Лабораторная работа 6. Файлы

Язык ассемблера не содержит средств для работы с файлами. Если такая необходимость возникает, то программа должна содержать фрагменты кода, в которых производится обращение к средствам операционной системы, осуществляющим взаимодействие с файловой системой.

Работа с файлами в MS DOS (имена 8.3)

В основе файловой системы MS DOS лежит древовидная структура каталогов. Корень этой структуры представляет собой совокупность ограниченного числа дескрипторов, описывающих файлы и каталоги (подкаталоги) следующего уровня. Подкаталог представляет собой файл особого типа, который содержит дескрипторы файлов и подкаталогов очередного нижележащего уровня. В отличие от корневого каталога количество дескрипторов в подкаталоге не ограничено и определяется только размером диска. Дескриптор представляет собой экземпляр структуры размером 32 байта. Поля этой структуры содержат различную информацию о файле: идентификатор файла и его характеристики — дата и время создания (модификации), номер начального кластера, длина файла и его атрибуты.

Для использования файла в программе необходимо выполнить следующие операции:

- создание нового файла;
- открытие существующего файла;
- запись/чтение в/из файл(а);
- закрытие файла.

Создание нового файла

Прежде, чем что-то записывать, необходимо создать файл. Для этого используется функция DOS 3Ch.

Вход:	AH	3Ch	
	CX	атрибут файла:	
		Бит 7:	файл можно открывать разным процессам в Novell Netware
		Бит 6:	не используется
		Бит 5:	архивный бит (1, если файл не сохранялся)
		Бит 4:	директория (должен быть 0 для функции 3Ch)
		Бит 3:	метка тома (игнорируется функцией 3Ch)
		Бит 2:	системный файл
		Бит 1:	скрытый файл
		Бит 0:	файл только для чтения
	DS:DX	адрес ASCIZ-строки с полным именем файла (ASCIZ - это строка ASCII-символов, оканчивающаяся нулем)	
Выход:	CF=0	и AX = идентификатор файла, если не произошла ошибка	
	CF=1	и AX = 3, если путь не найден	
	CF=1	и AX = 4, если слишком много открытых файлов	
	CF=1	и AX = 5, если доступ запрещен	

Имя файла должно быть в формате 8.3 — 8 символов имени и 3 символа расширения. Естественно, можно использовать только английский буквы, цифры и некоторые другие символы. Строка с именем файла должна заканчиваться нулевым байтом. Если файл уже существует, то его содержимое будет удалено.

Об ошибке можно узнать, проверяя значение флага CF (1 — ошибка, 0 — нет ошибки). Аналогично для других функций работы с файлами. Если флаг CF равен 0, то в регистре AX будет находиться дескриптор (или описатель) файла. Дескриптор — это просто специальное число, по которому операционная система отличает один открытый файл от другого.

Пример:

```
use16
org 100h

jmp start
;------- Объявление переменных ----------
    file1 db 'file1.txt',0
;--------------------------------------

start:
    mov ah, 3ch       ; функция создания файла
    mov dx, file1     ; указываем адрес строки с именем файла
    xor cx, cx        ; CX=0 - обычный файл, без атрибутов
    int 21h
int 20h
```

Запись данных в файл

Запись в файл выполняется функцией DOS 40h. Этой функции нужно передать в регистре BX тот самый дескриптор, который был получен при создании файла.

Вход:	AH	40h
	BX	1 для STDOUT или 2 для STDERR
	DS:DX	адрес начала строки
	CX	длина строки (число записываемых байт)
Выход:	CF	0
	AX	число записанных байтов (или код ошибки, если CF=1)

Эта функция предназначена для записи в файл, но, если в регистр BX поместить число 1, она будет выводить данные на STDOUT, а если BX = 2, то на устройство STDERR, которое всегда выводит данные на экран и не перенаправляется в файлы.

Нужно всегда сравнивать возвращаемое значение AX (число записанных байтов) с CX (запрошенное число байт для записи)

- если AX = CX, то запись была успешной
- если AX < CX, то случилась ошибка (скорее всего переполнение)

Эта функция может выводить на экран символ $.

Пример:

```
use16
org 100h

jmp start
;------- Определяем переменные ----------
    stroka db 'symbol $'
;--------------------------------------
```

```
start:
    mov ah, 40h      ; AH = 40h (номер функции DOS)
    mov bx, 1        ; пишем в STDOUT
    mov dx, stroka   ; адрес записываемой строки
    mov cx, 8        ; длина строки, включая символ $
    int 21h
int 20h
```

Закрытие файла

После работы с файлом нужно его закрыть с помощью функции DOS 3Eh.

Вход:	AH	3Eh
	BX	идентификатор файла
Выход:	CF=0	если не произошла ошибка
	CF=1	и AX = 6, если неправильный идентификатор

Если вы не закроете файл, то он будет закрыт системой при выходе из программы.

