax
    push bx
```

```

push cx

xor ah, ah
mov bx, 16
xor si, si
add si, 1

mov cx, 2
c2:
    div bl
    cmp ah, 10
    jl digit
    add ah, 7
digit:
    add ah, '0'
    push bx
    mov bx, dx
    mov [bx + si], ah
    pop bx
    dec si
    xor ah, ah
    loop c2

pop cx
pop bx
pop ax
retn
byte_to_hex endp

pctype proc near
push ds
mov ax, 0F000h
mov ds, ax
xor ax, ax
mov ax, [0FFFEh]
pop ds

mov bx, 0F8h
mov cx, 8
check_array:
    cmp bl, al
    je eject
    inc bx
    loop check_array
jmp default

eject:
    sub bx, 0F8h
    shl bx, 1
    add bx, offset types_arr

    mov dx, [bx]
    jmp exit

default:
    mov dx, offset defaultmessage
    add dx, 25
    call byte_to_hex
    sub dx, 25

exit:

```

```

        retn
pctype endp

type_sys_info proc near
    mov dx, offset dosver
    add dx, 12
    call byte_to_dec

    mov al, ah
    add dx, 3
    call byte_to_dec
    sub dx, 15

    mov ah, 9
    int 21h

    mov al, bh
    mov dx, offset oems
    add dx, 20
    call byte_to_hex
    sub dx, 20

    int 21h

    mov al, bl
    mov dx, offset users
    add dx, 29
    call byte_to_hex
    mov al, ch
    add dx, 2
    call byte_to_hex
    mov al, cl
    add dx, 2
    call byte_to_hex
    sub dx, 33

    int 21h

    retn
type_sys_info endp

pctype1 db 'PC2 модель 80', 13, 10, '$'
pctype2 db 'PC Convertible', 13, 10, '$'
pctype3 db 'PC2 модель 30', 13, 10, '$'
pctype4 db 'PC/XT (FB)', 13, 10, '$'
pctype5 db 'АТ либо PC2 модель 50 или 60', 13, 10, '$'
pctype6 db 'PCjr', 13, 10, '$'
pctype7 db 'PC/XT (FE)', 13, 10, '$'
pctype8 db 'PC', 13, 10, '$'

dosver db 'Версия DOS: ', 2 dup(?), '.', 2 dup(?), 13, 10, '$'
oems db 'Серийный номер OEM: ', 2 dup(?), ' HEX', 13, 10, '$'
users db 'Серийный номер пользователя: ', 6 dup(?), ' HEX$'

defaultmessage db 'Неизвестный номер модели ', 2 dup(?), 13, 10, '$'

types_arr dw pctype1, pctype2, pctype3, pctype4, pctype5, pctype6,
pctype7, pctype8

end main

```

- имя файла : lab1_exe.asm

```

AStack segment stack
    dw 64 dup(?)
AStack ends

data segment
    pctype1 db 'PC2 модель 80', 13, 10, '$'
    pctype2 db 'PC Convertible', 13, 10, '$'
    pctype3 db 'PC2 модель 30', 13, 10, '$'
    pctype4 db 'PC/XT (FB)', 13, 10, '$'
    pctype5 db 'AT либо PC2 модель 50 или 60', 13, 10, '$'
    pctype6 db 'PCjr', 13, 10, '$'
    pctype7 db 'PC/XT (FE)', 13, 10, '$'
    pctype8 db 'PC', 13, 10, '$'

    dosver db 'Версия DOS: ', 2 dup(?), '.', 2 dup(?), 13, 10, '$'
    oems db 'Серийный номер OEM: ', 2 dup(?), ' HEX', 13, 10, '$'
    users db 'Серийный номер пользователя: ', 6 dup(?), ' HEX$'

    defaultmessage db 'Неизвестный номер модели ', 2 dup(?), 13, 10, '$'

    types_arr dw pctype1, pctype2, pctype3, pctype4, pctype5, pctype6,
    pctype7, pctype8 ; массив строк
data ends

code segment
    assume ss:AStack, cs:code, ds:data

    byte_to_dec proc near
        push ax
        push bx
        push cx

        xor ah, ah
        mov bx, 10 ; делитель
        xor si, si
        add si, 1 ; заполняется с конца

        mov cx, 2
    c1:
        div bl
        add ah, '0' ; получается код соответствующей цифры
        push bx
        mov bx, dx
        mov [bx + si], ah ; заносим символ в строку
        pop bx
        dec si
        xor ah, ah
        loop c1

        pop cx
        pop bx
        pop ax
        retn
    byte_to_dec endp

    byte_to_hex proc near

```

```

push ax
push bx
push cx

xor ah, ah
mov bx, 16
xor si, si
add si, 1

mov cx, 2
c2:
    div bl
    cmp ah, 10
    jl digit
    add ah, 7          ; всё аналогично, но если цифра >= 10,
digit:                ; дополнительно прибавляется 7, и
получается код соответствующей буквы
    add ah, '0'
    push bx
    mov bx, dx
    mov [bx + si], ah
    pop bx
    dec si
    xor ah, ah
    loop c2

pop cx
pop bx
pop ax
retn
byte_to_hex endp

pctype proc near
mov ax, 0F000h
mov ds, ax
xor ax, ax          ; проверка предпоследнего байта ROM BIOS
mov ax, [0FFFEh]

mov bx, 0F8h
mov cx, 8
check_array:        ; проверка типа ПК на соответствие
имеющимся
    cmp bl, al
    je eject        ; если соответствует
    inc bx
    loop check_array
    jmp default     ; если не подошёл ни один

eject:
    mov ax, data
    mov ds, ax
    sub bx, 0F8h
    shl bx, 1
    add bx, offset types_arr    ; получение нужной строки

    mov dx, [bx]
    jmp exit

default:
    push ax
    mov ax, data

```

```

        mov ds, ax
        pop ax
        mov dx, offset defaultmessage ; запись номера в сообщение о
неизвестном типе ПК
        add dx, 25
        call byte_to_hex
        sub dx, 25

        exit:
        retn
pctype endp

type_sys_info proc near
    mov dx, offset dosver
    add dx, 12
    call byte_to_dec

    mov al, ah
    add dx, 3
    call byte_to_dec
    sub dx, 15

    mov ah, 9
    int 21h

    mov al, bh
    mov dx, offset oems                ; последовательное занесение
байтов                                ; с информацией о системе в
AL                                     ; и перевод в нужные СС с записью в
строки                                ;
    call byte_to_hex
    sub dx, 20

    int 21h

    mov al, bl
    mov dx, offset users
    add dx, 29
    call byte_to_hex
    mov al, ch
    add dx, 2
    call byte_to_hex
    mov al, cl
    add dx, 2
    call byte_to_hex
    sub dx, 33

    int 21h

    retn
type_sys_info endp

main proc far
    xor ax, ax
    push ds
    push ax

    call pctype                        ; получение типа ПК и вывод на
экран

```



```

        mov ah, 9
        int 21h

        mov ah, 30h                ; получение информации о версии ОС
        int 21h

        call type_sys_info         ; её вывод на экран

        retf
    main endp
code ends
end main

```