Лабораторна робота №3
Робота з таблицею ASCII

Мета роботи

Ознайомитись з командами умовного те безумовного переходу, арифметичними діями в асемблері, отримати уяву про побудову циклів, поглибити знання функцій для введення і виведення тексту, закріпити навички роботи зі строковими величинами.

Хід роботи

1. Постановка задачі та розробка алгоритму.

Задача цієї лабораторної роботи: організувати вивід всіх 256 символів ASCII на екран у вигляді таблиці 16x16. Слід пам'ятати, що побудована нами таблиця повинна бути якомога повнішою, тобто треба використовувати ті функції виводу символу на екран, що не обробляють службових символів (не видають звуковий сигнал по коду 7, не обробляли перехід на інший рядок тощо). Виводити символи ми будемо починаючи з 0-го і закінчуючи 255. Щоб вивести символи у вигляді таблиці 16x16 ми повинні в той момент, коли номер символу стане кратним 16, робити перехід на новий рядок.
Всі ці етапи будуть проходити в циклі з лічильником, що буде змінюватися від 0 до 255.
Алгоритм поставленої перед нами задачі наступний:

1. Встановлюємо код символу в 0.
2. Встановлюємо лічильник в 256.
3. Виводимо символ.
4. Збільшуємо код символу на 1.
5. Якщо код символу кратний 16 – переходимо на наступний рядок.
6. Зменшуємо лічильник.
7. Якщо лічильник не дорівнює 0, повертаємося на 3.
8. Завершення програми.

За цим алгоритмом студенти повинні побудувати блок-схему.

2. Написання програми.

Скориставшись будь-яким текстовим редактором, що не використовує службових символів, необхідно за складеним алгоритмом написати програму на мові асемблера.
Для виведення символів найкраще користуватися функцією 09h переривання INT 10h. Ця функція базової системи вводу-виводу виведе абсолютно всі символи ASCII не обробляючи службових символів.
Враховуючи, що ця функція інтерпретує як символ значення в регістрі AL, доцільно буде з самого початку використовувати цей регістр для зберігання коду символу. Щоб побудувати таблицю 16x16 нам потрібно контролювати кратність 16-ти коду поточного символу. Якщо код символу кратний 16 – треба перемістити курсор на наступний рядок. Контролювати кратність можна поділивши код символу на 16 і перевіривши залишок.
Переміщати курсор можна використовуючи функції BIOS переміщення курсору, або виводячи на екран символи 10, 13 (але в останньому випадку ми не можемо користуватися функцією 09h переривання INT 10h.
Для побудови циклу можна скористатися або командами умовного переходу, або командою LOOP.

3. Тестування програми. Аналіз результатів.

Після написання програми необхідно перевірити її працездатність.

При програмуванні на низькому рівні найкраще це зробити за допомогою дебагера, наприклад Turbo Debugger'а, що постачається разом з пакетами Turbo Assembler та Turbo C.

4. Висновки.

