**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМ**

**ЛАБОРАТОРНА РОБОТА №1.4.**

**Тема: Робота з масивами**

Виконав:

Студент групи 3СТ Чума Д. І.

Перевірив:

Асист. Ваш Ю. В.

Ужгород 2022

**Мета роботи:** Набути вміння і навичок працювати з масивами при

програмуванні на Асемблері.

1. **Теоретичні відомості**

При описі масиву вказується кількість елементів у ньому та їх тип:

**x dw 30 dup(?),**

проте не вказується як нумеруються (індексуються) його елементи.

Описанню може відповідати масив:

x [0..29] x [1..30] x [k..29+k]

Автор програми може накладати на масив різноманітні діапазони зміни індексу. Іноді діапазон індексів суворо визначено в умові задачі. Якщо нумерація строго не визначена, тоді краще почати нумерацію з нуля.

Як залежить адреса елемента від індексу цього елемента:

x dw 30 dup(?); x [k..29+k] =>

адреси (x[i])=x+2\*(i-k)

адреси (x[i])=x+(type x)\*(i-k)

при k =0

адреси (x[i])=x+(type x)\*i

Тому зазвичай вважають , що елементи в асемблері нумеруються з 0:

x dw 30 dup(?);

x [0..29]

Для багатомірних масивів ситуація аналогічна:

А [ N \* M ]

N,M - const

N - кількість рядків

M - кількість стовбців

Припустимо, що елементи матриці розміщені в пам'яті по рядках (можна розміщувати й по стовбцям):

адреси (A [i,j])=A+M\*(type A)\*(i-k\*M)+(type A)\*(i-k\*N)

Найбільш доступний вигляд ця залежність приймає при нумерації з нуля. Як здійснюється доступ до елементів масиву? До цього моменту ми працювали з командами, у яких для операндів вказувалися їх точні адреси.

**Приклад 1.** Розрахунок суми елементів масиву.

prog segment

assume cs:prog, ds:prog

data\_array dw 10 dup(?)

sum dw ?

start:

mov ax,cs

mov ds,ax

xor ax,ax

mov bx,ax

mov cx,10

next:

add ax,data\_array [bx]

add bx,2

loop next

mov sum,ax

ret

prog ends

end start

Проте команда може працювати й з виконавчою (ефективною) адресою:

А вик = (А+[рег]) mod 2^16, де

[рег] - зміст регістра

Спершу ніж виконати команду, центральний процесор додасть до адреси А (вказаній у команді) поточний зміст регістра - отримає нову адресу й буде працювати з операндом за цією адресою.

**Команда LEA.** При використанні регістрів-модифікаторів часто доводиться записувати в них ті чи інші адреси. Нехай, наприклад, нам необхідно занести в регістр ВХ адресі змінної X:

X DW 88

Щоб зробити це, можна завести ще одну змінну, значенням якої буде адреса X:

Y DW X

а потім передати значення цієї змінної в регістр ВХ:

MOV BX,Y

Але зрозуміло, що це дещо штучний спосіб завантаження адреси в регістр. Тому в систему команд персонального комп’ютера уведена спеціальна команда такого завантаження:

Завантаження виконуючої адреси (load effective address): LEA r16,A

Ця команда розраховує виконуючу адресу іншого операнда й заносить її в регістр r16: r16:=Авик. Прапорці команда не змінює. У якості першого операнда може бути вказаний будь-який регістр загального призначення, а в якості іншого - будь-який адресний вираз (з модифікаторами чи без них).

Одним з прикладів, де корисна команда LEA, є вивід рядка по операції OUTSTR. Ця операція, нагадаємо, потребує, щоб початкова адреса рядка, що буде виведена знаходилася в регістрі DX. Це доцільно робити по команді LEA:

S DВ 'a+Ь=с','$'

…

LEA DX,S ; DX:=адреси S

OUTSTR ; буде виведено: а+Ь=з

Розглянемо особливості команди LEA. При цьому будемо вважати, що в програмі є наступні описання:

Q DW 45 R DW -8

По-перше відмітимо, що команда LEA дуже схожа на команду MOV, але між ними є й принципова відмінність: якщо LEA заносить в регістр саму адресу, вказану в команді, то MOV заносить вміст комірки з цією адресою:

LEA BX,Q; BX:=адреси Q

MOV BX,Q; BX:=вміст Q (-45)

