**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**

Студентка гр. 7381 Кревчик А.Б.

Преподаватель Ефремов М.А.

Санкт-Петербург

2019

**Цель работы.**

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

**Ход работы.**

TETR\_TO\_HEX – вспомогательная функция, переводит из двоичной в шестнадцатеричную систему.

BYTE\_TO\_HEX – переводит число из регистра AL в шестнадцатиричную систему.

WRD\_TO\_HEX – переводит число из регистра AX в шестнадцатиричную систему.

BYTE\_TO\_DEC – переводит число из регистра AL в десятичную систему.

PRINT – печатает сообщение на экран.

TYPE\_PC – получает тип ПК.

PRINT\_TYPE\_PC – определяет и выводит тип ПК.

VERSION\_DOS – определяет версию системы.

OEM\_NUM – определяет серийный номер ОЕМ.

USER\_NUM - определяет серийный модуль пользователя.

Программа выводит на экран тип IBM PC, версию ОС, серийный номер OEM и серийный номер пользователя.

Результаты работы программы представлены на рис. 1-3.



Рисунок 1 – Результат выполнения «плохого» .EXE модуля

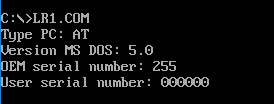


Рисунок 2 – Результат выполнения «хорошего» .СOM модуля

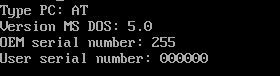


Рисунок 3 – Результат выполнения «хорошего» .EXE модуля

**Выводы.**

В ходе лабораторной работы были изучены различия в структурах исходных текстов модулей типов .COM и .EXE, структур файлов загрузочных модулей и способов их загрузки в основную память.

**Ответы на контрольные вопросы.**

**Отличия исходных текстов COM и EXE программ.**

1. Сколько сегментов должна содержать COM-программа?

Ровно один сегмент – сегмент кода.

1. EXE-программа?

Один и больше.

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

ORG – сдвигает адресацию в программе на 256 бай для расположения PSP,

ASSUME – ставит сегментным регистрам в соответствие требуемые сегменты.

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

Нельзя использовать команды с дальней адресацией, поскольку в COM-программе отсутствует таблица настроек, которая указывает, какие абсолютные адреса при загрузке должны быть изменены, так как до загрузки неизвестно, куда будет загружена программа.

**Отличия форматов файлов COM и EXE модулей.**

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

COM-файл содержит данные и машинные команды. Код начинается с адреса 0h.

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

В «плохом» файле EXE данные и код содержатся в одном сегменте. Код располагается с адреса 300h. С адреса 0h располагается заголовок, таблица настроек, а также зарезервированные директивой ORG 100h байт.

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

В «хорошем» EXE код, стек и данные выделены в отдельные сегменты, тогда как в «плохо» всего один сегмент и для данных и для кода

Также, в «хорошем» EXE с адреса 200h идет сегмент стека, когда в «плохом» здесь располагаются зарезервированные ORG 100h байт и нет сегмента стека. В EXE программах нет необходимости в директиве ORG, поскольку загрузчик ставит программу после PSP.

**Загрузка COM модуля в основную память.**

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

После загрузки COM-программы в память, сегментные регистры указывают на начало PSP. Начало кода определяется директивой ORG от начала выделенного фрагмента (100h).

1. Что располагается с адреса 0?

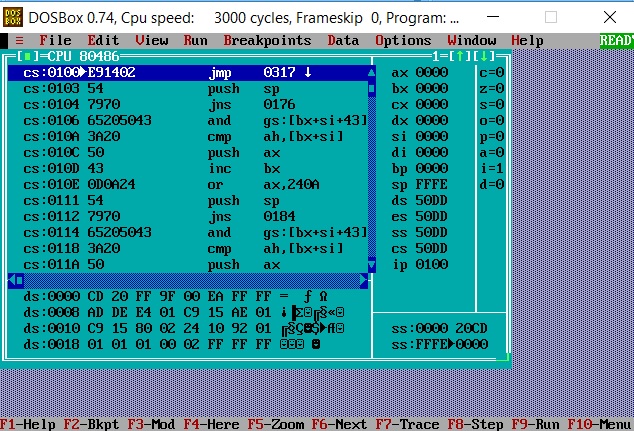
PSP.

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

Все сегментные регистры указывают на начало PSP.

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

Стек занимает весь фрагмент памяти, выделенный под программу и определяется регистрами SS и SP. Он занимает адреса 0000h-FFFEh.



