Лабораторна робота №6

ПЕРЕТВОРЕННЯ З ОДНІЄЇ КОДОВОЇ ТАБЛИЦІ В ІНШУ

Мета: Навчитися працювати з текстовими файлами на рівні операційної системи, закріпити навички обробки текстових рядків, навчитися працювати з різними кодовими таблицями та використовувати команду XLAT.

Обладнання: будь-який текстовий редактор що не використовує службових символів, компілятор асемблера (TASM або MASM), дебагер (рекомендується Turbo Debuger).

Хід роботи

1. Постановка задачі та розробка алгоритму.

В цій лабораторній роботі ми вперше починаємо працювати з файлами та файловою системою. Перед нами стоїть дві задачі: по-перше ми повинні навчитися працювати з файлами: відкривати файл для зчитування, читати з файлу, створювати новий файл, записувати в файл тощо. Крім того, ми вчимося переводити символи з однієї кодової системи в іншу. Не секрет, що в таблиці символів ASCII стандартними є лише 128 перших символів. Всі інші символи можуть мати різні кодування в різних операційних системах та при різних настройках комп'ютера. Так, наприклад, IBM cp437 – кодування другої половини ASCII таблиці, що використовується BIOS – не містить взагалі символів кирилиці.

Символи кирилиці містять такі кодові таблиці, як KOI8-r (використовується в Linux та інших безкоштовних операційних системах, є транспортним кодування в мережі Internet), ISO 8859-5 (використовується в багатьох комерційних UNIX- системах) і cp1251 (використовується в Windows). Але в усіх цих таблицях коди символів мають різне кодування. Тому не рідко виникають ситуації, коли отримане повідомлення, або текст якоїсь статті ми не в змозі прочитати через те, що написане воно було в іншій кодовій таблиці.

В нашій лабораторній роботі ми навчимося перекодувати символи з однієї таблиці в іншу. У даному випадку, у нас своя таблиця: коли ми ведемо у перший фалй тільки цифри, то вони нормально відобразяться у другому; а от коли напишемо букви, то отримаємо „кашу" – це ж і є наша власна таблиця кодування, за винятком слова – „TASM!". Освоївши на лекційних заняттях функції роботи з файлами, ми відкриємо потрібний нам файл, будемо читати з нього символи, міняти їх кодування та записувати до іншого файлу.

Алгоритм нашої програми буде наступний.

1. Відкриваємо перший файл для читання.
2. Відкриваємо (створюємо) другий файл для запису.
3. Зчитуємо символ з першого файлу.
4. Якщо жодного символу не зчиталося, значить вже кінець файлу, переходимо на крок 8.
5. Якщо це символ кирилиці – перекодовуємо його.
6. Записуємо результат перекодування в другий файл.
7. Переходимо на шаг 3.
8. Закриваємо всі файли.

За цим алгоритмом потрібно скласти блок-схему та написати програму.

2. Написання програми.

Однією з важливих задач операційної системи є керування розміщенням даних у зовнішній пам'яті (пам'яті довгострокового збереження). У сучасних ПК як пристрої

зовнішньої пам'яті найчастіше використовуються дискові нагромаджувачі, у першу чергу — нагромаджувач на твердому магнітному диску (HDD або «вінчестер»). Порція інформації (найчастіше однорідної), що зберігається в зовнішній пам'яті під визначеним ім'ям, називається файлом.

Підсистема обліку розміщення інформації на пристрої зовнішньої пам'яті називається файловою системою. Від неї залежить, яка одиниця простору пам'яті є найменшої, чи можуть файли займати незв'язані ділянки пам'яті, як іменуються файли, якими властивостями може володіти файл і які операції над ним можна робити і т.п. У DOS для дискових нагромаджувачів використовується файлова система FAT. Вона дозволяє розділяти дисковий простір HDD на кілька розділів і організовувати в кожнім з розділів ієрархічну структуру каталогів, що містять файли. Каталог,, що представляється звичайно у виді іменованої сукупності файлів, також по суті є файлом визначеної структури, що містить список файлів, що зберігаються в ньому. (Для роботи з каталогами в DOS маються спеціальні функції).

Ім'я файлу (і, відповідно, каталогу) може містити від 1 до 8 символів безпосередньо в імені і від 0 до 3 символів — у розширенні файлу, що звичайно пояснює його тип. Ім'я і розширення відокремлюються крапкою. Сумарна довжина імені, таким чином, складає максимум 12 байт.

