МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

факультет програмної інженерії та бізнесу

кафедра інженерії програмного забезпечення

**Лабораторна робота №6**

з дисципліни « Програмування на асемблері »

на тему: «РЯДКОВІ ІНСТРУКЦІЇ І МАСИВИ»

Виконав: студент 2 курсу групи № 622п

121 «Інженерія програмного забезпечення»

(код спеціальності)

Зайченко Я. І.

(ПІБ студента)

Прийняв: ст. викладач каф. 603

\_ Дем’яненко В. А.\_\_\_\_\_\_

(ПІБ викладача)

Національна шкала:

Кількість балів:

Харків – 2024

**Мета роботи:** вивчити команди переміщення даних, команди повторення строкових інструкцій; вивчити організацію масивів мовою Асемблер.

Структура звіту

1. написання коду програми з ім'ям *Lab6.asm.*
2. процес створення виконуваного файлу та процес відладки;
3. код програми з коментарями про зміст регістрів усіх рядків у сегменті коду.
4. написання іншої програми й стеження за її роботою. Записаний результат роботи програми.
5. висновок.

Виконання роботи

Створюємо новий asm файл з кодом поданим нижче:

;Задано два масиви ArrayA, ArrayB, що складаються з 10 елементів кожен.

;Порівняти ці масиви поелементно, і якщо елементи є ;однаковими, то записати у відповідний елемент ;третього масиву (Difference) 'Y', якщо різними – то 'N'

;Знайти суму й кількість всіх однакових і різних

;елементів двох масивів (ArrayA, ArrayB)

.model tiny

.stack 100h

.data

ArrayA db 05,10,06,44,20,32,05,11,46,0

ArrayB db 35,10,15,44,20,02,65,10,46,0

Difference db 10 dup (0)

NumOfDiff dw 0

NumOfEqual dw 0

SumOfDiff dd 0

SumOfEqual dd 0

.code start:

mov ax,@data

mov ds,ax

push ds

pop es

mov di,offset Difference

mov cx,10

mov al,'Y'

cld

rep stosb

mov si,offset ArrayA

mov di,offset ArrayB

mov bx,offset Difference

mov cx,10

cld

findDE:

cmpsb

jne NotEqual

inc NumOfEqual

inc bx

dec di

dec si

mov al,byte ptr ds:[si]

cbw

add SumOfEqual, ax

mov al,byte ptr ds:[di]

cbw

add SumOfEqual, ax

inc si

inc di

jmp NextElement

NotEqual:

inc NumOfDiff

mov byte ptr ds:[bx],'N'

inc bx

dec di

dec si

mov al,byte ptr ds:[si]

cbw

add SumOfDiff, ax

mov al,byte ptr ds:[di]

cbw

add SumOfDiff, ax

inc si

inc di

NextElement:

loop findDE

mov ax,4c00h

int 21h

end start

Для відладки було використано емулятор emu8086, результат відладки показано на рисунку 1

