**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по практической работе № 2**

**по дисциплине «Операционные системы»**

Тема: **Исследование интерфейсов программных модулей**

|  |  |  |
| --- | --- | --- |
| Студент гр. 9382 |  | Русинов Д.А. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2021

**Цель работы.**

Исследование интерфейса управляющей программы и загрузочных модулей. Этот интерфейс состоит в передаче запускаемой программе управляющего блока, содержащего адреса и системные данные. Так загрузчик строит префикс сегмента программы (PSP) и помещает его адрес в сегментный регистр. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

**Задание.**

Шаг 1. Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .COM, который выбирает и распечатывает следующую информацию:

1. Сегментный адрес недоступной памяти, взятый из PSP, в шестнадцатеричном виде.
2. Сегментный адрес среды, передаваемой программе, в шестнадцатеричном виде.
3. Хвост командной строки в символьном виде.
4. Содержимое области среды в символьном виде.
5. Путь загружаемого модуля.

Сохраните результаты, полученные программой, и включите их в отчет.

Шаг 2. Оформление отчета в соответствии с требованиями. В отчет включите скриншот с запуском программы и результатами.

**Выполнение работы.**

Была разработана .COM-программа, которая выполняет задание. Результаты выполнения программы приведены для запуска модуля с аргументами и без.





**Ответы на вопросы.**

**Сегментный адрес недоступной памяти**

1. На какую область памяти указывает адрес недоступной памяти?

На первый байт после участка памяти, отведенного под программу

2) Где расположен этот адрес по отношению области памяти, отведенной

программе?

Адрес недоступной памяти по отношению к области памяти,

отведенной программе, расположен после. В сторону увеличения адресов.

1. Можно ли в эту область памяти писать?

Можно, так как DOS не имеет механизмов защиты перезаписи памяти программами, для которых эта память не выделялась.

**Среда передаваемая программе**

1. Что такое среда?

Участок памяти, который содержит переменные среды. Переменные среды хранят некоторую информацию о состоянии системы.

1. Когда создается среда? Перед запуском приложения или в другое время?

Среда создается при запуске ОС. Среда копируется в адресное пространство запущенной программы. Среда может изменяться в соответствии с требованиями программы.

1. Откуда берется информация, записываемая в среду?

Из файла AUTOEXEC.BAT. Он расположен в корневом каталоге загрузочного устройства.

**Выводы.**

Был исследован интерфейс управляющей программы и загрузочных модулей. Был исследован префикс сегмента программы (PSP) и среды, передаваемой программе.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл: LAB2COM.ASM

TESTPC SEGMENT

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

ORG 100H

STARTUP: JMP MAIN

ENV\_ADDR DB 'Environment address:', '$'

UNAVAILABLE DB 'Unavailable memory:', '$'

AX\_REGISTER DB ' ', 0DH, 0AH, '$'

EMPTY DB 'Empty', 0DH, 0AH, '$'

TAIL DB 'Command tail is: ', '$'

ENV DB 'Environment content: ', '$'

ENV\_TAB DB ' ', '$'

NEW\_LINE DB 0DH, 0AH, '$'

PATH DB 'Path is: ', '$'

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

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

PUSH BX

MOV BH, AH

CALL BYTE\_TO\_HEX

MOV [DI], AH

DEC DI

MOV [DI], AL

DEC DI

MOV AL, BH

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 SI

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 SI

RET

BYTE\_TO\_DEC ENDP

WRITESTRING PROC NEAR

PUSH AX

MOV AH, 09H

INT 21H

POP AX

RET

WRITESTRING ENDP

CLEARSTRING PROC NEAR

PUSH SI

PUSH CX

XOR SI, SI

MOV CX, 5

CLEARING:

MOV [OFFSET AX\_REGISTER+SI], 0FF20h

INC SI

LOOP CLEARING

MOV [OFFSET AX\_REGISTER+SI], 0FF20h

POP CX

POP SI

RET

CLEARSTRING ENDP

\_PRINT\_INFO PROC NEAR

CALL WRITESTRING

MOV DI, OFFSET AX\_REGISTER

ADD DI, 5

CALL WRD\_TO\_HEX

MOV DX, OFFSET AX\_REGISTER

CALL WRITESTRING

CALL CLEARSTRING

RET

\_PRINT\_INFO ENDP

PRINT\_UNAVAILABLE\_MEMORY PROC NEAR

MOV DX, OFFSET UNAVAILABLE

MOV AX, DS:[02h]

CALL \_PRINT\_INFO

RET

PRINT\_UNAVAILABLE\_MEMORY ENDP

PRINT\_ENV\_ADDR PROC NEAR

MOV DX, OFFSET ENV\_ADDR

MOV AX, DS:[2Ch]

CALL \_PRINT\_INFO

RET

PRINT\_ENV\_ADDR ENDP

PRINT\_COMMAND\_TAIL PROC NEAR

PUSH CX

PUSH AX

XOR CX, CX

MOV DX, OFFSET TAIL

CALL WRITESTRING

MOV CL, DS:[80h]

CMP CL, 0

JE \_EMPTY\_TAIL

MOV SI, 0

\_PRINT\_TAIL\_SYMBOL:

MOV DL, DS:[81h+SI]

MOV AH, 02h

INT 21h

INC SI

LOOP \_PRINT\_TAIL\_SYMBOL

JMP \_EXIT\_TAIL\_PRINT

\_EMPTY\_TAIL:

MOV DX, OFFSET EMPTY

CALL WRITESTRING

\_EXIT\_TAIL\_PRINT:

POP AX

POP CX

RET

PRINT\_COMMAND\_TAIL ENDP

PRINT\_ENVIRONMENT PROC NEAR

PUSH DX

PUSH AX

PUSH SI

PUSH DS

XOR SI, SI

MOV DX, OFFSET ENV

CALL WRITESTRING

MOV DS, DS:[2CH]

\_READ\_ENV:

MOV DL, [SI]

CMP DL, 0

JE \_EOF

MOV AH, 02h

INT 21h

INC SI

JMP \_READ\_ENV

\_EOF:

INC SI

MOV DL, [SI]

CMP DL, 0

JE \_END\_READING\_ENV

POP DS

MOV DX, OFFSET NEW\_LINE

CALL WRITESTRING

MOV DX, OFFSET ENV\_TAB

CALL WRITESTRING

PUSH DS

MOV DS, DS:[2Ch]

JMP \_READ\_ENV

\_END\_READING\_ENV:

POP DS

MOV DX, OFFSET NEW\_LINE

CALL WRITESTRING

MOV DX, OFFSET PATH

CALL WRITESTRING

PUSH DS

MOV DS, DS:[2Ch]

ADD SI, 3

\_READING\_PATH:

MOV DL, [SI]

CMP DL, 0

JE \_EXIT\_PRINT\_ENV

MOV AH, 02h

INT 21h

INC SI

JMP \_READING\_PATH

\_EXIT\_PRINT\_ENV:

POP DS

POP SI

POP AX

POP DX

RET

PRINT\_ENVIRONMENT ENDP

MAIN:

CALL PRINT\_UNAVAILABLE\_MEMORY

CALL PRINT\_ENV\_ADDR

CALL PRINT\_COMMAND\_TAIL

CALL PRINT\_ENVIRONMENT

XOR AL, AL

MOV AH, 4CH

INT 21H

TESTPC ENDS

END STARTUP