МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Операционные системы»

Тема: Исследование интерфейсов загрузочных модулей

Студентка гр. 0382 _____ Деткова А.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2022

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.

Написать и отладить программный модуль типа .COM, который выбирает и распечатывает следующую информацию:

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

Выполнение работы.

В ходе работы разработана программа lab2.asm с исходным кодом.

Все сегментные регистры вначале работы указывают на начало PSP, поэтому для получения информации использовались сегментные регистры ES и DS. Адресация вида: ES/DS:[<смещение до адреса>].

Чтобы получить сегментный адрес недоступной памяти нужно рассмотреть два байта по смещению 2H в блоке памяти PSP. Сегментный адрес рассматривает и выводит на экран процедура print_seg_adr_of_first. Для этого по смещению ES:[0002H] берутся два байта в регистр AX, слово переводится в 16 сс с помощью процедуры wrd_to_hex. Результат помещается в строку SegAddrOfFirst и выводится на экран.

Для получения сегментного адреса среды используется процедура print_seg_addr_of_env. Аналогично прошлой процедуре берутся два байта по смещению 002CH от начала PSP и аналогично преобразуются и выводятся в строке SegAddrOfEnv.

Хвост командной строки выводится с помощью процедуры print_tail_cmd_str. Длина хвоста находится по адресу ES:[0080H], сохраняется в CX. Со смещения 0081H находится сам хвост, по хвосту проходим посимвольно (побайтово), пока CX не равен 0. Символы выводятся на экран с помощью функции 02H прерывания 21H.

Содержимое области среды и путь загружаемого модуля выводятся с помощью процедуры print_env_and_path. Сначала определяется сегментный адрес среды, а потом начинается считывание, каждая строка оканчивается двумя нулями, в регистре AX — два символа, AL — печатается, потом AH переносится в AL, и AH считывается, считывание происходит до тех пор, пока весь AX не станет равен 0 (AH = 00H, AL = 00H). После вывода области среды, выводится путь к файлу, это делается аналогично хвосту.

Рисунок 1: Результат выполнения (хвост пуст)

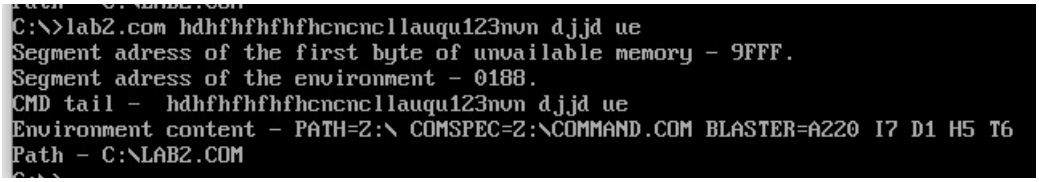

Рисунок 2: Результат выполнения (хвост не пуст)

Ответы на контрольные вопросы.

Сегментный адрес недоступной памяти:

1. На какую область памяти указывает адрес недоступной памяти?

   На конец сегмента обычной памяти.

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

   По адресу от начала программы со смещением 2Н.

3. Можно ли в эту область памяти писать?

   Да, можно.

Среда передаваемая программе:

1. Что такое среда?

(Wiki) Среда окружения (англ. Environment) — в информатике совокупность значений системных переменных, путей, открытых файловых дескрипторов и других ресурсов операционной системы, передаваемые процессу (программе) при его запуске. Кратко, среда окружения — совокупность переменных, в которых хранятся настройки ОС,

Хранится в памяти в виде символьных строк вида: имя=параметр.

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

Среда создается при запуске ОС. При загрузке модуля в оперативную память среда копируется в адресное пространство запускающей программы.

3. Откуда берется информация, записываемая в среду?

   Из пакетного файла autoexec.bat.

Выводы.

В ходе работы были изучены принципы устройства PSP и состав некоторых ее частей. Была написана программа на языке ассемблера, выводящая информацию из PSP.