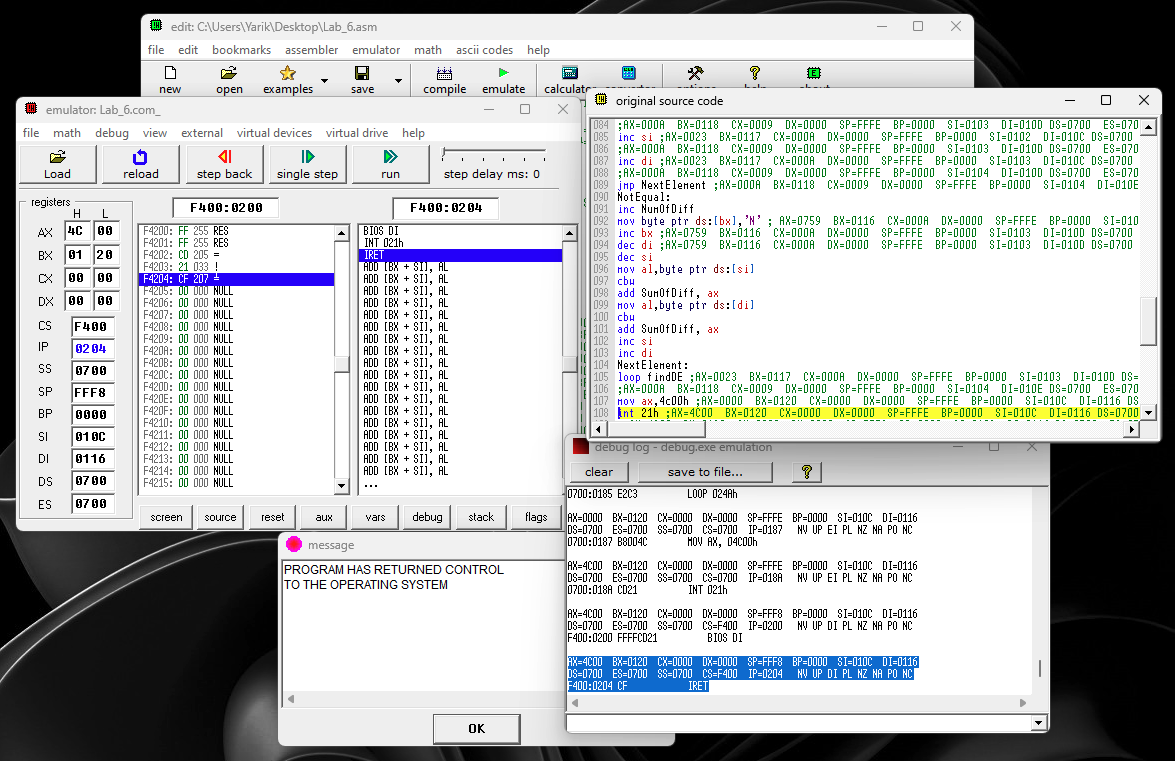


Рисунок 1 – відладка коду на асемблері

Програмний застосунок emu8086 має в собі шаблони виконуваних файлів \*.com, \*.exe. в нашому випадку будемо створювати \*.сom файл.

При відладці записано значення регістрів усіх рядків в сегменті коду.

.model tiny

.stack 100h

.data ;AX=0000 BX=0000 CX=008C DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000 DS=0700 ES=0700 SS=0700 CS=0700 IP=0100

ArrayA db 05,10,06,44,20,32,05,11,46,0

ArrayB db 35,10,15,44,20,02,65,10,46,0

Difference db 10 dup (0)

NumOfDiff dw 0

NumOfEqual dw 0

SumOfDiff dd 0

SumOfEqual dd 0

.code

start:

mov ax,@data ;AX=0000 BX=0000 CX=008C DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000 DS=0700 ES=0700 SS=0700 CS=0700 IP=012C

mov ds,ax ;AX=0700 BX=0000 CX=008C DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000 DS=0700 ES=0700 SS=0700 CS=0700 IP=012E

push ds ;X=0700 BX=0000 CX=008C DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000 DS=0700 ES=0700 SS=0700 CS=0700 IP=0130

pop es ;AX=0700 BX=0000 CX=008C DX=0000 SP=FFFC BP=0000 SI=0000 DI=0000 DS=0700 ES=0700 SS=0700 CS=0700 IP=0131

mov di,offset Difference ;AX=0700 BX=0000 CX=008C DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000 DS=0700 ES=0700 SS=0700 CS=0700 IP=0132