**Загрузка «хорошего» EXE модуля в основную память.**

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

DS и ES устанавливаются на начало сегмента PSP, SS – на начало сегмента стека, CS – на начало сегмента кода. В IP загружается смещение точки входа в программу, которая берётся из метки после директивы END.

1. На что указывают регистры DS и ES?

Начало сегмента PSP.

1. Как определяется стек?

Для стека в программе выделяется отдельный сегмент с параметром STACK. При запуске программы в SS заносится адрес сегмента стека, а в SP – адрес верхушки стека.

1. Как определяется точка входа?

С помощью директивы END, после которой указывается метка, куда переходит программа при запуске.

ПРИЛОжение а

исходный код .Сom модуля

TESTPC SEGMENT

ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING

ORG 100H

START: JMP BEGIN

;„ЂЌЌ›…

PC db 'Type PC: PC', 13, 10 ,'$'

PC\_XT db 'Type PC: PC/XT', 13, 10, '$'

AT db 'Type PC: AT', 13, 10, '$'

PS2\_1 db 'Type PC: PS2 model 30', 10, 13, '$'

PS2\_2 db 'Type PC: PS2 model 50 or 60', 10, 13, '$'

PS2\_3 db 'Type PC: PS2 model 80', 10, 13, '$'

PCjr db 'Type PC: PCjr',10, 13, '$'

PC\_CONVERTIBLE db 'Type PC: PC Convertible',10, 13, '$'

VERS db 'Version MS DOS: . ', 10, 13, '$'

OEM db 'OEM serial number: ', 10, 13, '$'

USER db 'User serial number: ', 10, 13, '$'

;ЏђЋ–…„“ђ›

TETR\_TO\_HEX PROC NEAR

and AL,0Fh

cmp AL,09

jbe NEXT

add AL,07

NEXT: add AL,30h

ret

TETR\_TO\_HEX ENDP

BYTE\_TO\_HEX PROC NEAR ;Ў ©в ў AL ЇҐаҐў®¤Ёвбп ў ¤ў  бЁ¬ў®«  иҐбв­. зЁб«  ў AX

push CX

mov AH,AL

call TETR\_TO\_HEX

xchg AL,AH

mov CL,4

shr AL,CL

call TETR\_TO\_HEX

pop CX

ret

BYTE\_TO\_HEX ENDP

WRD\_TO\_HEX PROC NEAR ;ЇҐаҐў®¤ ў 16 б/б 16-вЁ а §ап¤­®Ј® зЁб« , ў AX - зЁб«®, DI -  ¤аҐб Ї®б«Ґ¤­ҐЈ® бЁ¬ў®«

push BX

mov BH,AH

call BYTE\_TO\_HEX

mov [DI],AH

dec DI

mov [DI],AL

dec DI

mov AL,BH

xor AH,AH

call BYTE\_TO\_HEX

mov [DI],AH

dec DI

mov [DI],AL

pop BX

ret

WRD\_TO\_HEX ENDP

BYTE\_TO\_DEC PROC NEAR

push AX

push CX

push DX

xor AH,AH

xor DX,DX

mov CX,10

loop\_bd: div CX

or DL,30h

mov [SI],DL

dec SI

xor DX,DX

cmp AX,10

jae loop\_bd

cmp AL,00h

je end\_l

or AL,30h

mov [SI],AL

end\_l: pop DX

pop CX

pop AX

ret

BYTE\_TO\_DEC ENDP

PRINT PROC NEAR

push AX

mov AH, 09h

int 21h

pop AX

ret

PRINT ENDP

TYPE\_PC PROC NEAR ;Ї®«гзҐ­ЁҐ вЁЇ  ЏЉ

push DS

mov BX,0F000H

mov DS,BX

sub AX,AX

mov AH,DS:[0FFFEH]

pop DS

ret

TYPE\_PC ENDP

PRINT\_TYPE\_PC PROC NEAR ;®ЇаҐ¤Ґ«Ґ­ЁҐ Ё ўлў®¤ вЁЇ  ЏЉ

push AX

push BX

push DI

mov DX, OFFSET PC

cmp AH, 0FFh

je print\_msg

mov DX, OFFSET PC\_XT

cmp AH, 0FEh

je print\_msg

mov DX, OFFSET PC\_XT

cmp AH, 0FBh

je print\_msg

mov DX, OFFSET AT

cmp AH, 0FCh

je print\_msg

mov DX, OFFSET PS2\_1

cmp AH, 0FAh

je print\_msg

mov DX, OFFSET PS2\_2

cmp AH, 0FCh

je print\_msg

mov DX, OFFSET PS2\_3

cmp AH, 0F8h

je print\_msg

mov DX, OFFSET PCjr

cmp AH, 0FDh

je print\_msg

mov DX, OFFSET PC\_CONVERTIBLE

cmp AH, 0F9h

je print\_msg

mov AL,AH

call BYTE\_TO\_HEX

mov DX, AX

print\_msg:

