ЛАБОРАТОРНА РОБОТА № 7

ОРГАНІЗАЦІЯ ЦИКЛІВ І РОБОТА З ЦІЛОЧИСЛЕНИМИ МАСИВАМИ

Мета: ознайомитися з основними командами мови Assembler для організації циклів і роботи з цілочисельними масивами; набути практичних навичок в написанні програм з використанням циклів та масивів на мові Assembler

Хід роботи:

Завдання 1: . Написати програму для обробки одномірного масиву для початкових даних(табл.7.5) в знаковому форматі. Виконати покрокове виконання асемблерного коду та навести значення регістрів при їх виконанні. Відмітити нормальні та аномальні результати, зробити аналіз результатів.

15	Знайти суму квадратів всіх від'ємних елементів масиву A={a[i]}, які задовольняють умові c<=a[i]<=d. Тип даних SHORTINT.

Лістинг програми:

```
#include <stdio.h>
#include <locale.h>
#include<time.h>
#include<stdlib.h>
#include<math.h>
#define MAX_N 10

int main() {
	short int a[MAX_N],c=0,d=0, summ = 0,mul;
	printf("c<=a[i]<=d\n\n");
	do {
		printf("Enter the values of the range [-32768...32767]:\n");
			printf("c = "); scanf_s("%hi", &c);
		printf("d = "); scanf_s("%hi", &d);
		if (c >= d)
		{
			printf("c can not be greater or equal d! Enter values again.\n\n");
		}
	} while (c >= d);
	int n = MAX_N;
	short int res = 0;
	for (int i = 0; i < n; i++)
	{
		a[i] = rand() % 10 - 5;
		if (a[i]<0 && (a[i] > c && a[i] < d) )
		{
			mul = pow(a[i], 2);
```

					ЖДТУ.18.125.15.000 – Лр7		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Макарчук В.В			Звіт з лабораторної роботи №7	Літ.	Арк. 1 / Аркушів
Перевір.		Байлюк Є.М					
Керівник						ФІКТ Гр. КБ-2[1]	
Н. контр.							
Зав. каф.							

```
                summ += mul;
            }
            printf("A[%hi] = %hi\n", i, a[i]);
        }




        printf("\n%hi", summ);

        _asm
        {
            mov ax, c; // ax = c
            mov bx, d; // bx = d
            mov ecx, n // <ecx> = n
            mov dx, 0 // <dx> = 0
            dec ecx // зменшуємо значення в регістрі <ecx> на 1
            cycle : // цикл cycle
            shl ecx, 1 // зсув вліво на 1 розряд
            mov si, a[ecx] // <si> = a[ecx]
            cmp si, 0 // порівнюємо значення регістра <si> з 0
            jl exit1 // якщо менше - перейти до циклу exit1
            jmp exit4;


            exit1:
            cmp si, ax // порівнюємо значення регістра <si> з значенням  в регістрі <ax>
            jg exit2 // якщо менше - перейти до циклу exit1
            jmp exit4;

            exit2:
            cmp si, bx // порівнюємо значення регістра <si> з значенням  в регістрі <bx>
            jl exit3 // якщо більше - перейти до циклу exit1
            jmp exit4;
            exit3:
            imul si, si;
            add dx, si;

            exit4 : // цикл exit1
            shr ecx, 1 // зсув вправо на 1 розряд
            dec ecx // зменшуємо значення в регістрі <ecx> на 1
            cmp ecx, 0 // порівнюємо значення регістра <ecx> з 0
            jnl cycle // поки не менше - перейти до циклу cycle

            mov res, dx // res = <dx>



        }


        if (res > 32767 || res < -32768)
        {
            printf("Overflow!\n");
        }
        else
        {
            printf("Result = %hi\n", res);
        }
        system("pause");
        return 0;
}
```