DOS підтримує ряд наступних функцій для роботи з файлами:

№ у	Опис	Вхід	Вихід
3Ch	Створити файл/ новий файл	DS:DX → ASCIIZ-ім'я файлу*; CX = атрибути	AX = дескриптор
3Dh	Відкрити файл	DS:DX → ASCIIZ-ім'я файлу; AL = код доступу (0 - читання, 1 - запис, 2 - читання/запис)	AX = дескриптор
3Eh	Закрити файл	BX = дескриптор	
3Fh	Читати з файлу	BX = дескриптор; CX = число байт; DS:DX → буфер-приймач	AX = число лічених байт
40h	Писати у файл	BX = дескриптор; CX = число байт; DS:DX → буфер-джерело	AX = число записаних байт

*ASCIIZ – '(шлях\)ім'я',0 (якщо шлях не зазначений, використовується поточний каталог).

Усі функції у випадку помилки встановлюють CF і повертають у AX код помилки:

Hex	Dec	Значення	Hex	Dec	Значення
1	1	Невірний номер функції	0Ah	10	Невірне оточення
2	2	Файл не знайдений	0Bh	11	Невірний формат
3	3	Шлях не знайдений	0Ch	12	Невірний код доступу
4	4	Занадто багато відкритих файлів	0Dh	13	Невірна дата
5	5	Доступ не дозволений	0Eh	14	(не використовується)
6	6	Невірний дескриптор	0Fh	15	Задано невірний диск
7	7	Зруйновано блоки упр. пам'яттю	10h	16	Не можна видаляти тек.
8	8	Недостатньо пам'яті	11h	17	каталог
9	9	Невірна адреса блоку пам'яті	12h	18	Не той пристрій
					Більше немає шуканих файлів

Стандартні дескриптори:

0 CON - стандартне введення (клавіатура);
1 CON - стандартний висновок (екран);
2 стандартний пристрій помилок (екран);
3 AUX - асинхронний адаптер (COM1);
4 стандартний принтер (LPT1).

Атрибути (біти):

	a	d	s	v	h	r

a - (archive) архівний;
d - (directory) каталог;
v - (volume) мітка тому;
s - (system) системний;
h - (hidden) схований;
r - (read only) тільки для читання.

Розглянемо основні поняття, які необхідно мати при використанні цих функцій.

Починаючи з версії 2 у MS-DOS використовується дескрипторний метод роботи з файлами. При створенні або відкритті файлові привласнюється 16-розрядний двоїчний номер, називаний дескриптором (або описателем). Надалі при виконанні операцій читання, запису й інших необхідно вказувати привласнений файлові дескриптор.

Мається п'ять визначених дескрипторів, що відповідають стандартним пристроям уведення-висновку: клавіатурі, екранові і комунікаційним портам. Використання визначених дескрипторів дозволяє здійснювати введення-висновок з цими пристроями точно так само, як і з файлами, а також організовувати потоки даних, що не залежать від типу джерел і приймачів даних.

3. Тестування програми. Аналіз результатів.

Після написання програми її обов’язково треба перевірити за допомогою дебагера. Ми повинні з’ясувати, чи вірно працює наша програма, чи не шкодить вона зайвих файлів, чи правильно перекодує символи, чи не виникає нескінченних циклів або інших помилок при роботі нашої програми.

По закінченню тестування ми повинні отримати дієздатну програму, що буде перекодовувати текстовий файл з однієї кодової таблиці в іншу.

4. Висновки.

В цій лабораторній роботі ми вперше ознайомились з функціями роботи з файлами. Ми навчилися відкривати та створювати файли, зчитувати з файлів інформацію та записувати данні в файл, ми навчилися закривати файли при завершенні роботи з ними.

Також ми познайомилися з різними кодовими таблицями та навчились перекодовувати символи з однієї кодової таблиці в іншу. Такі навички стануть важливим і необхідним інструментом в нашій подальшій роботі та професійній діяльності.

5. Контрольні запитання.

1). Які ви знаєте функції для створення та відкриття файлу?
2). Які ви знаєте функції для зчитування та запису в файл?
3). Яку ви знаєте функцію для закриття файлу.
4). Які кодові таблиці вам відомі?
5). Як працює команда XLAT?

