МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

ІКНІ
Кафедра **ПЗ**

ЗВІТ

до лабораторної роботи № 5
на тему: "*Складення та відлагодження циклічної програми мовою асемблера мікропроцесорів x86 для Windows*"
з дисципліни: "Архітектура комп'ютера"

Лектор:
доцент кафедри ПЗ
Крук О.Г.

Виконав:
студент групи ПЗ-22
Коваленко Д.М.

Прийняв:
доцент кафедри ПЗ
Крук О.Г.

«_____» _____ 2022 р.
$\sum$ = _____.....

Львів — 2022

Тема. Складення та відлагодження циклічної програми мовою асемблера мікропроцесорів x86 для Windows.

Мета. Ознайомитись на прикладі циклічної програми з основними командами асемблера; розвинути навики складання програми з вкладеними циклами; відтранслювати і виконати в режимі відлагодження програму, складену відповідно до свого варіанту; перевірити виконання тесту.

Індивідуальне завдання

Завдання для **ПЗ-22**

Варіант	Розмір матриці (n × m)	Операції оброблення матриці	b	c	Умова*
1	(8 × 7)	1. Обчисліть скалярний добуток 3-го і 4-го стовпців. 2. Обчисліть кількість і суму елементів 7-го рядка, які задовільняють вказаній умові.	-37	69	$b <= a_i < c$
2	(6 × 8)	1. Обчисліть скалярний добуток 2-го і 5-го рядків. 2. Обчисліть кількість і суму елементів 2-го стовпця, які задовільняють вказаній умові.	-42	77	$b < a_i <= c$
3	(7 × 8)	1. Обчисліть скалярний добуток 1-го і 3-го стовпців. 2. Обчисліть кількість і суму елементів 6-го рядка, які задовільняють вказаній умові.	-51	82	$b <= a_i <= c$
4	(6 × 9)	1. Обчисліть скалярний добуток 3-го і 5-го рядків. 2. Обчисліть кількість і суму елементів 9-го стовпця, які задовільняють вказаній умові.	-67	94	$b < a_i < c$
5	(8 × 6)	1. Обчисліть скалярний добуток 3-го і 4-го стовпців. 2. Обчисліть кількість і суму елементів 8-го рядка, які задовільняють вказаній умові.	-29	48	$b < a_i <= c$
6	(6 × 8)	1. Обчисліть скалярний добуток 1-го і 4-го рядків. 2. Обчисліть кількість і суму елементів 5-го стовпця, які задовільняють вказаній умові.	-35	55	$a_i <= b$ або $a_i > c$
7	(8 × 7)	1. Обчисліть скалярний добуток 2-го і 7-го стовпців. 2. Обчисліть кількість і суму елементів 4-го рядка, які задовільняють вказаній умові.	-43	60	$a_i < b$ або $a_i >= c$
8	(7 × 8)	1. Обчисліть скалярний добуток 5-го і 3-го рядків. 2. Обчисліть кількість і суму елементів 3-го стовпця, які задовільняють вказаній умові.	-29	83	$a_i <= b$ або $a_i >= c$
9	(8 × 7)	1. Обчисліть скалярний добуток 7-го і 4-го стовпців. 2.	-46	72	$a_i < b$ або $a_i > c$

Хід роботи

Програма 1

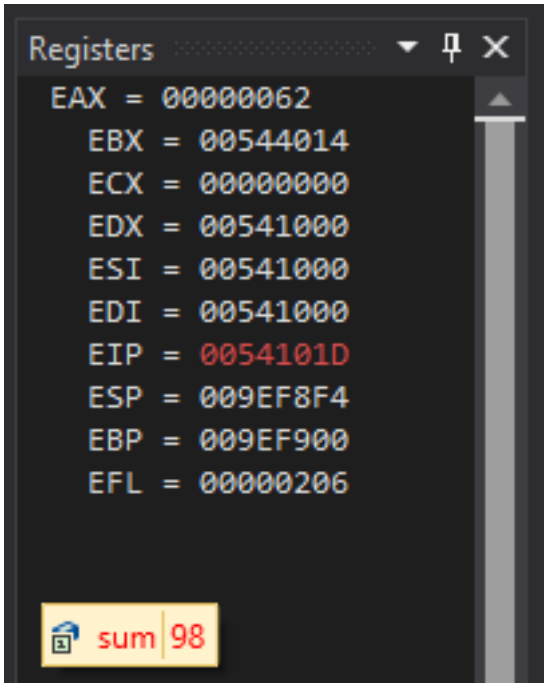

Рис. 1: Стан регістрів та змінної *sum* після виконання програми

$$17 + 3 - 51 + 242 - 113 = 98_{10} = 62_{16}$$

Програма 2

3 рядок: $a = -81, -78, -82, -39, -90, -78, 24$

5 рядок: $b = 56, -19, -86, 34, -83, -99, -31$

Скалярний добуток: $\sum_{i=1}^{7} a_i b_i = -81 \cdot 56 - 78 \cdot (-19) - 82 \cdot (-86) - 39 \cdot 34 - 90 \cdot (-83) - 78 \cdot (-99) + 24 \cdot (-31) =$

$$= -4536 + 1482 + 7052 - 1326 + 7470 + 7722 - 744 = 17120$$

```
 10, 64,-94, 77, 99, 18, 52
-23,-77,-45, 65, 77, 66,-24
-81,-78,-82,-39,-90,-78, 24
-18,-64,-74,-28,-16,-40, 91
 56,-19,-86, 34,-83,-99,-31
-70,-58, 13, 98, 90, 46,-77
 97, 85,-10, 57, 88, 99,-26
-11, 69, 32, 42,-51, 37,-51
```