Команда	Значення регістра				EFLAGS/FLAGS
	ax	bx	ecx	dx	
mov ax, c	-5	н/в	н/в	н/в	-
mov bx, d	н/в	10	н/в	н/в	-
Mov ecx,n	н/в	н/в	10	н/в	-
mov dx, 0	н/в	н/в	н/в	0	-
dec ecx	н/в	н/в	9	н/в	-
cycle	Мітка циклу cycle(код буде виконуватись в цій мітці, якщо умова – істина)				
shl ecx, 1	Зсув біта операнда вліво на 1 розряд				
mov si, a[ecx]	Заносимо в регістр елемент масиву з відповідним індексом				
cmp si, 0	Крок масиву порівнюється з 0				
jl exit1	Порівнюємо елемент масиву з нижньою границею – c				
exit1	Мітка циклу exit1(код буде виконуватись в цій мітці, якщо умова – істина				
cmp si, ax	Порівнюємо елемент масиву з верхньою границею – d				
jg exit2					
exit2:	Мітка циклу exit1(код буде виконуватись в цій мітці, якщо умова – істина				
cmp si, bx	Порівнюємо елемент масиву з верхньою границею – d				
jl exit3					
exit3:	Мітка циклу exit1(код буде виконуватись в цій мітці, якщо умова – істина				
imul si, si;	Множим элемент сам на себя				
add dx, si;	Добавляем полученый квадрат				
exit4	Мітка циклу exit1(код буде виконуватись в цій мітці, якщо умова – істина				
shr ecx, 1	Зсув біта операнда вправо на 1 розряд				
dec ecx	Зменшуємо значення в регістрі на 1				
cmp ecx, 0	Порівнюємо значення регістра з 0				
jnl cycle	Перейти на мітку циклу, якщо умова є неві				
mov res, dx	н/в	н/в	н/в	28	

Рис. 1. Результат виконання програми

```
c<=a[i]<=d

Enter the values of the range [-32768...32767
c = -5
d = 10
A[0] = -4
A[1] = 2
A[2] = -1
A[3] = -5
A[4] = 4
A[5] = -1
A[6] = 3
A[7] = 3
A[8] = -3
A[9] = -1

28Result = 28
Press any key to continue . . .
```

Завдання 2: Написати програму для обробки двовимірного масиву для початкових даних(табл.7.6) в знаковому форматі. Виконати покрокове виконання асемблерного коду та навести значення регістрів при їх виконанні. Відмітити нормальні та аномальні результати, зробити аналіз результатів.

15	Знайти суму квадратів всіх від'ємних елементів масиву A={a[i][j]}, які задовольняють умові c<=a[i][j]<=d. Тип даних **SHORTINT.**

Лістинг програми:

```
#include <stdio.h>
#include <locale.h>
#include<time.h>
#include<stdlib.h>
#include<math.h>
#define MAX_N 5

int main() {
	short int a[MAX_N][MAX_N], c = 0, d = 0, summ = 0, mul,tmp=0;
	printf("c<=a[i][j]<=d\n\n");
	do {
		printf("Enter the values of the range [-32768...32767]:\n");
		printf("c = "); scanf_s("%hi", &c);
		printf("d = "); scanf_s("%hi", &d);
		if (c >= d)
		{
			printf("c can not be greater or equal d! Enter values again.\n\n");
		}
	} while (c >= d);
	int n = MAX_N * MAX_N;
	short int res = 0;
	for (int i = 0; i < MAX_N; i++)
```