Код програми:

```
SEGM segment
assume CS:SEGM, DS:SEGM, ES:SEGM, SS:SEGM
org 100h
Begin:
;open file
 mov ah,3dh ; Ф-ція для відкриття фалу
 mov al,0 ;режим відкриття (0-для зчитування, 1- для записи, 2- для зчитування і запису )
 lea dx,file ;адрес строки з іменем файла
 int 21h

 jc f_n_f ;cf=1   ; якщо файл не знайдений, який ми хочемо відкрити, то CF встановиться в 1
(помилка, вихід)

 mov bx,ax
 mov handle,bx ;"описатель" файла, можна так сказати, "указатель" на файл

 ;create file
 mov ah,3ch  ; Ф-ція для створення файлу
 lea dx,sec_file  ;адрес строки з іменем файла
 xor cx,cx  ;обнулення регістру  cx
 int 21h

 jc cnt_create ;  CF встановиться в 1, якщо файл не можливо створити

 mov bx,ax
 mov handle_two,bx  ;"описатель" файла, можна так сказати, "указатель" на файл
 ;;;;

 ;reading in bufer
next_char:  ; будемо вибирати по одній букві із файлу. і обробляти їх відповідно
 mov ah,3fh  ;Ф-ція, за допомогою якою модна читати із файлу через  "описатель"
 mov cx,1    ; кількість байт, які ми хочемо прочитати
 lea dx,bufer ; адрес буфера для зчитування
 mov bx,handle  ; "описатель" файла, можна так сказати, "указатель" на файл
 int 21h

 jc close_file  ; якщо неможливо зчитати із файла, то CF встановиться в 1

 cmp cx,ax  ; ax - кількість прочитаних байт, якщо рівно 1, значить є чимвол ідемо на
обробку..., якщо немає кінець файлу
 je  is_char      ; ==
 ;not char
 mov end_file,1 ;тимчасова змінна, в якій ми зберігаємо наявність кінця файлу..

 mov cx,ax  ;skilki zapisalo byte


is_char:
 ;;;;;;;;;;;;;;;;;;;
 ;Детальніша робота ф-ції "xlat" - описана в лаб. роб. № 5
 ;;;;;;;;;;;;;;;;;;;;
 mov ah,bufer  ; поміщуємо символ із буфера, який ми хочемо обробити....
 lea bx,my_table  ; встановлюємо ефективний адрес з нашою таблицею символів
 mov al,ah
 xlat
 mov bufer,al  ; заносимо в буфер оброблений символ

 mov ah,40h ;Ф-ція, для запису у файл через  "описатель"
 mov bx,handle_two  ;"описатель" файла, можна так сказати, "указатель" на файл
 lea dx,bufer  ; встановлюємо ефективний адрес з буфером...
 int 21h

 ;;;;;;;;;;;;;;
```

```
 cmp end_file,1    ;порівнюємо чи ще немає кінця файлу
 je close_file     ; якщо кінець, то перейти на закриття файлу

 jmp next_char
 ;---------

 ;close my file
close_file:
; Ф-ції, для закриття файлів, щоб все було коректно...
 mov ah,3eh
 mov bx,handle_two
 int 21h

 mov ah,3eh
 mov bx,handle
 int 21h
 jmp exit


f_n_f:
 mov ah,09h
 lea dx,file_not_found
 int 21h

cnt_create:
 mov ah,09h
 lea dx,not_create_file
 int 21h

exit:
 mov ax,4c00h
 int 21h

 ;data
 file db 'first.txt',0
 sec_file db 'second.txt',0
 file_not_found db 'Error: File ',34,'first.txt',34,' not found$'
 not_create_file db 'Error: Can not create file$'
 handle dw ?
 handle_two dw ?
 bufer db 200 dup(?)
 end_file db 0
 my_table db 'QWERTYUIOPASDFGHJKLZXCVBNMqwertyuNopasdasduioqwen1234567890qwejkl-
xcvbnmsdfbaAsdlfkMXvXsjkdlfnqk2-l45738euhfasAhdfa87w9heMjaskdnfau0wuftwefsdfSWFHAS'
 SEGM ends
end Begin
```

Додаткові завдання за варіантами:

0. Написати на мові асемблер com-програму, щоб перекодувати символи з однієї таблиці в іншу (довільну).
1. Написати на мові асемблер exe-програму, щоб перекодувати символи з однієї таблиці в іншу (довільну).
2. Написати на мові асемблер com-програму, щоб перекодувати з кирилиці на латиницю за правилом найпростішої заміни символів з однієї таблиці в іншу.
3. Написати на мові асемблер exe-програму, щоб перекодувати з кирилиці на латиницю за правилом найпростішої заміни символів з однієї таблиці в іншу.
4. Написати на мові асемблер com-програму, щоб перекодувати символи строки шляхом додавання до літер строки кодів символів таблиці.
5. Написати на мові асемблер exe-програму, щоб перекодувати символи строки шляхом додавання до літер строки кодів символів таблиці.
6. Написати на мові асемблер com-програму, для кодування строки шляхом додавання до літер строки кодів обмеженого рядка з файлу.
7. Написати на мові асемблер exe-програму, для кодування строки шляхом додавання до літер строки кодів обмеженого рядка з файлу.
8. Написати на мові асемблер com-програму, для розкодування строки шляхом віднімання від літер строки кодів обмеженого рядка з файлу.
9. Написати на мові асемблер exe-програму, для розкодування строки шляхом віднімання від літер строки кодів обмеженого рядка з файлу.