mov cx,10 ;AX=0700 BX=0000 CX=008C DX=0000 SP=FFFE BP=0000 SI=0000 DI=0116 DS=0700 ES=0700 SS=0700 CS=0700 IP=0135

mov al,'Y' ;AX=0700 BX=0000 CX=000A DX=0000 SP=FFFE BP=0000 SI=0000 DI=0116 DS=0700 ES=0700 SS=0700 CS=0700 IP=0138

cld ;AX=0759 BX=0000 CX=000A DX=0000 SP=FFFE BP=0000 SI=0000 DI=0116 DS=0700 ES=0700 SS=0700 CS=0700 IP=013A

rep ;AX=0759 BX=0000 CX=000A DX=0000 SP=FFFE BP=0000 SI=0000 DI=0116 DS=0700 ES=0700 SS=0700 CS=0700 IP=013B

stosb ;AX=0759 BX=0000 CX=000A DX=0000 SP=FFFE BP=0000 SI=0000 DI=0116 DS=0700 ES=0700 SS=0700 CS=0700 IP=013C

; rep2 AX=0759 BX=0000 CX=0009 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0117 DS=0700 ES=0700 SS=0700 CS=0700 IP=013B

;stosb2 AX=0759 BX=0000 CX=0009 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0117 DS=0700 ES=0700 SS=0700 CS=0700 IP=013C

;rep3 AX=0759 BX=0000 CX=0008 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0118 DS=0700 ES=0700 SS=0700 CS=0700 IP=013B

;stosb3 AX=0759 BX=0000 CX=0008 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0118 DS=0700 ES=0700 SS=0700 CS=0700 IP=013C

;rep4 AX=0759 BX=0000 CX=0007 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0119 DS=0700 ES=0700 SS=0700 CS=0700 IP=013B

;stosb4 AX=0759 BX=0000 CX=0007 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0119 DS=0700 ES=0700 SS=0700 CS=0700 IP=013C

;rep5 X=0759 BX=0000 CX=0006 DX=0000 SP=FFFE BP=0000 SI=0000 DI=011A DS=0700 ES=0700 SS=0700 CS=0700 IP=013B

;stosb5 AX=0759 BX=0000 CX=0006 DX=0000 SP=FFFE BP=0000 SI=0000 DI=011A DS=0700 ES=0700 SS=0700 CS=0700 IP=013C

;rep6 AX=0759 BX=0000 CX=0005 DX=0000 SP=FFFE BP=0000 SI=0000 DI=011B DS=0700 ES=0700 SS=0700 CS=0700

;stosb6 AX=0759 BX=0000 CX=0005 DX=0000 SP=FFFE BP=0000 SI=0000 DI=011B DS=0700 ES=0700 SS=0700 CS=0700 IP=013C

;rep7 AX=0759 BX=0000 CX=0004 DX=0000 SP=FFFE BP=0000 SI=0000 DI=011C DS=0700 ES=0700 SS=0700 CS=0700 IP=013B

;stosb7 AX=0759 BX=0000 CX=0004 DX=0000 SP=FFFE BP=0000 SI=0000 DI=011C DS=0700 ES=0700 SS=0700 CS=0700 IP=013C

;rep8 AX=0759 BX=0000 CX=0003 DX=0000 SP=FFFE BP=0000 SI=0000 DI=011D DS=0700 ES=0700 SS=0700 CS=0700 IP=013B

;stosb8 AX=0759 BX=0000 CX=0003 DX=0000 SP=FFFE BP=0000 SI=0000 DI=011D DS=0700 ES=0700 SS=0700 CS=0700 IP=013C

;rep9 AX=0759 BX=0000 CX=0002 DX=0000 SP=FFFE BP=0000 SI=0000 DI=011E DS=0700 ES=0700 SS=0700 CS=0700 IP=013B

;stosb9 AX=0759 BX=0000 CX=0002 DX=0000 SP=FFFE BP=0000 SI=0000 DI=011E DS=0700 ES=0700 SS=0700 CS=0700 IP=013C

;rep10 AX=0759 BX=0000 CX=0001 DX=0000 SP=FFFE BP=0000 SI=0000 DI=011F DS=0700 ES=0700 SS=0700 CS=0700 IP=013B

;stosb10 AX=0759 BX=0000 CX=0001 DX=0000 SP=FFFE BP=0000 SI=0000 DI=011F DS=0700 ES=0700 SS=0700 CS=0700 IP=013C

;rep11 AX=0759 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0120 DS=0700 ES=0700 SS=0700 CS=0700 IP=013B