У команді LEA другий операнд може бути будь-якою адресою - і адресою байта, і адресою слова й т.д. Проте в якості цього операнда неможна вказати константний вираз чи ім'я регістра:

LEA CX.88 ; помилка LEA CX,BX ; помилка

Якщо в якості іншого операнда вказано модифікатор адреси, то спочатку розраховується виконуюча адреса й лише потім відбувається завантаження в регістр: mov si,2

LEA AX,Q[SI]; АХ:=Авик-Q+[SI]=адреси(Q+2)=адреси(R)

Цим можна скористатися для передачі в який-небудь регістр значення регістра-модифікатора, збільшеного чи зменшеного на деяке число:

MOV BX,50

LEA CX,[BX+2] ;CX:=[BX]+2-50+2=52

LEA DI,[DI-3] ;DI:=DI-3 (SUB DI,3)

1. **Текст програми**

Програма розроблена на мові програмування C++.

Варіант 17:

17. A [5,3]. Знайти номер рядка, у якому знаходиться мінімальний елемент.

#include <iostream>

int main()

{

std::cout << "Input elements like \"X Y Z\"\n";

int a [5][3];

for (int i = 0; i < 5; i++) {

int x, y, z;

std::cin>>x>>y>>z;

a[i][0] = x;

a[i][1] = y;

a[i][2] = z;

}

int min, row, rowMinObj = 0;

\_asm{

xor ebx, ebx

xor eax, eax

mov esi, 0

mov row, 0

lea ebx, a

mov eax, [ebx][esi]

mov min, eax

jmp m1

m: add bx, 12

mov si, 0

add row, 1

cmp row, 5

je stop

m1: mov eax, [ebx][esi]

cmp eax, min

jge m2

mov min, eax

mov eax, row

mov rowMinObj, eax

m2: add esi, 4

cmp esi, 12

jge m

jmp m1

stop:

add rowMinObj, 1

}

std::cout << "Min is: " << min << " Row is: " << rowMinObj;

}

1. **Результати роботи програми**

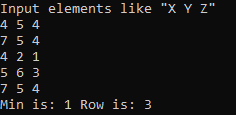


Рисунок 1. Результат роботи

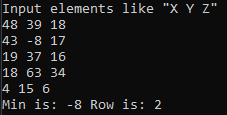


Рисунок 2. Результат роботи з від’ємним числом

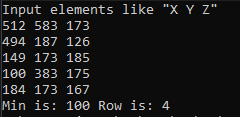


Рисунок 4. Результат

1. **Висновки**

На даній лабораторній роботі було ознайомлено з роботою із масивами на Асемблері. Та покращено знання використання асемблерної вставки на мові програмування C++.

1. **Відповіді на контрольні питання**
2. Вкажіть, як нумеруються елементи масиву?

Автор програми може накладати на масив різноманітні діапазони зміни індексу. Іноді діапазон індексів суворо визначено в умові задачі. Якщо нумерація строго не визначена, тоді краще почати нумерацію з нуля.

1. Вкажіть, як залежить адреса елемента масиву від індексу цього елементу?

x dw 30 dup(?); x [k..29+k] =>

адреси (x[i])=x+2\*(i-k)

адреси (x[i])=x+(type x)\*(i-k)

при k =0

адреси (x[i])=x+(type x)\*i

Тому зазвичай вважають , що елементи в асемблері нумеруються з 0:

x dw 30 dup(?);

x [0..29]

Для багатомірних масивів ситуація аналогічна:

А [ N \* M ]

N,M - const

N - кількість рядків

M - кількість стовбців

Припустимо, що елементи матриці розміщені в пам'яті по рядках (можна розміщувати й по стовбцям):

адреси (A [i,j])=A+M\*(type A)\*(i-k\*M)+(type A)\*(i-k\*N)

Найбільш доступний вигляд ця залежність приймає при нумерації з нуля.

1. Вкажіть, як здійснюється доступ до елементів масиву?

За допомогою виконавчої (ефективної) адреси:

А вик = (А+[рег]) mod 2^16, де

[рег] - зміст регістра

Спершу ніж виконати команду, центральний процесор додасть до адреси А (вказаній у команді) поточний зміст регістра - отримає нову адресу й буде працювати з операндом за цією адресою.

Команда LEA. При використанні регістрів-модифікаторів часто доводиться записувати в них ті чи інші адреси.