**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. 8381 _____ Почаев Н.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

**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.

**Основные теоретические положения.**

При начальной загрузке программы формируется PSP (Program Segment Prefics) – специальная область оперативной памяти размером 256 (100h) байт. Он располагается с адреса, кратного границе сегмента. PSP может использоваться в программе для определения имен файлов и параметров из командной строки, введенной при запуске программы на выполнение, объема доступной памяти, переменных окружения системы и т.д. Регистр SS при этом инициализируется значением сегмента, находящегося сразу за PSP, т.е. первого сегмента программы.

При этом необходимо учитывать, что стек «растет вниз» (при помещении в стек содержимое регистра SP, указывающего на вершину стека, уменьшается, а при считывании из стека – увеличивается). Таким образом, при помещении в стек каких-либо значений они могут затереть PSP и программы, находящиеся в младших адресах памяти, что может привести к непредсказуемым последствиям. Поэтому рекомендуется всегда явно описывать сегмент стека в тексте программы, задавая ему размер, достаточный для нормальной работы. При загрузке модулей типа COM все сегментные регистры указывают на адрес PSP. При загрузке модуля типа EXE сегментные регистры DS и ES указывают на PSP.

Структура PSP схематически представлена в табл. 1.

Таблица 1 – Структура PSP

| Смещение (HEX) | Длина поля (байт) | Содержимое |
|---|---|---|
| +0 | 2 | INT 21 (EXE-программы могут делать сюда JMP или RET для выхода) |
| +2 | 2 | Вершина доступной памяти системы в параграфах. Программа не должна модифицировать содержимое памяти за этим адресом. |
| +4 | 1 | Зарезервировано |
| +5 | 5 | FAR CALL к диспетчеру функций DOS |
| +6 | | Доступные байты в программном сегменте (только для COM) |
| +0ah | 4 | Адрес прерывания (завершения) 22h (IP, CS) |
| +0eh | 4 | Адрес обработки Ctrl-Break – прерывание 23h (IP, CS) |
| +12h | 4 | Обработчик критических ошибок – вектор прерывания 24h (IP, CS) |
| +16h | 16h | Резервная область DOS |
| +2ch | 2 | Сегментный адрес окружения |
| +2eh | 2eh | Резервная область DOS |
| +5ch | 10h | Область форматируется как стандартный неоткрытый блок управления файлом (FCB) |
| +6ch | 14h | Область форматируется как стандартный неоткрытый блок управления файлом (FCB). Перекрывается, если FCB с адреса 5ch открыт. |

| | | |
|---|---|---|
| +80h | 1 | Длина области UPA (с адреса 81H) также смещение умалчиваемой DTA |
| +81h | 7fh | Хвост командной строки - символы из командной строки DOS (исключая директивы переназначения) |
| 100h | | Размер префикса программного сегмента |

Область среды содержит последовательность символьных строк вида:

имя=параметр

Каждая строка завершается байтом нулей.

В первой строке указывается имя COMSPEC, которая определяет используемый командный процессор и путь к COMMAND.COM. Следующие строки содержат информацию, задаваемую командами PATH, PROMPT, SET. Среда заканчивается также байтом нулей. Таким образом, два нулевых байта являются признаком конца переменных среды. Затем идут два байта, содержащих 00h, 01h, после которых располагается маршрут загруженной программы. Маршрут также заканчивается байтом 00h.

**Постановка задачи.**

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.

**Описание программы.**

В результате выполнения лабораторной работы была написана программа, описание функций которой представлено в таблице ниже.

- TETR_TO_HEX - вспомогательная для byte_to_hex процедура;
- BYTE_TO_HEX - конвертация байта в неупакованный 16-ый формат;
- WRD_TO_HEX - конвертация слова в неупакованный 16-ый формат;
- BYTE_TO_DEC - конвертация байта в неупакованный 10-ый формат;
- FUNCTION_NOT_AVAILABLE_MEMORY - получение адреса недоступной памяти;
- FUNCTION _ENVIRONMENT_SEGMENT_ADDRESS - получение сегментного адреса среды;
- FUNCTION _COMMAND_TAIL - получение хвоста командной строки;
- FUNCTION _ENVIRONMENT_DATA - получение содержимого среды;
- PRINT_STRING - вывод строки на экран.

**Выполнение работы.**

Выполнение работы производилось на базе операционной системы Windows XP (32 bit), запускаемой в системе виртуализации VMware Workstation, в редакторе Notepad++. Сборка и отладка модулей производились с помощью компилятора MASM и отладчика AFD. Также в работе был использован консольный файловый менеджер Far Manager и HEX-редактор HxD.

Был написан текст исходного .COM модуля, который обеспечивает вывод на экран следующей информации:

- Сегментный адрес недоступной памяти, взятый из PSP, в шестнадцатеричном виде;
- Сегментный адрес среды, передаваемой программе, в шестнадцатеричном виде;
- Хвост командной строки, в символьном виде;

- Содержимое области среды в символьном виде;
- Путь загружаемого модуля.

