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 программных модулей

Студент гр. 0382 _____ Кондратов Ю.А.

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 состоит префикс сегмента программы (PSP) и помещает его адрес в сегментные регистр. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

Задание.

Требуется 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 стоки в символьном виде.
4. Содержимое области среды в символьном виде.
5. Путь загружаемого модуля.

Выполнение работы.

Для выполнения поставленных задач был написан файл и исходным кодом main.asm.

Так как сегментный регистр DS указывает на начало PSP, то для получения информации из префикса можно использовать адресацию вида ds:[<смещение поля>].

Для получения информации об адресе недоступной памяти происходит запись значения по адресу ds:[2h] в регистр ax, далее вызывается функция wrd_to_hex, которая переводит шестнадцатеричное число в двух байтах в его строковое представление. Далее полученная информация выводится на консоль.

Для получения информации об адресе недоступной памяти происходит запись значения по адресу ds:[2Ch] в регистр ax, далее вызывается функция wrd_to_hex. Полученная информация выводится на консоль.

Для вывода на консоль хвоста командной строки сначала происходит запись в регистр CL длины хвоста командной строки при помощи обращения к адресу ds:[80h]. Если длина хвоста командной строки равна нулю, ты выводится сообщение о том, что хвост пуст. Иначе производится считывание хвоста. Считывание производится по одному символу, каждый считанный символ выводится на консоль при помощи функции print_symbol.

Для вывода на консоль содержимого области среды в символьном виде производится также посимвольно, но так как признаком конца являются два нулевых байта, то используется очередь длины 2 байта, в качестве которой используется регистр AX. Считывание производится в цикле, на каждой итерации которого сначала выводится на консоль символ, содержащийся в AL, далее в AL помещается символ из AH, а в AH записывается новый символ. Если содержимое регистра AX равно 0000h, то считывание заканчивается. Далее пропускается два байта и начинается считывание пути к исполняемому модулю. Оно производится аналогично считыванию хвоста командной строки.

Результат работы программы при пустом и непустом хвосте командной строки представлен на рисунке 1 и 2 соответственно.

```
C:\>MAIN.COM
Adress of not available memory: 9FFFh
Adress of enviroment: 0188h
CMD tail is empty.
Enviroment content: PATH=Z:\ COMSPEC=Z:\COMMAND.COM BLASTER=A220 I7 D1 H5 T6
Executable module path: C:\MAIN.COM
```

Рисунок 1 – Результат выполнения программы с пустым хвостом командной

```
C:\>MAIN.COM lakdjf91i3jl.a,mdf 91q8ijl ,.mz9o8cuijq kl4r
Adress of not available memory: 9FFFh
Adress of enviroment: 0188h
CMD tail is:  lakdjf91i3jl.a,mdf 91q8ijl ,.mz9o8cuijq kl4r
Enviroment content: PATH=Z:\ COMSPEC=Z:\COMMAND.COM BLASTER=A220 I7 D1 H5 T6
Executable module path: C:\MAIN.COM
```

Рисунок 2 – Результат выполнения программы при непустом хвосте командной строки

Контрольные вопросы.

Сегментный адрес недоступной памяти:

1. На какую область памяти указывает адрес недоступной памяти?

   На сегмент, расположенный сразу после памяти, выделенной программе.

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

   Этот адрес расположение в PSP по смещению 2h, то есть в том же сегменте, в котором расположена программ по смещению 2h.

3. Можно ли в эту область памяти писать?

   Да, можно. В MS DOS не предусмотрена защита от перезаписи.

Среда, передаваемая программе:

1. Что такое среда?

   Среда – это область памяти, в которой в виде символьных строк записаны переменные среды, хранящие какую-либо информацию, в формате имя=значение.

2. Когда создается среда?

   При запуске ОС. При запуске программы происходит копирование в ее адресное пространство среду запускающей программы.

3. Откуда берется информация, записываемая в среду?

   Из файла ATUOEXEC.COM, который исполняется после старта командного интерпретатора COMMAND.COM

Выводы.

В ходе работы были изучены основные принципы устройства PSP и некоторые из его полей: информация, хранящаяся в них и её назначение. Была написана программа на языке Ассемблера, выводящая на экран информацию, требуемую в задании.

ПРИЛОЖЕНИЕ А.

Исходный код модулей

main.asm:

```
main_seg SEGMENT
    ASSUME CS:main_seg, DS:main_seg, ES:NOTHING, SS:NOTHING
    ORG 100h

start:
    jmp begin

data:
    NMEM_ADDR db "Adress of not available memory:    h",0Dh, 0Ah,"$"
    ENV_ADDR db "Adress of enviroment:     h",0Dh,0Ah,"$"
    EMPTY_TAIL_MSG db "CMD tail is empty.",0Dh,0Ah,"$"
    TAIL_MSG db "CMD tail is: ", "$"
    ENV_MSG db "Enviroment content: ","$"
    PATH_MSG db "Executable module path: ","$"

begin:
    call main
    xor al, al
    mov ah, 4Ch
    int 21h

print_nmem_addr PROC NEAR
    mov ax, ds:[2h]
    mov di, OFFSET NMEM_ADDR + 35
    call wrd_to_hex
    mov dx, OFFSET NMEM_ADDR
    call print
    ret
print_nmem_addr ENDP

print_env_addr PROC NEAR
    mov ax, ds:[2Ch]
    mov di, OFFSET ENV_ADDR + 25
    call wrd_to_hex
    mov dx, OFFSET ENV_ADDR
    call print
    ret
print_env_addr ENDP

print_symbol PROC NEAR
    push ax
    mov ah, 02h
    int 21h
    pop ax
    ret
print_symbol ENDP

print_cmd_tail PROC NEAR
    mov cl, ds:[80h]
    cmp cl, 0h
    je empty_tail
```

```
    mov dx, OFFSET TAIL_MSG
    call print
    mov si, 81h
loop_tail:
    mov dl, ds:[si]
    call print_symbol
    inc si
    loop loop_tail
    mov dl, 0Dh
    call print_symbol
    mov dl, 0Ah
    call print_symbol
    ret
empty_tail:
    mov dx, OFFSET EMPTY_TAIL_MSG
    call print
    ret
print_cmd_tail ENDP

print_env PROC NEAR
    mov dx, OFFSET ENV_MSG
    call print
    mov es, ds:[2Ch]
    xor di, di
    mov ax, es:[di]
    cmp ax, 00h
    jz loop_fin
    add di, 2
read_loop:
    mov dl, al
    call print_symbol
    mov al, ah
    mov ah, es:[di]
    inc di
    cmp ax, 00h
    jne read_loop
loop_fin:
    mov dl, 0Dh
    call print_symbol
    mov dl, 0Ah
    call print_symbol
    mov dx, OFFSET PATH_MSG
    call print
    add di, 2
    mov dl, es:[di]
    inc di
path_loop:
    call print_symbol
    mov dl, es:[di]
    inc di
    cmp dl, 00h
    jne path_loop
    ret
print_env ENDP

print PROC NEAR
    mov ah, 09h
    int 21h
```

```
    ret
print 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

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 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

main PROC NEAR
    call print_nmem_addr
    call print_env_addr
    call print_cmd_tail
    call print_env
    ret
main ENDP

main_seg ENDS
END start
```