Лабораторна робота №4

Робота з одновимірними та двовимірними масивами

Мета: Навчитися працювати з масивами. Освоїти методи описання та ініціалізацію масивів.

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.

При роботі з масивами необхідно чітко представляти собі, що всі елементи масиву розташовуються в пам'яті комп'ютера послідовно. Але таке розташування нічого не говорить про призначення і порядок використання цих елементів. І тільки лише програміст за допомогою складеного їм алгоритму обробки визначає, як потрібно трактувати цю послідовність байт, що складають масив. Так, ту саму область пам'яті можна трактувати як одномірний масив, і одночасно ті ж самі дані можуть трактуватися як двомірний масив. Усі залежить тільки від алгоритму обробки цих даних у конкретній програмі. Самі по собі дані не несуть ніякої інформації про своєму *"значеннєвому"*, або *логічному*, типі. Помнете про цей принциповий момент.

Це розуміння можна поширити і на ***індекси елементів масиву***. Асемблер не підозрює про їхнє існування і йому абсолютно все рівно, які їх чисельні значеннєві значення.

Для того щоб локалізувати визначений елемент масиву, до його імені потрібно додати *індекс*. Тому що ми моделюємо масив, те повинні подбати і про моделювання індексу. У мові асемблера індекси масивів — це звичайні адреси, але з ними працюють особливим образом. Іншими словами, коли при програмуванні на асемблері ми говоримо про індекс, те скоріше маємо на увазі під цим не номер елемента в масиві, а деяка адреса. Давайте ще раз звернемося до опису масиву. Наприклад, у програмі статично визначена послідовність даних:

```
mas     dw      0,1,2,3,4,5
```

Нехай ця послідовність чисел трактується як одномірний масив. Розмірність кожного елемента визначається директивою **dw**, тобто вона дорівнює **2** байти. Щоб одержати доступ до третього елемента, потрібно до адреси масиву додати **6**. Нумерація елементів масиву в ассемблере починається з нуля.

Тобто в нашому випадку мова, фактично, йде про **4-м** елемент масиву — 3, але про це знає тільки програміст; мікропроцесорові в даному випадку все рівно — йому потрібний тільки адреса.

У загальному випадку для одержання адреси елемента в масиві необхідно початковий (базовий) адреса масиву скласти з добутком індексу (номер елемента мінус одиниця) цього елемента на розмір елемента масиву:

база + (індекс*розмір елемента)

Архітектура мікропроцесора надає досить зручні програмно-апаратні засоби для роботи з масивами. До них відносяться базові й індексні регістри, що дозволяють реалізувати кілька режимів адресації даних. Використовуючи дані режими адресації, можна організувати ефективну роботу з масивами в пам'яті. Згадаємо ці режими:

- ***індексна адресація зі зсувом*** — режим адресації, при якому ефективна адреса формується з двох компонентів:
  - **постійного (базового)** — указівкою прямої адреси масиву у виді імені ідентифікатора, що позначає початок масиву;
  - **змінного (індексного)** — вказівкою імені індексного регістра.

  Приклад:

```
mas     dw      0,1,2,3,4,5
...
        mov     si,4
        mov     ax,mas[si] ;помістити si-й елемент масиву mas у регістр ax:
```

- ***базова індексна адресація зі зсувом*** — режим адресації, при якому ефективна адреса формується максимум із трьох компонентів:

- **постійного** (необов'язковий компонент), у якості якої може виступати пряма адреса масиву у виді імені ідентифікатора, що позначає початок масиву, або безпосереднє значення;
- **змінного (базового)** — вказівкою імені базового регістра;
- **змінного (індексного)** — вказівкою імені індексного регістра.

Цей вид адресації зручно використовувати при обробці двомірних масивів. Приклад використання цієї адресації ми розглянемо далі при вивченні особливостей роботи з двомірними масивами.

