Министерство образования Республики Беларусь

Учреждение Образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Лабораторная работа №5

по дисциплине «Программирование на языке ассемблера»

на тему «Работа с файлами»

вариант №8

|  |  |  |
| --- | --- | --- |
| Выполнил:  студент гр. 250504  Солодков М. Д |  | Проверил:  Туровец Н.О. |

Минск 2023

**Цель работы:** Ознакомиться с основными операциями обработки файлов, получить понятие о работе с параметрами командной строки.

**Теоретические сведения**

Для выполнения работы требуется рассмотреть следующие элементы языка ассемблера и операционной системы:

1. *Работа с файлами.*

Для работы с файлами в данной лабораторной работе лучше всего ис-пользовать функции DOS, которые обращаются к файлу через 16-битный иден-тификатор (дескриптор) файла. Такой подход более прост, чем использование более старых описателей файла (37-байтного блока управления файлом FCB) или функций низкого уровня доступа к диску (прерывание BIOS 13h).

Первые пять значений идентификаторов такого формата инициализиру-ются системой следующим образом:

-- 0 – STDIN – стандартное устройство ввода (клавиатура),

-- 1 – STDOUT – стандартное устройство вывода (экран),

-- 2 – STDERR – устройство вывода сообщений об ошибках (всегда экран),

-- 3 – AUX – последовательный порт (СОМ1),

-- 4 – PRN – параллельный порт (LPT1).

Работа с файлами выполняется через функции DOS в стандартном поряд-ке:

1) создание или открытие существующего файла;

2) выполнение файловых операций чтения или записи данных;

3) закрытие файла.

Дополнительно доступны операции: удаление, поиск и управление.

2. *Работа с командной строкой.*

Передача параметров программы через командную строку при запуске программы – решение, которое позволяет построить гибкую по входным дан-ным программу без дополнительных диалогов с пользователем.

При запуске программы DOS помещает всю командную строку (включая последний символ 0Dh) в блок PSP запущенной программы по смещению 81h и ее длину в байт 80h. Длина командной строки, хранящейся в PSP, не может быть больше 126 символов (командная строка большей длины доступна начи-ная с DOS 4.0 в переменной среды CMDLINE).

При загрузке программы, в начале отводимого для нее блока памяти, со-здается структура данных PSP (префикс программного сегмента) размером 256 байт (100h). Затем DOS создает копию текущего окружения для загружаемой программы, помещает полный путь и имя программы в конец окружения, за-полняет поля PSP, сама программа записывается в память, начиная с адреса PSP:0100h.

При запуске *com*-программы регистры устанавливаются следующим об-разом:

1) AL = FFh, если первый параметр командной строки содержит непра-вильное имя диска (например, z:/something), иначе – AL = 00h.

2) АН = FFh, если второй параметр содержит неправильное имя диска, иначе АН = 00h.

3) CS = DS = ES = SS = сегментный адрес PSP.

4) SP = адрес последнего слова в сегменте (обычно FFFEh или меньше, ес-ли не хватает памяти).

При запуске *exe*-программы регистры SS:SP устанавливаются в соответ-ствии с сегментом стека, определенным в программе.

Затем в стек помещается слово 0000h и выполняется переход на начало программы (PSP:0100h для *com*, собственная точка входа для *exe*).

Описание полей блока PSP приведено в таблице 1.