;stosb11 AX=0759 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0120 DS=0700 ES=0700 SS=0700 CS=0700 IP=013C

mov si,offset ArrayA ;AX=0759 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0120 DS=0700 ES=0700 SS=0700 CS=0700 IP=013D

mov di,offset ArrayB ;X=0759 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0102 DI=0120 DS=0700 ES=0700 SS=0700 CS=0700 IP=0140

mov bx,offset Difference ;AX=0759 BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0102 DI=010C DS=0700 ES=0700 SS=0700 CS=0700 IP=0143

mov cx,10 ;AX=0759 BX=0116 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0102 DI=010C DS=0700 ES=0700 SS=0700 CS=0700 IP=0146

cld ;AX=0759 BX=0116 CX=000A DX=0000 SP=FFFE BP=0000 SI=0102 DI=010C DS=0700 ES=0700 SS=0700 CS=0700 IP=0149

findDE:

cmpsb ;AX=0759 BX=0116 CX=000A DX=0000 SP=FFFE BP=0000 SI=0102 DI=010C DS=0700 ES=0700 SS=0700 CS=0700 IP=014A

;AX=0023 BX=0117 CX=0009 DX=0000 SP=FFFE BP=0000 SI=0103 DI=010D DS=0700 ES=0700 SS=0700 CS=0700 IP=014A

jne NotEqual ;AX=0759 BX=0116 CX=000A DX=0000 SP=FFFE BP=0000 SI=0103 DI=010D DS=0700 ES=0700 SS=0700 CS=0700 IP=014B

;AX=0023 BX=0117 CX=0009 DX=0000 SP=FFFE BP=0000 SI=0104 DI=010E DS=0700 ES=0700 SS=0700 CS=0700 IP=014B

inc NumOfEqual ;AX=0023 BX=0117 CX=0009 DX=0000 SP=FFFE BP=0000 SI=0104 DI=010E DS=0700 ES=0700 SS=0700 CS=0700 IP=014D

inc bx ;AX=0759 BX=0116 CX=000A DX=0000 SP=FFFE BP=0000 SI=0103 DI=010D DS=0700 ES=0700 SS=0700 CS=0700 IP=0170

;AX=0023 BX=0117 CX=0009 DX=0000 SP=FFFE BP=0000 SI=0104 DI=010E DS=0700 ES=0700 SS=0700 CS=0700 IP=0151

dec di ;AX=0759 BX=0117 CX=000A DX=0000 SP=FFFE BP=0000 SI=0103 DI=010D DS=0700 ES=0700 SS=0700 CS=0700 IP=0171

;AX=0023 BX=0118 CX=0009 DX=0000 SP=FFFE BP=0000 SI=0104 DI=010E DS=0700 ES=0700 SS=0700 CS=0700 IP=0152

dec si ;AX=0759 BX=0117 CX=000A DX=0000 SP=FFFE BP=0000 SI=0103 DI=010C DS=0700 ES=0700 SS=0700 CS=0700 IP=0172

;AX=0023 BX=0118 CX=0009 DX=0000 SP=FFFE BP=0000 SI=0104 DI=010D DS=0700 ES=0700 SS=0700 CS=0700 IP=0153

mov al,byte ptr ds:[si] ;AX=0759 BX=0117 CX=000A DX=0000 SP=FFFE BP=0000 SI=0102 DI=010C DS=0700 ES=0700 SS=0700 CS=0700 IP=0173

;AX=0759 BX=0117 CX=000A DX=0000 SP=FFFE BP=0000 SI=0102 DI=010C DS=0700 ES=0700 SS=0700 CS=0700 IP=0174

;AX=0023 BX=0118 CX=0009 DX=0000 SP=FFFE BP=0000 SI=0103 DI=010D DS=0700 ES=0700 SS=0700 CS=0700 IP=0154

;AX=0023 BX=0118 CX=0009 DX=0000 SP=FFFE BP=0000 SI=0103 DI=010D DS=0700 ES=0700 SS=0700 CS=0700 IP=0155

cbw ;AX=0705 BX=0117 CX=000A DX=0000 SP=FFFE BP=0000 SI=0102 DI=010C DS=0700 ES=0700 SS=0700 CS=0700 IP=0176

;AX=000A BX=0118 CX=0009 DX=0000 SP=FFFE BP=0000 SI=0103 DI=010D DS=0700 ES=0700 SS=0700 CS=0700 IP=0157

add SumOfEqual, ax ;AX=0005 BX=0117 CX=000A DX=0000 SP=FFFE BP=0000 SI=0102 DI=010C DS=0700 ES=0700 SS=0700 CS=0700 IP=0177