З представленням одномірних масивів у програмі на асемблері й організацією їхньої обробки всі досить просто. А як бути якщо програма повинна обробляти двомірний масив? Усі проблеми виникають як і раніше через те, що спеціальних засобів для опису такого типу даних в асемблері немає. Двомірний масив потрібно моделювати. На описі самих даних це майже ніяк не відбиває — пам'ять під масив виділяється за допомогою директив резервування й ініціалізації пам'яті.
Безпосереднє моделювання обробки масиву виробляється в сегменті коду, де програміст, описуючи алгоритм обробки асемблеру, визначає, що деяку область пам'яті необхідно трактувати як двомірний масив. При цьому ви вільні у виборі того, як розуміти розташування елементів двомірного масиву в пам'яті: по рядках або по стовпцях.
Якщо послідовність однотипних елементів у пам'яті трактується як двомірний масив, розташований по рядках, то адреса елемента (i, j) обчислюється по формулі
(база + кількість_елементів_у_рядку * розмір_елемента * i+j)
Тут i = 0...n–1 указує номер рядка, а j = 0...m–1 указує номер стовпця.
Наприклад, нехай мається масив чисел (розміром у 1 байт) mas(i, j) з розмірністю 4 на 4
(i= 0...3, j = 0...3):

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

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
Якщо ми хочемо трактувати цю послідовність як двомірний масив, приведений вище, і витягти, наприклад, елемент
mas(2, 3) = 23, то провівши нехитрий підрахунок, переконаємося в правильності наших міркувань:
Ефективна адреса mas(2, 3) = mas + 4 * 1 * 2 + 3 = mas + 11
Подивитеся на представлення масиву в пам'яті і переконаєтеся, що по цьому зсуві дійсно знаходиться потрібний елемент масиву.
Організувати адресацію двомірного масиву логічно, використовуючи розглянуту нами раніше базово-індексную адресацію. При цьому можливі два основних варіанти вибору компонентів для формування ефективної адреси:
сполучення прямої адреси, як базового компонента адреси, і двох індексних регістрів для збереження індексів:

```
mov ax,mas[ebx][esi]
```

сполучення двох індексних регістрів, один із яких є і базовим і індексним одночасно, а іншої — тільки індексним:

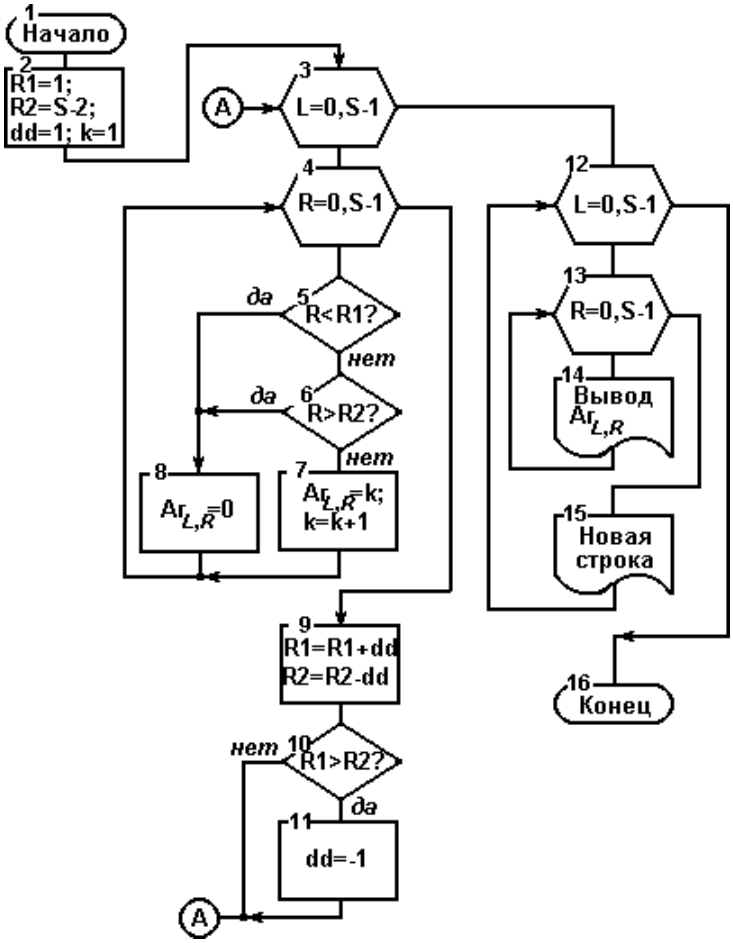
```
mov ax,[ebx][esi]
```

У програмі це буде виглядати приблизно так:

```
;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)
...
```