Если файл был открыт для записи, то все файловые буфера сбрасываются на диск, устанавливается время изменения файла и записывается его новая длина.

Пример. Ввести строку с клавиатуры и записать её в файл.

```
use16
org 100h

jmp start
;------- Определяем переменные ----------
    fileName db 'input.txt',0 ;
    s1 rb 52      ; макс. длина строки
    size db 50    ; размер буфера
    handle rw 1   ; дескриптор файла
;----------------------------------------

start:
    mov dx, s1   ;адрес строки
    mov [s1], 50 ;1-й байт - макс. количество символов в строке
    mov ah, 0ah  ;функция DOS для ввода строки
    int 21h

    mov ah, 3ch      ;создаем файл
    mov dx, fileName ;имя файла
    xor cx, cx       ;нет атрибутов - обычный файл
    int 21h          ;обращение к функции DOS
    jnc m1           ;если нет ошибки, то продолжаем
    jmp exit         ;иначе выход из программы
m1: mov [handle], ax ;сохранение дескриптора файла

    mov bx, ax       ;дескриптор файла
    mov ah, 40h      ;Функция DOS 40h (запись в файл)
    mov dx, s1       ;адрес строки с данными
    add dx, 2        ;пропускаем первые 2 байта
```

```
    movzx cx, [size] ;размер данных
    int 21h          ;обращение к функции DOS
    jnc close_file   ;Если нет ошибки, то закрыть файл
    jmp exit         ;иначе выход из программы

close_file:
    mov ah, 3eh      ;Функция DOS 3Eh (закрытие файла)
    mov bx, [handle] ;дескриптор
    int 21h          ;обращение к функции DOS

exit:
int 20h
```

Открытие существующего файла

Для открытия файла используется функция DOS 3Dh. В отличие от создания файла, эта функция завершится ошибкой, если файл не существует.

<table>
<tr><td rowspan="15">Вход:</td><td>AH</td><td colspan="2">3Dh</td></tr>
<tr><td rowspan="14">AL</td><td colspan="2">режим доступа:</td></tr>
<tr><td>Бит 0:</td><td>открыть для чтения</td></tr>
<tr><td>Бит 1:</td><td>открыть для записи</td></tr>
<tr><td>Бит 2:</td><td>зарезервирован (0)</td></tr>
<tr><td>Бит 3:</td><td>зарезервирован (0)</td></tr>
<tr><td rowspan="6">Бит 6-4:</td><td>режим доступа для других процессов:</td></tr>
<tr><td>000: режим совместимости (остальные процессы также должны открывать этот файл в режиме совместимости)</td></tr>
<tr><td>001: все операции запрещены</td></tr>
<tr><td>010: запись запрещена</td></tr>
<tr><td>011: чтение запрещено</td></tr>
<tr><td>100: запрещений нет</td></tr>
<tr><td>Бит 7:</td><td>файл не наследуется порождаемыми процессами</td></tr>
<tr><td>DS:DX</td><td colspan="2">адрес ASCIZ-строки с полным именем файла</td></tr>
<tr><td>CL</td><td colspan="2">маска атрибутов файла</td></tr>
<tr><td rowspan="6">Выход:</td><td>CF=0</td><td colspan="2">и AX = идентификатор файла, если не произошла ошибка</td></tr>
<tr><td>CF=1</td><td colspan="2">и AX = 2, если файл не найден</td></tr>
<tr><td>CF=1</td><td colspan="2">и AX = 3, если путь не найден</td></tr>
<tr><td>CF=1</td><td colspan="2">и AX = 4, если слишком много открытых файлов</td></tr>
<tr><td>CF=1</td><td colspan="2">и AX = 5, если доступ запрещен</td></tr>
<tr><td>CF=1</td><td colspan="2">и AX = 0Ch, неправильный режим доступа</td></tr>
</table>

DS:DX указывает на строку, в которой прописан полный путь к файлу. Если диск и/или путь не указан, они принимаются по умолчанию.

Файл должен существовать.

Чтение данных из файла

Чтение из файла выполняется функцией DOS 3Fh.

Вход:	AH	3Fh
	BX	идентификатор файла
	CX	число байтов
	DS:DX	адрес буфера для приема данных
Выход:	CF=0	и AX = число считанных байтов, если не произошла ошибка
	CF=1	и AX = 05h, если доступ запрещен
	CF=1	и AX = 6, если неправильный идентификатор

Если при чтении из файла число фактически считанных байтов в AX меньше, чем заказанное число в CX, то был достигнут конец файла. Каждая следующая операция чтения, так же как и записи, начинается не с начала файла, а с того байта, на котором остановилась предыдущая операция чтения/записи. Если требуется считать (или записать) произвольный участок файла, используют функцию 42h.