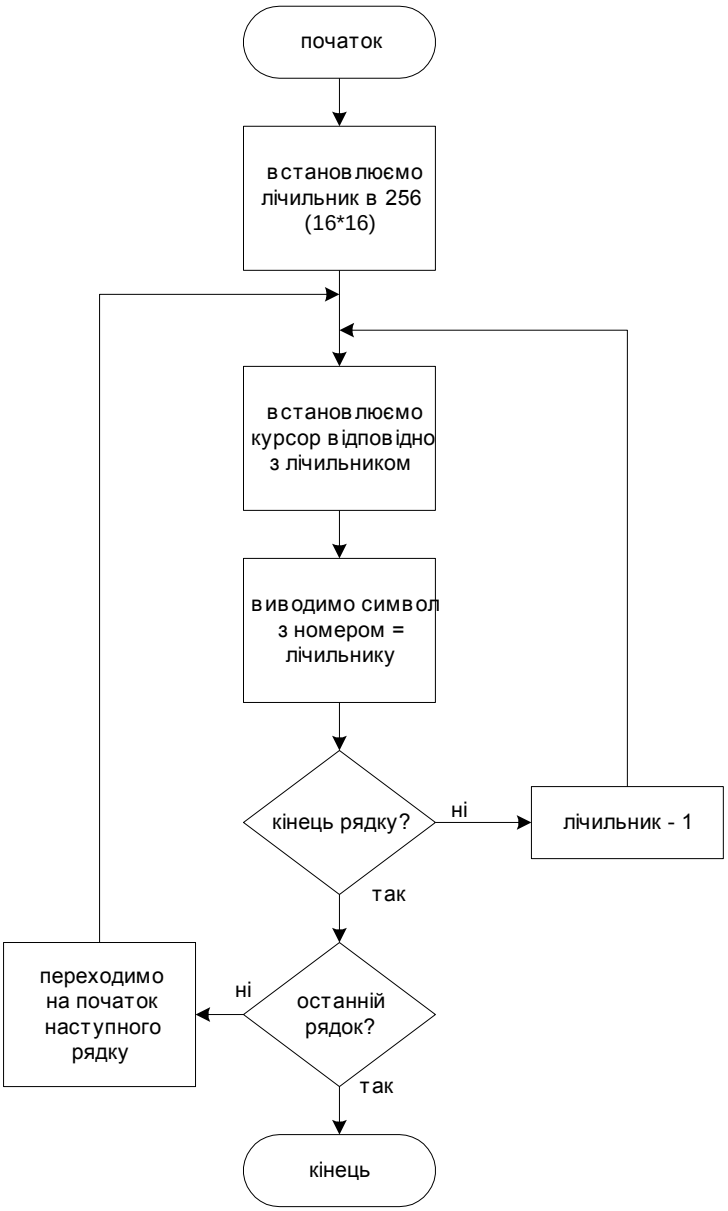
Спілкування між людиною та комп'ютером в більшості програм реалізується за допомогою введення та виведення символів в таблиці ASCII. Вміння працювати з символами цієї таблиці гарантує грамотне написання діалогових програм. Крім того студенти під час виконання третьої лабораторної роботи навчаються будувати цикли, використовувати команди логічних операцій, закріплюють навички роботи з дисплеєм.

5. Контрольні запитання.

1. Аналогом якої логічної операції є команда TEST?
2. Як будується цикл з використанням команди LOOP?
3. Які символи таблиці ASCII є службовими і зазвичай не відображаються на екрані?
4. Які функції BIOS управління курсором ви знаєте?
5. Які функції виводу символу на екран не обробляють службових символів?

Текст програми

```
1     model tiny
2     .code
3     .startup
4             mov cl,16 ; лічильник стовпчиків - індекс j
5             mov ch,16 ; лічильник рядків - індекс i
6             mov dh, cl ; встан. 16 рядок
7     l1:
8             mov dl, cl ; встан. стовпчик за лічильником
9             jmp l3
10    l2:
11            mov dh, cl ; встан. рядок за лічильником
12            jmp l3
13    l3:
14            mov ah,2 ; функція 02 – встан. курсору
15            mov bh,0
16            int 10h ; 10-те переривання (BIOS)
17            mov bx,8 ; звичайні атрибути символу
18            mov ax,0
19            mov al,16 ;
20            mul dl ;
21            add al,dh ; вираховуємо з положення в таблиці номер символу
22            mov ah,9 ; функція 09 – виведення символів
23            mov cx,1 ; виводимо 1 символ
24            int 10h ; 10-те переривання
25
26            mov cl, dl ; відновлюємо лічильник стовпчиків
27            loop l1
28
29            mov dl,16 ; повертаємось в початок нового рядку
30
31            mov cl, dh ; відновлюємо лічильник рядків
32            loop l2
33            mov dh, cl
34
35            ret
36            end
```



Базові інструкції, переривання, та більшість операторів, які тут використовуються, ми детально розглянули в першій та другій роботі.
Розглянемо нові функції, які тут використовуються.

mul – ця інструкція перемножує значення регістра AX на її аргумент без знаків.

jmp – ця інструкція виконує безумовний перехід на задану мітку.