;AX=000A BX=0118 CX=0009 DX=0000 SP=FFFE BP=0000 SI=0103 DI=010D DS=0700 ES=0700 SS=0700 CS=0700 IP=0158

mov al,byte ptr ds:[di] ;AX=0005 BX=0117 CX=000A DX=0000 SP=FFFE BP=0000 SI=0102 DI=010CDS=0700 ES=0700 SS=0700 CS=0700 IP=017B

;AX=0005 BX=0117 CX=000A DX=0000 SP=FFFE BP=0000 SI=0102 DI=010C DS=0700 ES=0700 SS=0700 CS=0700 IP=017C

;AX=000A BX=0118 CX=0009 DX=0000 SP=FFFE BP=0000 SI=0103 DI=010D DS=0700 ES=0700 SS=0700 CS=0700 IP=015C

;AX=000A BX=0118 CX=0009 DX=0000 SP=FFFE BP=0000 SI=0103 DI=010D DS=0700 ES=0700 SS=0700 CS=0700 IP=015D

cbw ;AX=0023 BX=0117 CX=000A DX=0000 SP=FFFE BP=0000 SI=0102 DI=010CDS=0700 ES=0700 SS=0700 CS=0700 IP=017E

;AX=000A BX=0118 CX=0009 DX=0000 SP=FFFE BP=0000 SI=0103 DI=010D DS=0700 ES=0700 SS=0700 CS=0700 IP=015F

add SumOfEqual, ax ;AX=0023 BX=0117 CX=000A DX=0000 SP=FFFE BP=0000 SI=0102 DI=010C DS=0700 ES=0700 SS=0700 CS=0700 IP=017F

;AX=000A BX=0118 CX=0009 DX=0000 SP=FFFE BP=0000 SI=0103 DI=010D DS=0700 ES=0700 SS=0700 CS=0700 IP=0160

inc si ;AX=0023 BX=0117 CX=000A DX=0000 SP=FFFE BP=0000 SI=0102 DI=010C DS=0700 ES=0700 SS=0700 CS=0700 IP=0183

;AX=000A BX=0118 CX=0009 DX=0000 SP=FFFE BP=0000 SI=0103 DI=010D DS=0700 ES=0700 SS=0700 CS=0700 IP=0164

inc di ;AX=0023 BX=0117 CX=000A DX=0000 SP=FFFE BP=0000 SI=0103 DI=010C DS=0700 ES=0700 SS=0700 CS=0700 IP=0184

;AX=000A BX=0118 CX=0009 DX=0000 SP=FFFE BP=0000 SI=0104 DI=010D DS=0700 ES=0700 SS=0700 CS=0700 IP=0165

jmp NextElement ;AX=000A BX=0118 CX=0009 DX=0000 SP=FFFE BP=0000 SI=0104 DI=010E DS=0700 ES=0700 SS=0700 CS=0700 IP=0166

