**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Операционные системы»**

Тема: **Исследование интерфейсов командных модулей**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8383 |  | Костарев К.В. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2020

**Цель работы.**

Исследование интерфейса управляющей программы и загрузочных модулей, а также префикса сегмента программы и среды, передаваемой программе.

**Основные теоретические сведения.**

При начальной загрузке программы формируется PSP, который размещается в начале первого сегмента программы. PSP занимает 256 байт и располагается с адреса, кратного границе сегмента. При загрузке модулей типа .COM все сегментные регистры указывают на адрес PSP. Формат PSP представлен в табл. 1.

Таблица 1 – Формат PSP

|  |  |  |
| --- | --- | --- |
| Смещение | Длина поля (байт) | Содержимое поля |
| 0 | 2 | int 20h |
| 2 | 2 | Сегментный адрес первого байта недоступной памяти |
| 4 | 6 | Зарезервировано |
| 0Ah | 4 | Вектор прерывания 22h |
| 0Eh | 4 | Вектор прерывания 23h |
| 12h | 4 | Вектор прерывания 24h |
| 2Ch | 2 | Сегментный адрес среды, передаваемой программе |
| 5Ch |  | Форматируется как стандартный неоткрытый блок управления файлом (FCB) |
| 6Ch |  | Перекрывается, если FCB с адреса 5Ch открыт |
| 80h | 1 | Число символов в хвосте командной строки |
| 81h |  | Хвост командной строки |

**Выполнение работы.**

Для выполнения лабораторной работы был написан и отлажен программный модуль типа .COM, который выбирает и печатает на экран следующую информацию:

1. Сегментный адрес недоступной памяти 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.

Результат работы программы представлен на рис. 1, исходный код программы в Приложении А.

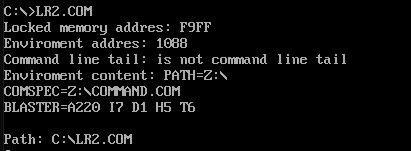


Рисунок 1 – Результат работы программы

Сегментный адрес недоступной памяти:

1. *На какую область памяти указывает адрес недоступной памяти?*

Адрес недоступной памяти указывает на служебную часть памяти, эту память DOS не может выделить под программу. Сам адрес указывает на сегментный адрес последнего параграфа памяти, используемого DOS для запуска программ.

1. *Где расположен этот адрес по отношению к области памяти, отведенной программе?*

Расположен сразу за областью памяти, которую DOS отводит пользовательским программам.

1. *Можно ли в эту область памяти писать?*

Возможно, так как DOS не контролирует обращение программ к памяти.

Среда, передаваемая программе:

1. *Что такое среда?*

Это последовательность символьных строк вида [имя] = [параметр]. COMSPEC определяет путь к COMMAND.COM, PATH определяет пути к программным файлам, которые будут вызваны. Признаком конца строки является байт нулей.

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

При запуске DOS создается корневая среда, относящаяся к COMMAND.COM. Затем, когда COMMAND.COM запускает пользовательскую программу или одна программа запускает другую создается процесс, который получает собственный экземпляр блока среды, по умолчанию являющийся копией среды родителя.

1. *Откуда берется информация, записываемая в среду?*

Из среды родителя.

**Выводы.**

В ходе выполнения данной лабораторной работы был исследован интерфейс управляющей программы и загрузочных модулей, а также префикс сегмента программы и среды, передаваемой программе. Для этого был написан программный модуль типа .COM, который непосредственно обращается к некоторым сегментам памяти в PSP и выводит на экран информацию из них.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД РЕАЛИЗОВАННОЙ ПРОГРАММЫ

LR2 SEGMENT

ASSUME CS:LR2, DS:LR2, ES:NOTHING, SS:NOTHING

ORG 100H

START: JMP BEGIN

ENTER db 13, 10, '$'

LOCKMEMORY db 'Locked memory addres: $'

ENVIROMENT db 'Enviroment addres: $'