Якщо ми позначимо розмірність матриці як S, номер рядка як L, а номер стовпця як R, і (маючи у виді, що реалізація алгоритму буде виконана мовою З) домовимося, що нумерація рядків і стовпців буде починатися з 0, то можна визначити, що в рядку з номером L ненульові елементи у верхній частині матриці лежать на стовпцях з номерами R1=L < R < R2=S-L, а в нижньої - R1=S-L-1 < R < R2=L. Отже, алгоритм може складатися з перебору матриці рядок за рядком з визначенням для кожного елемента, чи задовольняють його індекси вищенаведеним умовам. Якщо так - елементові привласнюється наступне значення з ЛП, якщо немає - 0.



Проведемо спрощення алгоритму, обійшовши обчислення граничних значень для кожного елемента і необхідності визначення, у верхню або нижню частину матриці мі попадаємо. Оборотний увага на те, що для першого рядка (L=0) R1=1, R2=S-2. Для кожного наступного рядка R1 збільшується на 1, а R2 зменшується на 1. Коли ми перетинаємо середину матриці, то напрямок модифікації змінюється на протилежне: тепер для кожного наступного рядка R1 зменшується на 1, а R2 збільшується на 1. Ознакою перетинання середини може бути умова R1 > R2, вона виконується в момент перетинання. Схема останнього алгоритму показана на малюнку.

Разом з описаними вище змінними R1 і R2, що одержують початкові значення для першого рядка матриці, мі вводимо змінну dd з початковим значенням 1 - це те значення, що буде модифікувати R1 і R2 для кожного наступного рядка, і змінну k - у якій буде значення поточного члена ЛП, початкове значення - 1 (блок 2). Далі організуються вкладені цикли. В зовнішньому циклі перебираються рядки (блок 3), а у внутрішньому - стовпці матриці (блок 4). У кожній ітерації внутрішнього циклу номер стовпця R порівнюється з граничними значеннями R1, R2 (блоки 5,6). Якщо він лежить у межах від R1 до R2, то

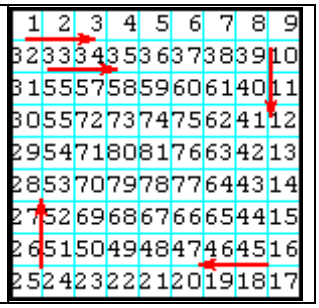
поточному членові матриці привласнюється значення k - поточного члена ЛП, а потім k збільшується на 1 (блок 7). Якщо ні, що текет членові привласнюється значення 0 (блок 8).

Після виходу з внутрішнього циклу модифікуються граничні значення: R1 збільшується на dd, а R2 зменшується на dd (блок 9). Нагадаємо, що початкове значення dd=1. Коли виконується умова R1 > R2 (блок 10) ми привласнюємо dd значення -1, далі модифікація границь буде відповідати правилам для нижньої частини матриці.

Після виходу з зовнішнього циклу, що почався в блоці 3, знову організуються вкладені циклі перебору рядків (блок 12) і стовпців (блок 13). У кожній ітерації внутрішнього циклу виводиться значення одного елемента матриці (блок 14), після виходу з внутрішнього циклу починається новий рядок висновку (блок 15).

2. Написання програми.

Заповнити матрицю ЛП, від лівого верхнього кута по спіралі: вправо - униз - уліво - нагору.



№ в AH	Описание	Вход	Выход
02Ah	Вивід символа на дисплей	AL-символ, який ми хочемо вивести	-

Для спрощення задачі, ми розглянемо приклад заповнення матриці 3 на 3. Звісно , що можна і більше, тільки тут уже починаються великі недоліки. Тому що потрібно буде використання додаткових функцій, для виведення елементів масиву, які більше 9.

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. Контрольні запитання.

1) Як описати масив у програмі ?
2) Як ініціалізувати масив, тобто як задати початкові значення його елементів?
3) організувати доступ до елементів масиву?
4) Як організувати масиви розмірністю більш однією?
5) Як організувати виконання типових операцій з масивами?