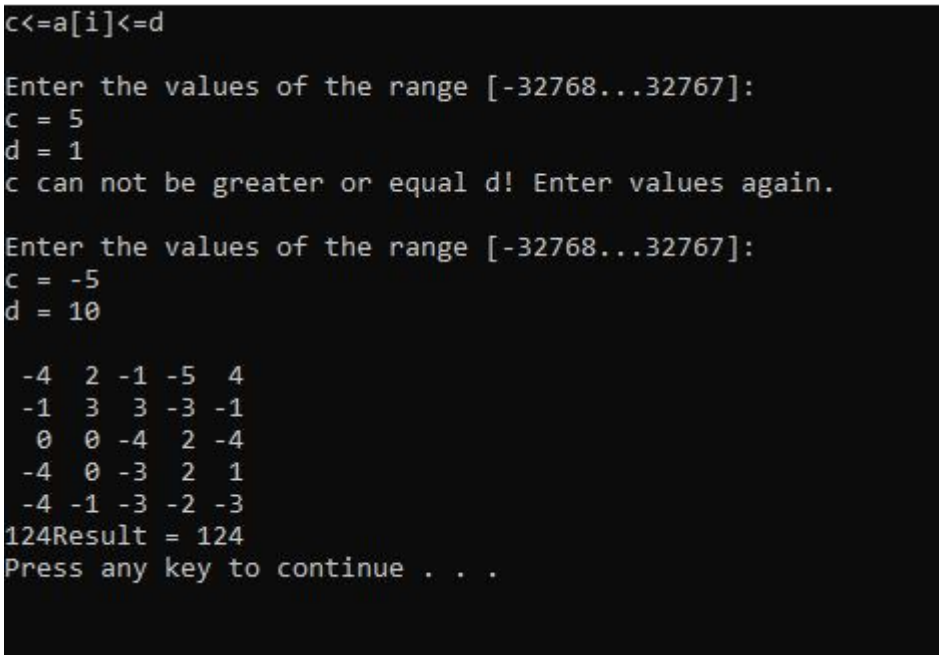
```
	{

		printf("\n");
			for (int j = 0; j < MAX_N; j++)
			{

				a[i][j] = rand() % 10 - 5;
				if (a[i][j] < 0 && (a[i][j] > c && a[i][j] < d))
				{
					mul = pow(a[i][j], 2);
					summ += mul;
				}
				printf("%3hi", a[i][j]);

			}



	}
	printf("\n%hi", summ);

	_asm
	{
		mov ecx, n // <ecx> = n
		lea si, a // завантаження зміщення масиву в регістр   SI(замінює shr ecx, 1 та
shl ecx, 1)
		cycle: // цикл cycle
		lodsw // завантажуємо слово з пам'яті, на який вказує регістр SI
			mov dx, d // <dx> = d
			mov bx, c // <bx> = c
			cmp ax, bx // порівнюємо з нижньою границею - c
			jl next // якщо менше - перейти до циклу next

			cmp ax, dx // порівнюємо з верхньою границею - d
			jg next // якщо більше - перейти до циклу next

			cmp ax, 0;
			jl exit1;
			jmp next;

		exit1:
			cmp ax, bx;
			jg exit2;
			jmp next;

		exit2:
			cmp ax, dx;
			jl exit3 // якщо більше - перейти до циклу exit1
			jmp next;
		exit3:
			imul ax, ax;
			add tmp, ax;
		next: // цикл next
			dec ecx // зменшуємо к-ть чисел на 1
			cmp ecx, 0 // порівнюємо к-ть чисел з 0
			jnl cycle // якщо не менше - перейти до циклу cycle
			mov cx, tmp;
			mov res, cx;
	}
	if (res > 32767 || res < -32768)
	{
		printf("Overflow!\n");
	}
	else
```

```
	{
		printf("Result = %hi\n", res);
	}
	system("pause");
	return 0;
}
```

Команда	Значення регістра				EFLAGS/FLAGS	
	ax	bx	ecx	dx	cx	
mov ecx	н/в	н/в	25	н/в	н/в	-
lea si, a	Беремо в регістр адресу першого елемента масиву -					
Mov ecx,n	н/в	н/в	10	н/в	-	
cycle	Мітка циклу cycle(код буде виконуватись в цій мітці, якщо умова – істина)					
lodsw	Завантажуємо слово з пам'яті, на який вказує регістр SI					
mov dx, d	н/в	н/в	н/в	10		
mov bx, c	н/в	-5	н/в	н/в		
cmp ax, bx	Порівнюємо елемент масиву з нижньою границею – c					
jl next	Якщо менше – перейти до мітки циклу next					
cmp ax, dx	Порівнюємо елемент масиву з верхньою границею – d					
jg next	Мітка циклу exit1(код буде виконуватись в цій мітці, якщо умова – істина					
cmp ax, 0	Порівнюємо елемент масиву з верхньою границею – d					
jl exit1;						
exit1:	Мітка циклу exit1(код буде виконуватись в цій мітці, якщо умова – істина					
cmp si, bx	Порівнюємо елемент масиву з верхньою границею – d					
cmp ax, bx;						
jg exit2;	Мітка циклу exit1(код буде виконуватись в цій мітці, якщо умова – істина					
exit2:	Множим элемент сам на себя					
cmp ax, dx	Добавляем полученый квадрат					
jl exit3	Мітка циклу exit1(код буде виконуватись в цій мітці, якщо умова – істина					
exit3:	Зсув біта операнда вправо на 1 розряд					
imul ax, ax;	Зменшуємо значення в регістрі на 1					
add tmp, ax;	Порівнюємо значення регістра з 0					
next	Мітка циклу next(код буде виконуватись в цій мітці, якщо умова – істина)					
dec ecx	н/в	н/в	24	н/в		
cmp ecx, 0	Порівнюємо значення регістра з 0					
jnl cycle	Поки індекс елементу масиву не менше 0 – перейти до циклу cycle					

mov cx, tmp;					124	
mov res, cx;					124	

Результат виконання Рис 2



Висновки:. Під час роботи ознайомилися з основними командами мови Assembler для організації циклів і роботи з цілочисельними масивами; набули практичних навичок в написанні програм з використанням циклів та масивів на мові Assembler.