TAIL db 'Command line tail: $'

NO\_TAIL db 'is not command line tail $'

ENVIROMENT\_CONTENT db 'Enviroment content: $'

PATH db 'Path: $'

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

WRITE\_SYMBOL PROC near

push AX

mov AH, 02H

int 21H

pop AX

ret

WRITE\_SYMBOL ENDP

WRITE\_STRING PROC near

push AX

mov AH, 09H

int 21H

pop AX

ret

WRITE\_STRING ENDP

WRITE\_HEX PROC near

push AX

mov AL, AH

call BYTE\_TO\_HEX

mov DL, AH

call WRITE\_SYMBOL

mov DL, AL

call WRITE\_SYMBOL

pop AX

call BYTE\_TO\_HEX

mov DL, AH

call WRITE\_SYMBOL

mov DL, AL

call WRITE\_SYMBOL

ret

WRITE\_HEX ENDP

WRITE\_LOCKMEMORY PROC near

push AX

push DX

mov DX, offset LOCKMEMORY

call WRITE\_STRING

mov AX, DS:[02H]

call WRITE\_HEX

mov DX, offset ENTER

call WRITE\_STRING

pop DX

pop AX

ret

WRITE\_LOCKMEMORY ENDP

WRITE\_ENVIROMENT PROC near

push AX

push DX

mov DX, offset ENVIROMENT

call WRITE\_STRING

mov AX, DS:[2CH]

call WRITE\_HEX

mov DX, offset ENTER

call WRITE\_STRING

pop DX

pop AX

ret

WRITE\_ENVIROMENT ENDP

WRITE\_TAIL PROC near

push AX

push SI

push CX

push DX

mov DX, offset TAIL

call WRITE\_STRING

xor CX, CX

mov CL, DS:[80H]

cmp CL, 0

jne RETAIL

mov DX, offset NO\_TAIL

call WRITE\_STRING

jmp POPING\_TAIL

RETAIL:

xor SI, SI

xor AX, AX

WRITING\_TAIL:

mov AL, DS:[81H + SI]

call WRITE\_SYMBOL

inc SI

loop WRITING\_TAIL

POPING\_TAIL:

mov DX, offset ENTER

call WRITE\_STRING

pop DX

pop CX

pop SI

pop AX

ret

WRITE\_TAIL ENDP

WRITE\_ENVIROMENT\_CONTENT\_AND\_PATH PROC near

push AX

push SI

push DX

push BX

push ES

mov DX, offset ENVIROMENT\_CONTENT

call WRITE\_STRING

xor SI, SI

mov BX, 2CH

mov ES, [BX]

WRITING\_CONTENT:

cmp BYTE PTR ES:[SI], 0H

je NEXT\_CONTENT

mov AL, ES:[SI]

mov DL, AL

call WRITE\_SYMBOL

jmp CHECKING

NEXT\_CONTENT:

mov DX, offset ENTER

call WRITE\_STRING

CHECKING:

inc SI

cmp WORD PTR ES:[SI], 0001H

je WRITE\_PATH

jmp WRITING\_CONTENT

WRITE\_PATH:

mov DX, offset PATH

call WRITE\_STRING

add SI, 2

WRITING\_PATH:

cmp BYTE PTR ES:[SI], 00H

je POPING\_CONTENT

mov AL, ES:[SI]

mov DL, AL

call WRITE\_SYMBOL

inc SI

jmp WRITING\_PATH

POPING\_CONTENT:

pop ES

pop BX

pop DX

pop SI

pop AX

ret

WRITE\_ENVIROMENT\_CONTENT\_AND\_PATH ENDP

BEGIN:

call WRITE\_LOCKMEMORY

call WRITE\_ENVIROMENT

call WRITE\_TAIL

call WRITE\_ENVIROMENT\_CONTENT\_AND\_PATH

xor AL,AL

mov AH,4CH

int 21H

LR2 ENDS

END START