NotEqual:

inc NumOfDiff

mov byte ptr ds:[bx],'N' ; AX=0759 BX=0116 CX=000A DX=0000 SP=FFFE BP=0000 SI=0103 DI=010D DS=0700 ES=0700 SS=0700 CS=0700 IP=0168

inc bx ;AX=0759 BX=0116 CX=000A DX=0000 SP=FFFE BP=0000 SI=0103 DI=010D DS=0700 ES=0700 SS=0700 CS=0700 IP=016C

dec di ;AX=0759 BX=0116 CX=000A DX=0000 SP=FFFE BP=0000 SI=0103 DI=010D DS=0700 ES=0700 SS=0700 CS=0700 IP=016D

dec si

mov al,byte ptr ds:[si]

cbw

add SumOfDiff, ax

mov al,byte ptr ds:[di]

cbw

add SumOfDiff, ax

inc si

inc di

NextElement:

loop findDE ;AX=0023 BX=0117 CX=000A DX=0000 SP=FFFE BP=0000 SI=0103 DI=010D DS=0700 ES=0700 SS=0700 CS=0700 IP=0185

;AX=000A BX=0118 CX=0009 DX=0000 SP=FFFE BP=0000 SI=0104 DI=010E DS=0700 ES=0700 SS=0700 CS=0700 IP=0185

mov ax,4c00h ;AX=0000 BX=0120 CX=0000 DX=0000 SP=FFFE BP=0000 SI=010C DI=0116 DS=0700 ES=0700 SS=0700 CS=0700 IP=0187

int 21h ;AX=4C00 BX=0120 CX=0000 DX=0000 SP=FFFE BP=0000 SI=010C DI=0116 DS=0700 ES=0700 SS=0700 CS=0700 IP=018A

; AX=4C00 BX=0120 CX=0000 DX=0000 SP=FFF8 BP=0000 SI=010C DI=0116 DS=0700 ES=0700 SS=0700 CS=F400 IP=0200

; AX=4C00 BX=0120 CX=0000 DX=0000 SP=FFF8 BP=0000 SI=010C DI=0116 DS=0700 ES=0700 SS=0700 CS=F400 IP=0204

end start

Запишемо іншу програму та простежимо за її роботою. На рисунку 2 подано результат роботи програми

// string.cpp: визначає точку входу для консольного

//застосування

#include "stdafx.h"  
#include <windows.h> //необхідний для роботи DWORD #include <stdio.h> // необхідний для роботи printf #include <conio.h> // необхідний для роботи \_getch() /\* Оголошення функції \*/  
DWORD lens (char \*); // Функція визначення довжини

рядка  
void cats(char\*, char\*,char\*); // функція злиття двох

//рядків у третій

/\* Оголошення змінних \*/

char a[]="fdhfjliop";

char b[]="12345";

char c[]="4";

int \_tmain(int argc, \_TCHAR\* argv[])

{

\_\_asm{  
/\* необхідно викликати процедуру cats, яка за свої

параметри має три покажчика на рядок типу char (у порядку a, b, c). Щоб процедура cats могла отримати параметри, що передаються за значенням, необхідно помістити їх у стек (у порядку c, b, a), а потім вирівняти стек \*/

LEA EAX, c; // поміщаємо у регістр EAX адресу першого // елемента рядка с

PUSH EAX; // поміщаємо вміст EAX у стек  
LEA EAX, b; // поміщаємо в регістр EAX адресу першого

// елемента рядка b

PUSH EAX; // поміщаємо вміст EAX у стек

LEA EAX, a; // поміщаємо в регістр EAX адресу першого // елемента рядка a

PUSH EAX; // поміщаємо вміст EAX у стек

CALL cats; // викликаємо процедуру

ADD ESP,12; // вирівнюємо стек

};

printf("%s\n", c); // виводимо сумарний рядок

\_getch();

return 0;

}

// функція визначення довжини рядка

DWORD lens (char \* s)

{

DWORD l=0;

\_\_asm

{  
CLD; /\* задаємо напрямок сканування (скидаємо

прапорець напрямку DF). Цей прапорець ураховується в рядкових операціях. Якщо прапорець дорівнює одиниці, то в рядкових операціях адреса автоматично зменшується. Щоб адреса автоматично збільшувалася, цей прапорець необхідно скинути\*/

MOV EDI, s; // поміщаємо адресу початку рядка в

// регістр EDI

MOV ESI,EDI; // зберігаємо адресу рядка в регістрі ESI

MOV ECX, 0ffffffffh; // поміщаємо в регістр- лічильник // циклу максимальне 32-бітове ціле число

XOR AL,AL; // запускаємо нескінченний цикл – операція // XOR від двох однакових елементів повертає

0

// таким чином, ця команда являє собою "поки 0 ..."