Программа была отлажена и запущена на виртуальной машине, результат её работы приведён на рис. 1.

```
E:\lr2>LR2.COM
Inaccessible  memery address:9FFF
Environment segment address:0518
Command line tail:

Environment data:
COMSPEC=C:\WINDOWS\SYSTEM32\COMMAND.COM
ALLUSERSPROFILE=C:\DOCUME~1\ALLUSE~1
APPDATA=C:\Documents and Settings\Администратор\Application Data
CLIENTNAME=Console
COMMONPROGRAMFILES=C:\PROGRA~1\COMMON~1
COMPUTERNAME=NIKITA-68071BC4
FP_NO_HOST_CHECK=NO
HOMEDRIVE=C:
HOMEPATH=\Documents and Settings\Администратор
LOGONSERVER=\\NIKITA-68071BC4
NUMBER_OF_PROCESSORS=1
OS=Windows_NT
PATH=C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
PROCESSOR_ARCHITECTURE=x86
PROCESSOR_IDENTIFIER=x86 Family 23 Model 1 Stepping 1, AuthenticAMD
PROCESSOR_LEVEL=23
PROCESSOR_REVISION=0101
PROGRAMFILES=C:\PROGRA~1
PROMPT=$P$G
SESSIONNAME=Console
SYSTEMDRIVE=C:
SYSTEMROOT=C:\WINDOWS
TEMP=C:\WINDOWS\TEMP
TMP=C:\WINDOWS\TEMP
USERDOMAIN=NIKITA-68071BC4
USERNAME=Администратор
USERPROFILE=C:\Documents and Settings\Администратор
BLASTER=A220 I5 D1 P330 T3

Start directory:
E:\LR2\LR2.COM
```

Рисунок 1 – Результат выполнения программы

Исходный код модуля приведён к Приложении А.

**Выводы.**

В результате выполнения данной лабораторной работы был исследован интерфейс управляющей программы и загрузочных модулей. Была написана программа, которая выводит на экран сегментный адрес недоступной памяти, взятый из PSP, сегментный адрес среды, передаваемой программе, хвост командной строки и путь загружаемого модуля.

**Ответы на контрольные вопросы.**

**Сегментный адрес недоступной памяти.**

1. На какую область памяти указывает адрес недоступной памяти?

   Первые 640 Кбайт адресного пространства с адресами от 00000h до 9FFFF11 (и, соответственно, с сегментными адресами от 0000h до 9FFFh) отводятся под основную оперативную память, которую еще называют стандартной (conventional). Начальный килобайт оперативной памяти занят векторами прерываний, которые обеспечивают работу системы прерываний компьютера, и включает 256 векторов по 4 байта каждый. Данная область также является служебной – в ней DOS не может выделить память для пользовательской программы.

2. Где расположен этот адрес по отношению области памяти, отведенной программе?

   Адрес недоступной памяти указывает на последний параграф памяти, отведённый для программы, как видно их табл. 1 и ответа на 1-ый вопрос, недоступная память располагается сразу после.

3. Можно ли в эту область памяти писать?

   Так как операционная система DOS использует «реальный» режим процессора (или режим реальных адресов; англ. real-address mode), в котором любому процессу доступна вся память, то запись возможна. Несмотря на то, что в MS-DOS имеются функции управления памятью, с помощью которых программы могут получить в свое распоряжение блоки памяти, ничто не помешает программе выполнить запись за пределами полученного блока или даже в системную область памяти, разрушив MS-DOS. Если в мультизадачной среде одна задача может писать данные в область памяти, отведенной другой задаче, она может разрушить и эту задачу, и ядро операционной системы. Поэтому в мультизадачных операционных системах, разработанных для процессоров серии Intel 80xxx или Pentium, применяется только защищенный режим работы процессора.

**Среда, передаваемая программе**

1. Что такое среда?

   Среда – область памяти, в которой в виде символьных строк хранятся значения переменных среды в формате «параметр=значение», 0, передаваемая программе при ее запуске.

   Например, в первой строке указывается имя COMSPEC, которая определяет используемый командный процессор и путь к COMMAND.COM.

   Следующие строки содержат информацию, задаваемую командами PATH, PROMPT, SET.

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

   Интерпретатор команд COMMAND.COM имеет свою среду, которую называют корневой средой. Она создается при запуске DOS. Передаваемая программе при запуске среда является копией среды родительского процесса.

   Поэтому, если COMMAND.COM запускает программу, то ей передается копия корневой среды.

3. Откуда берется информация, записываемая в среду?

   Файл AUTOEXEX.BAT - системный пакетный файл, который содержит информацию о ключевых переменных среды (напр. команды PATH, PROMPT, SET). В MS-DOS AUTOEXEC.BAT исполняется во время загрузки операционной системы. По информации из файла создается корневая среда. При запуске программы ей передается копия корневой среды.

