

Лекція-7. Двовірні масиви. Типові операції з масивами

- *Проблеми представлення багатовимірних масивів виникають через відсутність спеціальних засобів для опису такого типу даних. Двовірний масив потрібно моделювати. На описі самих даних це майже ніяк не відбивається — пам'ять під масив виділяється за допомогою директив резервування й ініціалізації пам'яті.*

Двовірні масиви

- Безпосереднє моделювання обробки масиву виробляється в сегменті коду, де програміст, описуючи алгоритм обробки асемблеру, визначає, що деяку область пам'яті необхідно трактувати як двовірний масив.
- При цьому ви вільні у виборі того, як розуміти розташування елементів двовірного масиву в пам'яті: по чи рядках по стовпцях.

Двовірні масиви

- Якщо послідовність однотипних елементів у пам'яті трактується як двовірний масив, розташований по рядках, то адреса елемента (i, j) обчислюється по формулі

$$(\text{база} + m * \text{розмір_елемента} * i + j)$$

де

- m – кількість елементів у рядку
- $i = 0 \dots n-1$ указує номер рядка,
- $j = 0 \dots m-1$ указує номер стовпця.

Двовірні масиви

- Наприклад, нехай маємо масив чисел (розміром у 1 байт) $mas(i, j)$ з розмірністю 4 на 4

($i = 0 \dots 3, j = 0 \dots 3$):

23040567

05060799

67080923

87090008

Двовірні масиви

- У пам'яті елементи цього масиву будуть розташовані в наступній послідовності:

23 04 05 67 05 06 07 99 67 08 09
23 87 09 00 08

Двовірні масиви

- Якщо ми хочемо трактувати цю послідовність як двовірний масив, приведений вище, і витягти, наприклад, елемент
- **$\text{mas}(2, 3) = 23$,**
- то провівши нехитрий підрахунок, переконаємося в правильності наших міркувань:
- **Ефективна адреса**
 $\text{mas}(2, 3) = \text{mas} + 4 * 1 * 2 + 3 = \text{mas} + 11$

Двовірні масиви

- Подивимось на представлення масиву в пам'яті і переконаємось, що по цьому зсуву дійсно знаходиться потрібний елемент масиву.

• 23 04 05 67 05 06 07 99 67 08 09 **23** 87
09 00 08

Двовірні масиви

- Організувати адресацію двовірного масиву логічно, використовуючи розглянуту нами раніше *базово-індексну адресацію*.



Доступ до елементів масиву

- При цьому можливі два основних варіанти вибору компонентів для формування ефективної адреси:
- сполученням прямої адреси, як базового компонента адреси, і двох індексних реєстрів для збереження індексів:
 - `mov ax,mas[ebx][esi]`
- сполученням двох індексних реєстрів, один із яких є і базовим і індексної одночасно, а іншої — тільки індексним:
 - `mov ax,[ebx][esi]`

;Фрагмент програми вибірки елемента
;масиву mas(2,3) і його обнуління

.data

mas db

23,4,5,67,5,6,7,99,67,8,9,23,87,9,0,8

i=2

j=3

.code

...

mov si,4*1*i

mov di,j

mov al,mas[si][di]

;у al елемент mas(2,3)

Приклад

- Як закінчений приклад розглянемо програму пошуку елемента в двомірному масиві чисел
- Елементи масиву задані статично.



```

MASM
MODEL small
STACK 256
.data
;матриця розміром 2x5
;для наочності її можна описати так:
;array  dw 2 DUP (5 DUP (?))
;але ми її ініціалізуємо
array  dw 1,2,3,4,5,6,7,3,9,0
;логічно це буде виглядати так:
;array= {1 2}; {3 4}; {5 6}; {7 3}; {9 0}
        elem dw 3
;елемент для пошуку
failed  db 0ah,0dh,'Немає такого
        елемента в масиві!','$'
success db 0ah,0dh,'Такий елемент у
        масиві присутній','$'
foundtime  db ?
;кількість знайдених елементів
fnd db ' раз(ів)',0ah,0dh,'$'

```

```

.code
main:
mov ax,@data
mov ds,ax
xor ax,ax
mov si,0 ;si=стовпці в матриці
mov bx,0 ;bx=рядка в матриці
mov cx,5
;число для зовнішнього циклу (рядки)
external:
;зовнішній цикл по рядках
mov ax,array[bx][si]
;у ax перший елемент матриці
push cx
; в лічильник зовнішній цикл
mov cx,2
; внутрішній цикл ( стовпці)
mov si,0
internal:
;внутрішній цикл по рядках
inc si
; на наступний елемент у рядку
;порівнюємо вміст поточного елемента
;в ax із шуканим елементом:
cmp ax,elem

```

```
;якщо поточний збігся із шуканим,  
;то перехід на here для обробки,  
;інакше цикл продовження пошуку  
je here  
;інакше — цикл по рядку cx=2 разів  
loop internal  
here:   jcxz  
move_next ;переглянули рядок?  
inc foundtime  
;інакше збільшуємо лічильник  
;що збіглися  
move_next: ;просування в матриці  
pop cx  
;відновлюємо CX зі стека (5)  
add bx,1  
;пересуваємося на наступну рядок  
loop external;цикл (зовнішній)  
cmp foundtime,0h  
;порівняння числа збігів з 0  
ja eqf  
;якщо більше 0, то перехід  
not_equal:  
;немає елементів, що збіглися
```

```
mov ah,09h  
mov dx,offset failed  
int 21h  
jmp exit ;на вихід  
eqf:  
;є елементи, що збіглися із шуканим  
mov ah,09h  
mov dx,offset success  
int 21h  
mov ah,02h  
mov dl,foundtime  
add dl,30h  
int 21h  
mov ah,09h  
mov dx,offset fnd  
int 21h  
exit:  
mov ax,4c00h  
int 21h  
end main
```

Домашнє завдання

- Розібрати програму сортування масиву, яка представлена в електронному конспекті лекцій