call PRINT

pop DI

pop BX

pop AX

ret

PRINT\_TYPE\_PC ENDP

VERSION\_DOS PROC NEAR ;®ЇаҐ¤Ґ«Ґ­ЁҐ ўҐабЁЁ бЁбвҐ¬л

push AX

push SI

mov SI, OFFSET VERS

add SI, 10h

call BYTE\_TO\_DEC

add SI, 3h

mov AL, AH

call BYTE\_TO\_DEC

mov DX, OFFSET VERS

call PRINT

pop SI

pop AX

ret

VERSION\_DOS ENDP

OEM\_NUM PROC NEAR ;®ЇаҐ¤Ґ«Ґ­ЁҐ бҐаЁ©­®Ј® ­®¬Ґа  OEM

push AX

push BX

push SI

mov AL, BH

mov SI, OFFSET OEM

add SI, 15h

call BYTE\_TO\_DEC

mov DX, OFFSET OEM

call PRINT

pop SI

pop BX

pop AX

ret

OEM\_NUM ENDP

USER\_NUM PROC NEAR ;®ЇаҐ¤Ґ«Ґ­ЁҐ бҐаЁ©­®Ј® ­®¬Ґа  Ї®«м§®ў вҐ«п

push CX

push DI

push AX

mov DI, OFFSET USER

add DI, 19h

mov AX, CX

call WRD\_TO\_HEX

mov AL, BL

mov DI, OFFSET USER

add DI, 14h

call BYTE\_TO\_HEX

mov [DI], AX

mov DX, OFFSET USER

call PRINT

pop AX

pop DI

pop CX

ret

USER\_NUM ENDP

BEGIN:

call TYPE\_PC

call PRINT\_TYPE\_PC

mov AH, 30h

int 21h

call VERSION\_DOS

call OEM\_NUM

call USER\_NUM

xor AL,AL

mov AH,4Ch

int 21h

TESTPC ENDS

END START

ПРИЛОжение Б

исходный код .EXE модуля

ASTACK SEGMENT STACK

DW 0100h DUP(?)

ASTACK ENDS

DATA SEGMENT

;„ЂЌЌ›…

PC db 'Type PC: PC', 13, 10 ,'$'

PC\_XT db 'Type PC: PC/XT', 13, 10, '$'

AT db 'Type PC: AT', 13, 10, '$'

PS2\_1 db 'Type PC: PS2 ¬®¤Ґ«м 30', 10, 13, '$'

PS2\_2 db 'Type PC: PS2 ¬®¤Ґ«м 50 or 60', 10, 13, '$'

PS2\_3 db 'Type PC: PS2 ¬®¤Ґ«м 80', 10, 13, '$'

PCjr db 'Type PC: PCjr',10, 13, '$'

PC\_CONVERTIBLE db 'Type PC: PC Convertible',10, 13, '$'

VERS db 'Version MS DOS: . ', 10, 13, '$'

OEM db 'OEM serial number: ', 10, 13, '$'

USER db 'User serial number: ', 10, 13, '$'

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA, ES:NOTHING, SS:ASTACK

;ЏђЋ–…„“ђ›

START: JMP BEGIN

TETR\_TO\_HEX PROC NEAR

and AL,0Fh

cmp AL,09

jbe NEXT

add AL,07

NEXT: add AL,30h

ret

TETR\_TO\_HEX ENDP

BYTE\_TO\_HEX PROC NEAR ;Ў ©в ў AL ЇҐаҐў®¤Ёвбп ў ¤ў  бЁ¬ў®«  иҐбв­. зЁб«  ў AX

push CX

mov AH,AL

call TETR\_TO\_HEX

xchg AL,AH

mov CL,4

shr AL,CL

call TETR\_TO\_HEX ;ў AL бв аи п жЁда

pop CX ;ў AH ¬« ¤и п

ret

BYTE\_TO\_HEX ENDP

WRD\_TO\_HEX PROC NEAR ;ЇҐаҐў®¤ ў 16 б/б 16-вЁ а §ап¤­®Ј® зЁб« , ў AX - зЁб«®, DI -  ¤аҐб Ї®б«Ґ¤­ҐЈ® бЁ¬ў®«