REPNE SCASB; // скануємо рядок байтів, поки

// не трапиться 0

SUB EDI,ESI; // віднімаємо з поточної адреси, яка

// зберігається в EDI, адресу початку // рядка (що зберігається в ESI), таким // чином, знаходимо довжину рядка разом // з термінальним символом

DEC EDI; // віднімаємо з отриманої довжини рядка 1

// виключаємо термінальний символ

MOV l, EDI; // поміщаємо в змінну l довжину рядка

}

return l; // повертаємо довжину рядка

}

// Функція злиття рядків s1+s2->s3

void cats (char\* s1, char\* s2,char\* s3)

{

\_\_asm{

/\* визначаємо довжину рядка s1\*/

CLD; // задаємо напрямок копіювання

MOV ESI,s1; // поміщаємо в регістр ESI покажчик

// на початок рядка s1

PUSH ESI; // поміщаємо покажчик на s1 у стек

CALL lens; // викликаємо функцію визначення довжини

// рядка

ADD ESP,4; // вирівнюємо стек

/\* копіюємо вміст рядка s1 у рядок s3\*/  
/\* функція lens повертає результат у регістр EAX \*/ MOV ECX, EAX; // поміщаємо довжину рядка s1 у

// регістр ECX  
MOV ESI,s1; // поміщаємо в регістр ESI покажчик

// на рядок s1  
MOV EDI,s3; // поміщаємо в регістр EDI покажчик

// на рядок s3  
REP MOVSB; /\* повторюємо, поки вміст ECX не

повернувся в 0, побайтове копіювання рядка, адреса початку якого зберігається в ESI, в рядок, адреса початку

якого зберігається в EDI (копіюємо вміст рядка s1 у рядок s3) \*/

/\* копіюємо вміст рядка s2 у рядок s3, починаючи з позиції після s1 \*/

MOV ESI,s2; // поміщаємо в ESI покажчик на перший // елемент рядка s2

PUSH ESI; // поміщаємо покажчик на s2 у стек  
CALL lens; // викликаємо функцію визначення довжини

// рядка

ADD ESP,4; // вирівнюємо стек

/\* функція lens повертає результат у регістр EAX \*/ MOV ECX,EAX; // поміщаємо довжину рядка s21 у

// регістр ECX  
REP MOVSB; /\* повторюємо, поки вміст ECX не

повернувся в 0, побайтове копіювання рядка, адреса початку якого зберігається в ESI (адреса початку рядка s2), у рядок, адреса початку якого зберігається в EDI (адреса кінця рядка s1 + 1) (копіюємо вміст рядка s2 в рядок s3) \*/