Таблица 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Смещение, байт** | **Размер, байт** | **Имя поля** | **Описание** |
| 0 | 2 | int20h | Двоичный код команды INT 20h (программы могут использовать эту команду для завершения своей работы) |
| 2 | 2 | mem\_top | Нижняя граница доступной памяти в системе в параграфах |
| 4 | 1 | reserv1 | Зарезервировано |
| 5 | 5 | call\_dsp | Команда CALL межсегментного вызова диспетчера MS-DOS |
| 10 | 4 | int 22h | Адрес завершения (Terminate Address) |
| 14 | 4 | int 23h | Адрес обработчика прерывания, который получает управление, если пользователь нажал комбинацию клавиш <Ctrl+Break> |
| 18 | 4 | int 24h | Адрес обработчика критической ошибки |
| 22 | 2 | parn\_psp | Сегмент PSP программы, запустившей данную программу (программы-родителя) |
| 24 | 20 | file\_tab | Таблица открытых файлов; если в этом поле находятся байты 0FFH, то таблица не используется |
| 44 | 2 | env\_seg | Сегмент блока памяти, содержащего переменные среды |
| 46 | 4 | ss\_sp | Адрес стека программы SS:SP |
| 50 | 2 | max\_open | Максимальное количество открытых файлов |
| 52 | 4 | file\_tba | Адрес таблицы открытых файлов |
| 56 | 24 | reserv2 | Зарезервировано |
| 80 | 3 | disp | Диспетчер функций MS-DOS |
| 83 | 9 | reserv3 | Зарезервировано |
| 92 | 16 | fcb1 | Если первый аргумент командной строки содержит правильное имя файла, поле форматируется как стандартный блок FCB |
| 108 | 20 | fcb2 | Заполняется для второго аргумента командной строки аналогично fcb1 |
| 128 | 1 | p\_size | Количество значащих символов в неформатированной области параметров, либо буфер обмена с диском DTA , назначенный по умолчанию |
| 129 | 127 | parm | Неформатированная область параметров, заполняется при запуске программы из командной строки |

**Код программы (.exe)**

.8086

.model small

.stack 100h

.data

message\_arguments\_error db "Wrong arguments. Try: laba5 file\_name.txt$"

message\_file\_missing db "Can't open file. Probably it's missing$"

message\_success db "Amount of strings with your word is $"

message\_input\_word db "This program counts strings with your word. Please enter the word:$"

word\_to\_find db 128 dup('$')

word\_buffer db 128 dup('$')

counter dw 0

file\_name db 128 dup(0)

file\_buffer db 1024 dup('$')

file\_buffer\_size equ 1024

file\_descriptor dw 0

number\_buffer db 9 dup("$")

crlf db 0Dh, 0Ah, '$'

.code

input\_string PROC

push bp

mov bp, sp

mov ah, 0Ah

mov dx, [bp + 4]

int 21h

mov bx, offset dx

mov ax, [bx + 1]

xor ah, ah

add bx, 2

add bx, ax

mov al, '$'

mov [bx], al

pop bp

ret 2

input\_string ENDP

output\_string PROC

push bp

mov bp, sp

mov ah, 09h

mov dx, [bp + 4]

int 21h

pop bp

ret 2

output\_string ENDP

output\_number PROC

push bp

mov bp, sp ;[bp + 4] bx

;[bp + 6] number\_buffer

mov ax, [bp + 4]

mov di, [bp + 6]

xor bx, bx

mov bx, 10

xor cx, cx

convert\_loop:

xor dx, dx

div bx

add dl, 30h

push dx

inc cx

cmp ax, 0

je final

jmp convert\_loop

final:

xor bx, bx

fill\_buffer:

pop si

add di, bx

mov [di], si

sub di, bx

inc bx

loop fill\_buffer

mov ah, 09h

mov dx, [bp + 6]

int 21h

pop bp

ret 4

output\_number ENDP

open\_file\_read PROC

push bp

mov bp, sp ;[bp + 4] file\_name

mov dx, [bp + 4]

mov ah, 3Dh

mov al, 00h

int 21h

jc missing\_file

pop bp

ret 2

missing\_file:

;mov ah, 3Eh

;int 21h

lea dx, message\_file\_missing

push dx

call output\_string

jmp exit

open\_file\_read ENDP

close\_file PROC

push bp

mov bp, sp ;[bp + 4] file\_descriptor

mov bx, [bp + 4]

mov ah, 3Eh

int 21h

pop bp

ret 2

close\_file ENDP

read\_from\_file PROC

push bp

mov bp, sp ;[bp + 4] file\_descriptor

mov bx, [bp + 4]

mov cx, file\_buffer\_size

lea dx, file\_buffer

mov ah, 3Fh

int 21h

pop bp

ret 2

read\_from\_file ENDP

strlen PROC ;bx = res

push bp

mov bp, sp ;[bp + 4] word

push si

push ax

mov si, [bp + 4]