push BX

mov BH,AH

call BYTE\_TO\_HEX

mov [DI],AH

dec DI

mov [DI],AL

dec DI

mov AL,BH

xor AH,AH

call BYTE\_TO\_HEX

mov [DI],AH

dec DI

mov [DI],AL

pop BX

ret

WRD\_TO\_HEX ENDP

BYTE\_TO\_DEC PROC NEAR ; ЇҐаҐў®¤ Ў ©в  ў 10б/б, SI -  ¤аҐб Ї®«п ¬« ¤иҐ© жЁдал

push AX ; AL б®¤Ґа¦Ёв Ёбе®¤­л© Ў ©в

push CX

push DX

xor AH,AH

xor DX,DX

mov CX,10

loop\_bd: div CX

or DL,30h

mov [SI],DL

dec SI

xor DX,DX

cmp AX,10

jae loop\_bd

cmp AL,00h

je end\_l

or AL,30h

mov [SI],AL

end\_l: pop DX

pop CX

pop AX

ret

BYTE\_TO\_DEC ENDP

PRINT PROC NEAR ;ЇҐз вм б®®ЎйҐ­Ёп ­  нЄа ­

push AX

mov AH, 09h

int 21h

pop AX

ret

PRINT ENDP

TYPE\_PC PROC NEAR ;Ї®«гзҐ­ЁҐ вЁЇ  ЏЉ

push DS

mov BX,0F000H

mov DS,BX

sub AX,AX

mov AH,DS:[0FFFEH]

pop DS

ret

TYPE\_PC ENDP

PRINT\_TYPE\_PC PROC NEAR ;ўлў®¤ вЁЇ  ЏЉ

push AX

push BX

push DI

mov DX, OFFSET PC

cmp AH, 0FFh

je print\_msg

mov DX, OFFSET PC\_XT

cmp AH, 0FEh

je print\_msg

mov DX, OFFSET PC\_XT

cmp AH, 0FBh

je print\_msg

mov DX, OFFSET AT

cmp AH, 0FCh

je print\_msg

mov DX, OFFSET PS2\_1

cmp AH, 0FAh

je print\_msg

mov DX, OFFSET PS2\_2

cmp AH, 0FCh

je print\_msg

mov DX, OFFSET PS2\_3

cmp AH, 0F8h

je print\_msg

mov DX, OFFSET PCjr

cmp AH, 0FDh

je print\_msg

mov DX, OFFSET PC\_CONVERTIBLE

cmp AH, 0F9h

je print\_msg

mov AL,AH

call BYTE\_TO\_HEX

mov DX, AX

print\_msg:

call PRINT

pop DI

pop BX

pop AX

ret

PRINT\_TYPE\_PC ENDP

VERSION\_DOS PROC NEAR

push AX

push SI

mov SI, OFFSET VERS

add SI, 10h

call BYTE\_TO\_DEC

add SI, 3h

mov AL, AH

call BYTE\_TO\_DEC

mov DX, OFFSET VERS

call PRINT

pop SI

pop AX

ret

VERSION\_DOS ENDP

OEM\_NUM PROC NEAR

push AX

push BX

push SI

mov AL, BH

mov SI, OFFSET OEM

add SI, 15h

call BYTE\_TO\_DEC

mov DX, OFFSET OEM

call PRINT

pop SI

pop BX

pop AX

ret

OEM\_NUM ENDP

USER\_NUM PROC NEAR

push CX

push DI

push AX

mov DI, OFFSET USER

add DI, 19h

mov AX, CX

call WRD\_TO\_HEX

mov AL, BL

mov DI, OFFSET USER

add DI, 14h

call BYTE\_TO\_HEX

mov [DI], AX

mov DX, OFFSET USER

call PRINT

pop AX

pop DI

pop CX

ret

USER\_NUM ENDP

BEGIN:

mov AX, DATA

mov DS, AX

mov BX, DS

call TYPE\_PC

call PRINT\_TYPE\_PC

mov AH, 30h

int 21h

call VERSION\_DOS

call OEM\_NUM

call USER\_NUM

xor AL, AL

mov AH, 4Ch

int 21h

CODE ENDS

END START