# ПРИЛОЖЕНИЕ А

# ИСХОДНЫЙ КОД ПРОГРАММЫ. LR2.ASM

```
TESTPC SEGMENT
      ASSUME  CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
      org 100h

START:  JMP BEGIN

; DATA SEGMENT

INACCESSIBLE_MEMORY db             "Inaccessible   memery address:            ",
0dh, 0ah, '$'
ENVIRONMENT_SEGMENT_ADDRESS db     "Environment segment address:            ",
0dh, 0ah, '$'
ENVIRONMENT_DATA db                "Environment data:
            ", 0dh, 0ah, '$'

END_OF_LINE db " ", 0dh, 0ah, '$'

START_PATH db "Start directory: ", 0dh, 0ah, '$'

COMMAND_TEXT db "Command line tail: "
COMMAND_TAIL db "                                                    ",    0dh,
0ah, '$'

; CODE SEGMENT

;-------------------------------------------------------------------------
--
PRINT_STRING PROC near
            push AX
            mov AH, 09h
            int   21h
            pop AX
            ret
PRINT_STRING ENDP
;-------------------------------------------------------------------------
--

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

; байт AL переводится в два символа шестн. числа в AX
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  ; в AL - старшая, в AH - младшая
      pop CX
      ret
BYTE_TO_HEX ENDP

; перевод в 16 с/с 16-ти разрядного числа
; в AX - число, DI - адрес последнего символа
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

; перевод в 10с/с, SI - адрес поля младшей цифры
BYTE_TO_DEC PROC near
      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
      ret
BYTE_TO_DEC ENDP

;-------------------------------------------------------------------------------
--

FUNCTION_NOT_AVAILABLE_MEMORY PROC NEAR
      push AX
      push DI

      mov AX, DS:[02h]
      mov DI, offset INACCESSIBLE_MEMORY
      add DI, 020h
      call WRD_TO_HEX

      mov DX, offset INACCESSIBLE_MEMORY
      call PRINT_STRING

      pop DI
      pop AX
      ret
FUNCTION_NOT_AVAILABLE_MEMORY ENDP

;-------------------------------------------------------------------------------
--

FUNCTION_ENVIRONMENT_SEGMENT_ADDRESS PROC NEAR
      push AX
      push DI

      mov AX, DS:[02Ch]
      mov DI, offset ENVIRONMENT_SEGMENT_ADDRESS
      add DI, 01Fh
      call WRD_TO_HEX

      mov DX, offset ENVIRONMENT_SEGMENT_ADDRESS
      call PRINT_STRING

      pop DI
```

```
	pop AX
	ret
FUNCTION_ENVIRONMENT_SEGMENT_ADDRESS ENDP

;-------------------------------------------------------------------------------
--

FUNCTION_COMMAND_TAIL PROC NEAR
	push AX
	push BX
	push CX
	push DX

	push SI
	push DI

	mov SI, 80h
	xor CX, CX
	mov CL, byte ptr cs:[SI]
	mov BX, offset COMMAND_TAIL

	inc SI
	cycle_begin:
		cmp CL, 0h
		jz cycle_end

		xor AX, AX
		mov AL, byte ptr cs:[SI]
		mov [BX], AL

		add BX, 1
		sub CL, 1
		add SI, 1

		jmp cycle_begin
	cycle_end:

	xor AX, AX
	mov AL, 0Ah
	mov [BX], AL
	inc BX
	mov AL, '$'
	mov [BX], AL

	mov DX, offset COMMAND_TEXT
	call PRINT_STRING
	mov DX, offset END_OF_LINE
```

```
      call PRINT_STRING

      pop DI
      pop SI

      pop DX
      pop CX
      pop BX
      pop AX

      ret
FUNCTION_COMMAND_TAIL ENDP

;-------------------------------------------------------------------------------
--

FUNCTION_ENVIRONMENT_DATA PROC NEAR
      push AX
      push DX
      push DS
      push ES

      mov DX, offset ENVIRONMENT_DATA
      call PRINT_STRING

      mov AH, 02h
      mov ES, DS:[02Ch]
      xor SI,SI

      cycle1_begin:
            mov DL, ES:[SI]
            int 21h
            cmp DL, 0h
            je cycle1_end
            inc SI
            jmp cycle1_begin
      cycle1_end:

      mov DX, offset END_OF_LINE
      call  PRINT_STRING

      inc SI
      mov DL, ES:[SI]
      cmp DL, 0h
      jne cycle1_begin

      mov DX, offset END_OF_LINE
```

```
      call PRINT_STRING

      mov DX, offset START_PATH
      call PRINT_STRING

      add SI, 3h
      mov AH, 02h
      mov ES, DS:[02Ch]

      cycle2_begin:
            mov DL, ES:[SI]
            cmp DL, 0h
            je cycle2_end
            int 21h
            inc SI
            jmp cycle2_begin
      cycle2_end:

      pop ES
      pop DS
      pop DX
      pop AX
      ret
FUNCTION_ENVIRONMENT_DATA ENDP

;-------------------------------------------------------------------------------
--

begin:

      call FUNCTION_NOT_AVAILABLE_MEMORY
      call FUNCTION_ENVIRONMENT_SEGMENT_ADDRESS
      call FUNCTION_COMMAND_TAIL
      call FUNCTION_ENVIRONMENT_DATA

      xor AL, AL
      mov AH, 4Ch
      int 21h

TESTPC      ENDS
            END START

; find me https://github.com/Nik-Poch
```