mov bx, 0 ;i = 0

count\_length:

mov al, [si + bx]

cmp al, '$'

je return\_count

inc bx

jmp count\_length

return\_count:

pop ax

pop si

pop bp

ret 2

strlen ENDP

strcmp PROC ;true = 1, false = 0

push bp

mov bp, sp ;[bp + 4] str1

;[bp + 6] str2

push si ;сохраняем

push di

push bx

push ax

push cx

mov si, [bp + 4]

mov di, [bp + 6]

push si

call strlen

push bx

push di

call strlen

mov ax, bx

pop bx

cmp ax, bx

jne false

mov bx, 0

mov cx, ax

compare:

;xor dh, dh

mov dl, [di]

mov al, [si]

cmp dl, al

jne false

inc si

inc di

inc bx

cmp bx, cx

je true

jmp compare

true:

mov dx, 1

pop cx

pop ax

pop bx

pop di

pop si

pop bp

ret 4

false:

mov dx, 0

pop cx

pop ax

pop bx

pop di

pop si

pop bp

ret 4

strcmp ENDP

count\_strings PROC

push bp

mov bp, sp ;[bp + 4] file\_buffer

;[bp + 6] word\_to\_find

;[bp + 8] word\_buffer

mov si, [bp + 4]

mov di, [bp + 6]

push si

call strlen

mov cx, bx ;strlen

mov ax, 0 ;i

mov dx, 0 ;j

mov bx, [bp + 8]

main\_cycle:

push ax

xor ah, ah

mov al, [si]

cmp al, ' '

je next1

cmp al, 0Dh

je next1

cmp al, 0Ah

je next1

cmp al, '$'

je next1

next2:

mov al, [si]

mov [bx], al

pop ax

inc ax

inc si

inc dx

inc bx

cmp ax, cx

jbe main\_cycle

next1:

mov al, '$'

mov [bx], al

mov bx, [bp + 8]

mov di, [bp + 6]

push di

push bx

call strcmp

cmp dx, 01h

jne skip\_inc\_counter

inc counter

check\_loop:

mov al, 0Dh

cmp [si], al

je skip\_inc\_counter

mov al, '$'

cmp [si], al

je skip\_inc\_counter

inc si

jmp check\_loop

skip\_inc\_counter:

mov dx, 0

mov bx, [bp + 8]

mov al, '$'

mov [bx], al

pop ax

inc ax

inc si

cmp ax, cx

jbe main\_cycle

pop bp

ret 6

count\_strings ENDP

start:

mov ax, @data

mov ds, ax

xor bx, bx

mov bl, es:[80h] ;CLA length

add bx, 80h ;last symbol

mov si, 82h ;first symbol

lea di, file\_name

parse\_path:

cmp BYTE PTR es:[si], ' '

je parsed\_path

cmp BYTE PTR es:[si], 0Dh

je parsed\_path

mov al, es:[si]

mov [di], al

inc di

inc si

cmp si, bx

jbe parse\_path

parsed\_path:

cmp si, bx

jb wrong\_arguments

lea dx, file\_name

push dx

call open\_file\_read

mov file\_descriptor, ax

push file\_descriptor

call read\_from\_file

push file\_descriptor

call close\_file

lea dx, message\_input\_word

push dx

call output\_string

lea dx, word\_to\_find

push dx

call input\_string

lea dx, crlf

push dx

call output\_string

lea dx, word\_buffer

push dx

lea dx, word\_to\_find

add dx, 2

push dx

lea dx, file\_buffer

push dx

call count\_strings

lea dx, message\_success

push dx

call output\_string

lea dx, number\_buffer

push dx

push counter

call output\_number

jmp exit

wrong\_arguments:

lea dx, message\_arguments\_error

push dx

call output\_string

jmp exit

exit:

mov ax, 4C00h

int 21h

end start

**Вывод программы**

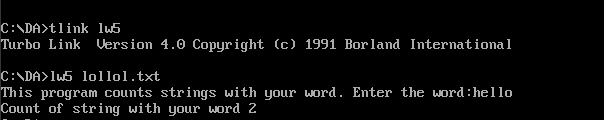


Рисунок 1 – Результат работы программы