Рис. 2: Двовимірний масив, що необхідно було транспонувати

+10	-23	-81	-18	+56	-70	+97	-11
+64	-77	-78	-64	-19	-58	+85	+69
-94	-45	-82	-74	-86	+13	-10	+32
+77	+65	-39	-28	+34	+98	+57	+42
+99	+77	-90	-16	-83	+90	+88	-51
+18	+66	-78	-40	-99	+46	+99	+37
+52	-24	+24	+91	-31	-77	-26	-51

Рис. 3: Відображення транспонованого масиву у пам'яті

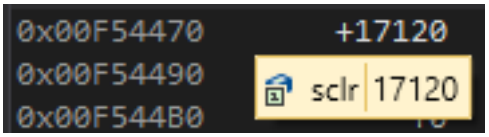

Рис. 4: Результат обчислення скалярного добутку

Стовпець 3: $-94, -45, -82, -74, 86, 13, -10, 95$
Кількість елементів, що задовільняють умову: 6
Сума елементів, що задовільняють умову: -114

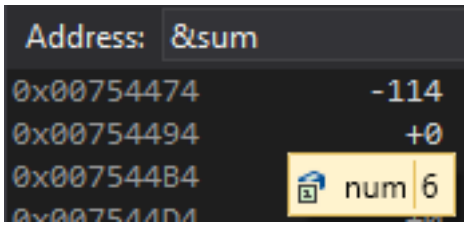

Рис. 5: Результат знаходження кількості та суми за заданою умовою

Код програми 2

DO NOT COPY *(pls)*

```
.586P
.MODEL FLAT, STDCALL

_DATA SEGMENT
tmp     DD 0
col     DD 7
row     DD 8
arr     DD  10, 64,-94, 77, 99, 18, 52
        DD -23,-77,-45, 65, 77, 66,-24
        DD -81,-78,-82,-39,-90,-78, 24
        DD -18,-64,-74,-28,-16,-40, 91
        DD  56,-19, 86, 34,-83,-99,-31
        DD -70,-58, 13, 98, 90, 46,-77
        DD  97, 85,-10, 57, 88, 99,-26
        DD -11, 69, 95, 42,-51, 37,-51
res     DD 224 DUP(?)   ; not initialized 4*7*8 bytes
sclr    DD 0
num     DD 0
sum     DD 0
_DATA ENDS

_TEXT SEGMENT
START:

T1:
```

```
    lea ECX, arr
    lea EDX, res
    mov EBX, 0

L1:             ; EAX = 0..row
    mov EAX, 0

L2:             ; EBX = 0..col
    lea ECX, arr    ; ECX = &arr
    lea EDX, res    ; EDX = &res

    mov tmp, EAX    ; arr += 4*(EAX*col+EBX)
    push EAX
    push EDX
    push ECX
    mov EAX, tmp
    mul col
    add EAX, EBX
    push EBX
    mov EBX, 4
    mul EBX
    pop EBX
    pop ECX
    add ECX, EAX
    pop EDX
    pop EAX

    mov tmp, EAX    ; res += 4*(EBX*row+EAX)
    push EAX
    push ECX
    push EDX
    mov EAX, EBX
    mul row
    add EAX, tmp
    push EBX
    mov EBX, 4
    mul EBX
    pop EBX
    pop EDX
    add EDX, EAX
    pop ECX
    pop EAX

    push EAX        ; *EDX = *ECX
    mov EAX, [ECX]
    mov [EDX], EAX
    pop EAX

    inc EAX         ; EAX += 1; if (EAX < row) goto L2;
    cmp EAX, row
    jl L2

    inc EBX         ; EBX += 1; if (EBX < col) goto L1;
    cmp EBX, col
    jl L1

T2:
    lea EDX, arr    ; row 3
    add EDX, 56
    mov EBX, EDX

    lea EDX, arr    ; row 5
    add EDX, 112
```

```
    mov ECX, EDX
    mov EDX, 0

L3:             ; EDX = 0..col
    mov EAX, [EBX]    ; sclr += (*EBX)*(*ECX)
    push EDX
    mov EDX, [ECX]
    imul EDX
    pop EDX
    add sclr, EAX

    add EBX, 4
    add ECX, 4

    inc EDX         ; EDX += 1; if (EDX < col) goto L1;
    cmp EDX, col
    jl L3

T3:             ; a <= -29 or a >= 83
    lea EAX, arr    ; EAX = arr[3]
    add EAX, 8
    mov EDX, 0

L4:
    cmp EDX, row    ; if (EDX >= row) goto T4;
    jge T4

    inc EDX

    mov EBX, [EAX]    ; if (EBX <= -29 || EBX >= 83) goto COUNT;
    cmp EBX, -29
    jle COUNT
    cmp EBX, 83
    jge COUNT
    add EAX, 28
    jmp L4

COUNT:
    inc num
    add EAX, 28
    jmp L4

T4:
    lea EAX, arr    ; EAX = arr[3]
    add EAX, 8
    mov EDX, 0

L5:
    cmp EDX, row    ; if (EDX >= row) goto DONE;
    jge DONE

    inc EDX

    mov EBX, [EAX]    ; if (EBX <= -29 || EBX >= 83) goto SUMUP;
    cmp EBX, -29
    jle SUMUP
    cmp EBX, 83
    jge SUMUP
    add EAX, 28
    jmp L5

SUMUP:
    add sum, EBX
```

```
        add EAX, 28
        jmp L5

DONE:

RET
_TEXT ENDS
END START
```

Висновки

Під час виконання лабораторної роботи я ознайомився на прикладі циклічної програми з основними командами асемблера; розвинув навики складання програми з вкладеними циклами; відтранслював і виконати в режимі відлагодження програму, складену відповідно до свого варіанту; перевірив виконання тесту.