Код програми:

```
SEGM segment
assume CS:SEGM, DS:SEGM, ES:SEGM, SS:SEGM
org 100h
Begin:
  lea dx,mas
```

```
  mov ax,0 ; vis' "X" - початкові елементи масиву
  mov si,0 ; vis' "Y" - початкові елементи масиву
  mov di,0 ; додатковий лічильник, за допомогою якого ми будемо звертатися до елементів
масиву

  mov cx,1 ;загальний лічильний, тобто цифри якими ми будемо заповняти масив

    cikl:
     cmp cx,10  ;порвінюємо з 10, оскільки, розглядаємо масив 3*3+1
     jl go_go ; якщо більше, то переходимо на вихід
     jmp exit
     go_go:

;Правий блок, який заповняє матрицю зліва на право...  RIGHT;;;;;;;;;;;;;
      cmp ax,si ; перевірка на початок основного відліку...
      je right_tmpl
      cmp right,1
      je right_tmpl
      jne end_go

      right_tmpl:
       mov right,1  ; допоміжні змінні за допомогою яких, ми будемо орієнтуватися, в яку
сторону матрицю заповняти, якщо "1" - значить .. в дану сторону рухаємся..
       mov left,0   ;  ----
       mov down,0  ;---
       mov up,0    ; ---

       call set_mas  ; визиваємо процедуру, яка записує дані в елемент масива
       inc di   ; збільшуємо на 1 ...
       inc di

       inc si

       cmp si,k  ; перевіряємо, чи ми не дійшли до правого краю....
       jne end_go
       inc cx
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Оскільки, ми спочатку заповнили весь масив "0", то ми можемо робити перевірку на те. коли
ми будемо записувати останній елемент масиву, щоб одразу вийти, і нічого не переписувати
одне поверх одного... :)
;.......Check end.............
 end_go:
       mov tmp,ax    ; тимчасово зберігаємо дані, які були в регістрі ax
       mul x ;множення x на ax
       shl ax,1  ; аналогічно= ax*2 , просто коли 2-їче число здвинути вліво на один
розряд, то воно збільшить в 2 рази
       mov bx,ax

       cmp mas[bx][di],0 ; порівнюємо з "0", якщо рівно "0" , значить ще не всі елементи
заповнені, а якщо !=0 , то на вихід
       mov ax,tmp
       je down_go  ;якщо рівно
       jmp chk_end

; Напрям до низу....DOWN ;;;;;;;;;;;;;
  down_go:
      cmp si,k
      je down_tmpl
      cmp down,1
      je down_tmpl
      jne left_go

      down_tmpl:
```

```
        mov right,0
        mov left,0
        mov down,1
        mov up,0

        mov tmp,ax
        call set_mas
        mov ax,tmp
        inc ax

        cmp ax,k
        jne left_go
        inc cx
;Напрям вліво LEFT ;;;;;;;;;;;;;
  left_go:
       cmp ax,k
       je left_tmpl
       cmp left,1
       je left_tmpl
       jne up_go

       left_tmpl:
        mov right,0
        mov left,1
        mov down,0
        mov up,0

        mov tmp,ax
        call set_mas
        mov ax,tmp
        dec di
        dec di

        dec si

        mov left_ax,ax
        mov ax,corner
        sub ax,k
        cmp si,ax
        mov ax,left_ax
        jne up_go
        inc cx
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Напрям вверх
;.....  UP ;;;;;;;;;;;;;
  up_go:
       mov left_ax,ax
       mov ax,corner
       sub ax,k
       cmp si,ax
       mov ax,left_ax
       je up_tmpl
       cmp up,1
       je up_tmpl
       jne chk_end

       up_tmpl:
        mov right,0
        mov left,0
        mov down,0
        mov up,1

        mov tmp,ax
```

```
       call set_mas
       mov ax,tmp
       dec ax

       mov tmp,ax
       mul x
       shl ax,1
       mov bx,ax

       cmp mas[bx][di],0
       mov ax,tmp
       je chk_end
       inc ax
       inc si

       inc di
       inc di
       dec k

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
     chk_end:
     inc cx
    jmp cikl

exit:
 mov di,0
 mov ax,0
 mov si,0
 mov cx,x ; лічильник   для виведення масива
 mov tmp_cx,cx
 Ox:
     ;Raw
  Oy:
 mov tmp_ax,ax
 mov tmp_si,si
 mul x
 shl si,1 ; множення на 2 ...
 shl ax,1

    mov bx,ax
    mov dx,mas[bx][si]
    mov al,dl

    add ax,30h   ; додаємо до нашої цифри у масиві 30h, оскільки, з саме з цього числа в
таблиці ASCII кодів, починаютсья цифри,тобто якщо у al=2 , то у таблиці кодів 2 - це
смайлик, але нам потрібне число, отже, ми до 2 додамо 30h=32h , це і є якраз наша 2-ка
    mov dl,al ;подаємо на вхід ф-ції. для виведення
    mov ah,02h  ; ф-ція яка виводить 1 символ на екран
    int 21h

    mov ah,02h ; аналогічна ф-ція, тільки тут ми виводимо пробіл між цифрами, для
коректного відображення масиву
    mov dl,0
    int 21h
    mov si,tmp_si
    inc si

  mov ax,tmp_ax
  loop Oy

    mov dl,13 ; перехід на наступну строку, і встановлення курсора на початок строки, коли
1 рядок масива буде виведено
    mov ah,02h
```

```
    int 21h
    mov dl,10
    mov ah,02h
    int 21h

  mov ax,tmp_ax
  inc ax
  mov si,0
  mov cx,tmp_cx
  inc di
  cmp di,cx ; порівнюємо, чи кількість наших рядків не дорівнює счетчику cx, якщо рівно,
значить всі елементи виведено. на вихід
  je exit_
  inc cx
 loop Ox ; цикл...

exit_:
 mov ax,4c00h ; ф-ця для коректного виходу, і для коректноого завершення роботи
 int 21h

; ПРоцедура , яка записує числа в елементи масиву "по лабіринту"
; вище уже згадувалось як потрібно звернутися до елементів масива, я не буду повторяти ...
 set_mas proc
 mul x ; умножаємо на кількість елементів по горизонталі, зазвичай ми звикли елемент "i"
 shl ax,1 ; *2
 mov bx,ax
 mov mas[bx][di],cx
 ret
 set_mas endp

 ;data
 mas dw 100 DUP (0)  ; створюємо строку довжиною 100, і не забуваємо, що це тип dw ... 2
байта
 x dw 3  ;розмірність по горизнтл.
 y dw 3  ; по вертикл.

 k dw 2 ;допоміжна змінна яка визначає кінець стороки масива..., тобто: x-1=3-1=2
 corner dw 2 ;все аналогічно, просто нам їхх 2 буде потрібно, тому я їх по різному назвав
 left_ax dw 0  ;
 tmp dw 0   ; допоміжні змінні


 right db 0
 left db 0
 down db 0
 up db 0

tmp_ax dw 0
tmp_si dw 0
tmp_cx dw 0


 SEGM ends
end Begin
```