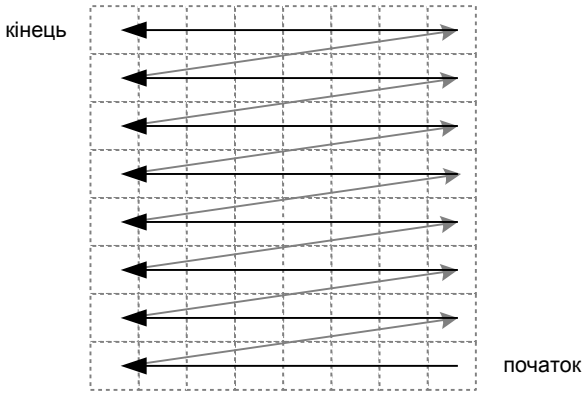
Також ми використовуємо дві функції переривання **10h** (переривання BIOS), а саме:
- **02h** – встановлення курсору
- **09h** – виведення символу з позиції курсору

Функція **02h** ініціалізується наступним чином:
AH = 02h (номер функції)
BH = номер відео-сторінки
DH = номер рядку
DL = номер стовпчику
Функція **09h**:
AH = 09h (номер функції)

AL = ASCII-код символу, що виводиться
BH = номер відео-сторінки
BL = атрибути символу (колір фону та символу)
CX = кількість символів, що виводиться
Ми використовуємо саме цю функцію для виведення на екран, тому що вона **не обробляє** службові символи ASCII, на відміну від аналогічної функції переривання 21h, яку ми використовували в 1-й роботі.
Використовуючи ці функції, блок-схему, та знання з першої роботи, можна легко написати програму. Але перед цим звичайно, треба зазирнути у наш улюблений розділ…

А чи знаєте ви, що…

У мові асемблера, як ми зазначили у другій роботі, лічильники циклів змінюються не від **0 до N**, а навпаки, від **N до 0**, іншими словами, відлік проходить з кінця. Цю особливість необхідно мати на увазі, особливо в цій роботі.
Так як таблицю ми будемо виводити, використовуючи цикли, символи ми також будемо виводити незвичним чином. Якщо звичайно ми виводимо елементи таблиці, починаючи з лівого верхнього кута до нижнього правого, в асемблері ми все робимо навпаки (див. малюнок). Це дуже принциповий момент і варто його **запам'ятати** (бажано, по можливості, разом з рештою матеріалу у цій роботі).



Додаткові завдання за варіантами:

0. Написати com-програму, яка виведе на екран тільки перші 16 символів ASCII-таблиці у вигляді матриці 4x4
1. Написати exe-програму, яка виведе на екран тільки перші 16 символів ASCII-таблиці у вигляді матриці 4x4
2. Написати com-програму, яка виведе на екран тільки другі 16 символів ASCII-таблиці у вигляді матриці 4x4
3. Написати exe-програму, яка виведе на екран тільки другі 16 символів ASCII-таблиці у вигляді матриці 4x4
4. Написати com-програму, яка виведе на екран тільки останні 16 символів ASCII-таблиці у вигляді матриці 4x4
5. Написати exe-програму, яка виведе на екран тільки останні 16 символів ASCII-таблиці у вигляді матриці 4x4
6. Написати com-програму, яка виведе на екран символи ASCII-таблиці з 64 по 127 у вигляді матриці 8x8
7. Написати exe-програму, яка виведе на екран символи ASCII-таблиці з 64 по 127 у вигляді матриці 8x8
8. Написати com-програму, яка виведе запит про код початкової і кінцевої літер з ASCII-таблиці і виведе на екран тільки ті літери, які опиняться поміж ними
9. Написати exe-програму, яка виведе запит про код початкової і кінцевої літер з ASCII-таблиці і виведе на екран тільки ті літери, які опиняться поміж ними