ПРИЛОЖЕНИЕ А

КОД МОДУЛЕЙ

Название файла: lab2.asm

```
MainSeg SEGMENT
    ASSUME CS:MainSeg, DS:MainSeg, ES:NOTHING, SS:NOTHING
    ORG 100H

start:
    jmp begin

data:
     SegAddrOfFirst db 'Segment adress of the first byte of unvailable
memory -     .', 0DH, 0AH, '$'
     SegAddrOfEnv db 'Segment adress of the environment -      .', 0DH,
0AH, '$'
    CMDTail db 'CMD tail - ', '$'
    CMDTail_empty db 'CMD tail is empty.', 0DH, 0AH, '$'
    Env db 'Environment content - ', '$'
    Path db 'Path - ', '$'

begin:
    call main
    xor AL,AL
    mov AH,4CH
    int 21H

_print PROC NEAR

    push AX

    mov AH, 09H
    int 21H

    pop AX

    ret

_print ENDP

byte_to_dec PROC NEAR

    ; AH - number, SI - adress of last symbol

    push CX
    push DX
    push AX

    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 AX
    pop DX
    pop CX

    ret

byte_to_dec ENDP

tetr_to_hex PROC NEAR

    and AL,0FH    ; save only last part of byte
    cmp AL,09
    jbe next
    add AL,07
  next:
    add AL,30H
    ret

tetr_to_hex ENDP

byte_to_hex PROC NEAR

    ; AL - number -> 2 symbols in 16 numb. syst. in AX

    push CX

    mov AH,AL    ; save AL
    call tetr_to_hex
    xchg AL,AH
    mov CL,4
    shr AL,CL
    call tetr_to_hex    ; AL - high numb ascii, AH - low numb ascii

    pop CX
    ret

byte_to_hex ENDP
```

```
wrd_to_hex PROC NEAR

    ; AX - number, DI - last symbol adress

    push BX
    push AX

    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 AX
    pop BX
    ret

wrd_to_hex ENDP

print_seg_adr_of_first PROC NEAR

    mov AX,ES:[0002H]
    mov DI,offset SegAddrOfFirst + 59
    call wrd_to_hex
    mov DX,offset SegAddrOfFirst
    call _print

    ret

print_seg_adr_of_first ENDP


print_seg_addr_of_env PROC NEAR

    mov AX,ES:[002CH]
    mov DI,offset SegAddrOfEnv + 39
    call wrd_to_hex
    mov DX,offset SegAddrOfEnv
    call _print

    ret

print_seg_addr_of_env ENDP


print_tail_cmd_str PROC NEAR

    xor CX,CX
    mov CL,ES:[0080H]
```

```
    cmp CL,0H
    je _empty_tail

    mov SI,0081H

    mov DX,offset CMDTail
    call _print

  _tail:
    mov DL,ES:[SI]
    call print_symbol
    inc SI
    loop _tail

    mov DL,0DH
    call print_symbol
    mov DL,0AH
    call print_symbol

    ret

  _empty_tail:
    mov DX,offset CMDTail_empty
    call _print

    ret

print_tail_cmd_str ENDP


print_symbol PROC NEAR

   push AX

   mov AH,02H
   int 21h

   pop AX

   ret

print_symbol ENDP


print_env_and_path PROC NEAR

    mov DX,offset Env
    call _print

    mov ES,DS:[002CH]
    xor DI,DI
    mov AX,ES:[DI]

    cmp AX,00H
```

```
    jz _fin

    add DI,2

  _read_symb:
    mov DL,AL
    call print_symbol
    mov AL,AH
    mov AH,ES:[DI]
    inc DI
    cmp AX,00H
    jne _read_symb

  _fin:
    mov DL,0DH
    call print_symbol
    mov DL,0AH
    call print_symbol

    mov DX,offset Path
    call _print

    add DI,2
    mov DL,ES:[DI]
    inc DI
  _path_read:
    call print_symbol
    mov DL,ES:[DI]
    inc DI
    cmp DL,00H
    jne _path_read

    ret

print_env_and_path ENDP


main PROC NEAR

    call print_seg_adr_of_first
    call print_seg_addr_of_env
    call print_tail_cmd_str
    call print_env_and_path
    ret

main ENDP

MainSeg ENDS
END start
```