MOV BYTE PTR [EDI],0; // фіксуємо кінець рядка

}

}

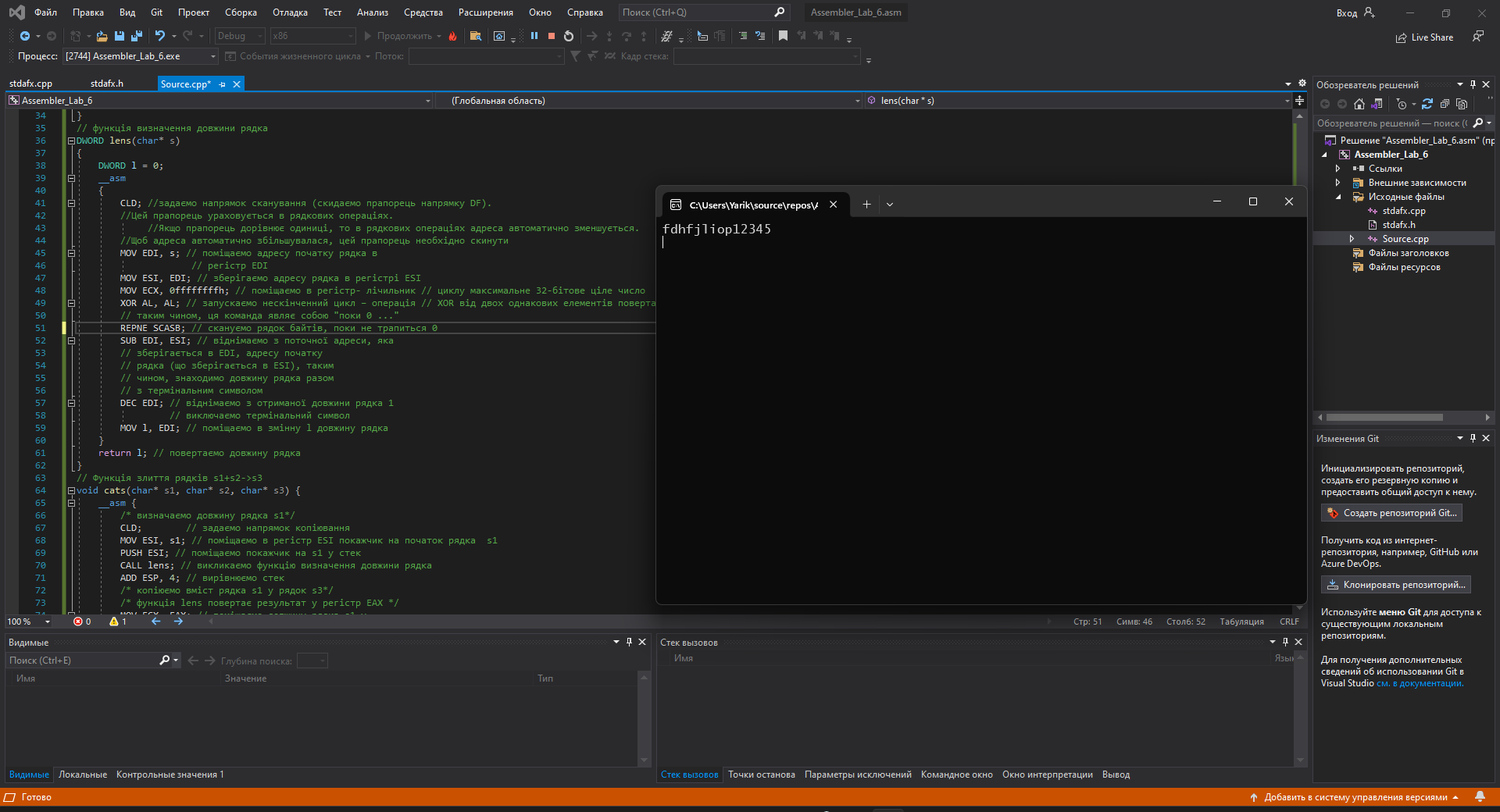


Рисунок 2 – результат роботи другої програми

Висновок

Під час виконання даної лабораторної роботи було проведено дослідження режимів адресації та вивчено правила їх використання в мові асемблера. Було ретельно проаналізовано процес створення виконуваних файлів з допомогою асемблера та визначено основні кроки цього процесу.

Одним із ключових результатів роботи стала розробка таблиці очікуваних значень операндів, що базується на функціонуванні програми. Ця таблиця дозволила зрозуміти взаємозв'язок між командами асемблера та їх вплив на дані, які вони обробляють.

Під час вивчення команд переміщення даних та команд повторення строкових інструкцій було отримано глибше розуміння роботи з даними в мові асемблера. Розглянуті команди дозволяють ефективно переміщати та обробляти дані в пам'яті, що є важливим аспектом при написанні програм на асемблері.

Окрім того, була вивчена організація масивів мовою асемблера. Зрозуміло, яким чином можна організувати масиви для зручного доступу до елементів та ефективної обробки даних в програмах на асемблері.

У цілому, виконання лабораторної роботи дозволило отримати значний досвід у використанні мови асемблера для роботи з даними, розширити знання про режими адресації та їх застосування, а також покращити вміння в організації та роботі з масивами.