Необходимо всегда сравнивать возвращаемое значение AX (число прочитанных байтов) с CX (запрошенное число байтов)

- если AX = CX (и CF = 0) - чтение было корректным и без ошибок
- если AX = 0 - достигнут конец файла (EOF)
- если AX < CX (но не нулевой), то:
  - при чтении с устройства - входная строка имеет длину AX байт
  - при чтении из файла - в процессе чтения достигнут EOF

Если вы читаете с устройства, то AX возвращает длину считанной строки с учетом завершающего возврата каретки CR (ASCII 0Dh).

Пример. Открыть файл для чтения и вывести из него строку:

```
use16
org 100h

jmp start
;------- Определяем переменные ----------
    fileName db 'input.txt',0 ;
    size db 50    ; размер буфера
    stroka rb 51    ; размер строки 50 + 1 для $
    handle rw 1   ; дескриптор файла
;--------------------------------------

start:
    mov ah, 3dh      ;Функция DOS 3Dh (открытие файла)
    xor al, al       ;Режим открытия - только чтение
    mov dx, fileName ;имя файла
    xor cx, cx       ;нет атрибутов - обычный файл
    int 21h          ;обращение к функции DOS
    jnc m1           ;если нет ошибки, то продолжаем
    jmp exit         ;иначе выход из программы
m1: mov [handle], ax ;сохранение дескриптора файла

    mov bx, ax       ;дескриптор файла
    mov ah, 3fh      ;Функция DOS 3fh (чтение из файла)
    mov dx, stroka   ;адрес буфера для данных
    mov cl, byte [size]   ;макс. кол-во читаемых данных
    int 21h          ;обращение к функции DOS
    jnc m2           ;Если нет ошибки, то работаем дальше
```

```
    jmp close_file   ;иначе закрываем файл

m2:
    mov bx, stroka
    add bx, ax          ;В AX количество прочитанных байтов
    mov byte [bx], '$' ;Добавление символа '$'

    mov ah, 9
    mov dx, stroka      ;вывод считанной строки
    int 21h

close_file:
    mov ah, 3eh       ;Функция DOS 3Eh (закрытие файла)
    mov bx, [handle] ;дескриптор
    int 21h           ;обращение к функции DOS

exit:
mov ah, 8
int 21h
int 20h
```

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Выбрать вариант в соответствии со списком. Входные данные записать в файл INPUT.TXT. Открыть этот файл для чтения, считать из него строку и, выполнив необходимые преобразования, записать результат в файл OUTPUT.TXT. При решении можно использовать функции и макросы.

ВАРИАНТЫ

1. Сколько раз в веденной строке встречается число 10?
2. Повторить все символы строки заданное количество раз.
3. Подсчитать в строке число букв A и B, если букв A больше, чем B, то удалить в строке все символы B.
4. Подсчитать количество слов в данном предложении.
5. Определить, сколько раз в строке встречается данный символ.
6. Дана символьная строка, заканчивается точкой. Найти длину самого длинного слова.
7. Имеется строка символов, содержащая буквы латинского алфавита и цифры. Определить количество цифр в строке.
8. Преобразовать строку, заменив все «:» запятыми, встречающиеся среди первых [n/2] символов и заменив точками все «!», встречающиеся среди символов, стоящих после [n/2] символов.
9. В предложении найти слова, начинающиеся и заканчивающиеся на одну и ту же букву.
10. Заменить в заданной строке знак "!" на сочетание "???".
11. Определить сколько раз в тексте встречаются гласные буквы.
12. В данной строке подсчитать сумму цифр, содержащихся в ней.
13. В заданной строке удвоить каждый символ.
14. Вводится строка. Поставить запятую после каждого пробела данной строки.
15. Вводится строка. Удалить из строки лишние пробелы.
16. Вводится строка. Удалить из строки пробелы, стоящие перед запятыми.
17. Дана строка и число n. Верно ли, что в ней есть по крайней мере n подряд идущих букв a?
18. Дана строка. Удалить из нее последовательности символов, расположенные между скобками. Скобки также удалить. После первой открывающей скобки другие открывающие скобки игнорируются. Если скобка не закрывается до конца строки, то удалять все до конца строки. Закрывающая скобка без парной открывающей игнорируется.
19. Заменить в исходной строке все 0 на заданный символ.
20. В исходной строке вместо заданного символа вставьте 0.

21. Удалить из строки все одинаковые символы, идущие подряд.
22. Добавить в строку недостающие пробелы (после знаков препинания обязательно должен быть пробел).
23. Подсчитать в строке число букв A и B. Найти разницу N между этими числами и вывести N раз букву, которая встречается чаще.
24. Определить является ли заданная строка палиндромом без учета регистра.
25. Написать программу, разбивающую текст на строки длиной не N символов.
26. Поменять местами в предложении все 0 и 1.