Додаткові завдання за варіантами:

0. Написати com-програму, яка дозволить ввести двовимірний масив 5x5. І вивести з нього головну діагональ
1. Написати exe-програму, яка дозволить ввести двовимірний масив 5x5. І вивести з нього головну діагональ
2. Написати com-програму, яка дозволить ввести двовимірний масив 4x4. І вивести з нього побічну діагональ
3. Написати exe-програму, яка дозволить ввести двовимірний масив 4x4. І вивести з нього побічну діагональ
4. Написати com-програму, яка дозволить ввести двовимірний масив 4x7. І вивести з нього тільки два перших рядки
5. Написати exe-програму, яка дозволить ввести двовимірний масив 4x7. І вивести з нього тільки два перших рядки
6. Написати com-програму, яка дозволить ввести двовимірний масив 5x6. І вивести з нього тільки два останніх рядки
7. Написати exe-програму, яка дозволить ввести двовимірний масив 5x6. І вивести з нього тільки два останніх рядки
8. Написати com-програму, яка дозволить ввести двовимірний масив 4x5. І вивести з нього тільки два перших стовбчика
9. Написати exe-програму, яка дозволить ввести двовимірний масив 4x5. І вивести з нього тільки два перших стовбчика
10. Написати com-програму, яка дозволить ввести двовимірний масив 4x5. І вивести з нього тільки останній стовбчик
11. Написати exe-програму, яка дозволить ввести двовимірний масив 4x5. І вивести з нього тільки останній стовбчик