МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І. І. МЕЧНИКОВА
ФАКУЛЬТЕТ МАТЕМАТИКИ, ФІЗИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

АРХІТЕКТУРА КОМП'ЮТЕРІВ ТА НИЗЬКОРІВНЕВЕ ПРОГРАМУВАННЯ

Частина ІІ

ЕЛЕКТРОННІ МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт
для студентів першого (бакалаврського) рівня вищої освіти
спеціальності F7 Комп'ютерна інженерія

Одеса
ОНУ імені І. І. Мечникова
2026

УДК 004.42(072)
А878

Укладачі:

І. В. Шаріпова, старший викладач кафедри комп'ютерних систем та технологій Одеського національного університету імені І. І. Мечникова;

Ю. М. Берков, старший викладач кафедри комп'ютерних систем та технологій Одеського національного університету імені І. І. Мечникова.

Рецензенти:

Ю. А. Гунченко, доктор технічних наук, професор, завідувач кафедри комп'ютерних систем та технологій Одеського національного університету імені І. І. Мечникова;

С. Г. Антощук, доктор технічних наук, професор, професор кафедри інформаційних систем Національного університету «Одеська політехіка».

Рекомендовано вченою радою факультету математики, фізики та інформаційних технологій ОНУ імені І. І. Мечникова. Протокол № 3 від 3 листопада 2025 р.

А878 **Архітектура** комп'ютерів та низькорівневе програмування [Електронний ресурс] : електрон. метод. вказівки до виконання лаб. робіт для студентів першого (бакалавр.) рівня вищ освіти спец. F7 Комп'ютерна інженерія / уклад.: І. В. Шаріпова, Ю. М. Берков. Одеса : ОНУ імені І. І. Мечникова, 2026. Ч. 2. 29 с. 1,1 МБ.

Розглянуто питання використання середовища програмування MS MASM 6 на мові асемблер, етапи створення програми на MASM, особливості використання відладчика та лабораторні роботи з завданнями. Наведені перелік робіт з темами завдань, довідкові матеріали, необхідні для виконання окремих частин, та список рекомендованої літератури.

Призначено для здобувачів спеціальності F7/123 Комп'ютерна інженерія.

УДК 004.42(072)

© Одеський національний університет імені І. І. Мечникова, 2026

ЗМІСТ

ВСТУП

Основу сучасних інформаційних технологій (ІТ) та їх базис становлять апаратні засоби комп'ютерної техніки. У зв'язку з цим важливе значення в підготовці фахівців в області ІТ та комп'ютерної інженерії надається вивченню архітектури комп'ютерів, зокрема, основних відомостей щодо їх апаратних та програмних засобів, організації роботи в цілому, способів подання даних та системного і низькорівневого програмування. Таким чином, курс «Архітектура комп'ютерів та низькорівневе програмування» надає можливість вивчення методів та засобів проектування сучасних комп'ютерів, визначення ефективного використання у практиці, способи подання програм і даних, а також сучасні архітектурні рішення.

Метою курсу «Архітектура комп'ютерів та низькорівневе програмування» є вивчення та опанування студентами основних відомостей про апаратні та програмні засоби сучасних комп'ютерів, способи подання програм і даних, про призначення, структуру й особливості функціонування окремих пристроїв комп'ютера, про організацію його роботи в цілому, а також сучасні архітектурні рішення, що сприяли значному підвищенню продуктивності комп'ютерів. Основна увага приділяється вивченню принципів роботи центральних пристроїв комп'ютера: процесора і оперативної пам'яті, архітектурі сучасних комп'ютерів, системі машинних команд та основам низькорівневого програмування

У методичних вказівках розглянуті ключові підсистеми операційних систем, а саме: підсистема управління оперативною пам'яттю, підсистема організації задач, керування процесами і потоками, управління файловою системою, управління засобами введення-виведення, мережевими засобами, засобами безпеки тощо.

1. Загальні положення

Метою лабораторних занять з дисципліни «Архітектура комп'ютерів та низькорівневе програмування» є набуття навичок роботи з сучасною обчислювальною технікою , її апаратним і програмним забезпеченням для розв'язку типових задач.

Методичні вказівки складено у відповідності до робочої програми з дисципліни «Архітектура комп'ютерів та низькорівневе програмування».

Лабораторні роботи виконується на лабораторних заняттях з дисципліни. Підготовка до лабораторних занять здійснюється здобувачами в часи самостійної роботи.

Для здобувачів денної форми навчання виконання всіх лабораторних робіт даного навчального видання є обов'язковим. Здобувачі заочної форми навчання

виконують лише роботи, передбачені робочою програмою навчальної дисципліни.

Перелік і кількість запропонованих до розв’язку задач визначається викладачем, який веде лабораторні заняття, відповідно до робочої програми дисципліни.

Усі лабораторні роботи складено за єдиним планом – на початку кожної роботи наведено її тему та мету, далі викладено короткі теоретичні відомості щодо тематики роботи з розглядом прикладів розв’язку типових задач, після чого містяться індивідуальні завдання та запитання.

Наприкінці методичних вказівок наведено порядок виконання і захисту робіт та перелік посилань на використане укладачами видання та рекомендовану для виконання роботи літературу.

Після виконання кожної роботи здобувачі складають звіт, котрий захищають. За результатами виконання та захисту роботи виставляються бали за спеціальною шкалою оцінювання, наведеною у робочій програмі. Бали, отримані за окремі роботи, формують загальну суму балів за практикум, яка враховується у підсумкову оцінку за модуль та семестр.

2. Основні вимоги з техніки безпеки

При виконанні лабораторних робіт у комп’ютерних залах здобувач повинен дотримуватися означених нижче правил техніки безпеки:

1. Не приступати до роботи, не прослухавши інструктаж з техніки безпеки та не зареєструвавшись у журналі користувачів ПК (перед початком виконання першої роботи).
2. Не вмикати чи вимикати живлення у комп'ютерному залі або на своєму робочому місці та не перезавантажувати ПК.
3. Не змінювати налагоджень комп'ютера та його операційної системи.
4. Не встановлювати та не видаляти жодних програм.
5. Не працювати за комп’ютером більше як удвох одночасно.
6. Закінчивши виконання лабораторної роботи, скопіювати результати на власний носій інформації та закрити всі відкриті програми без збереження на ПК власних результатів.
7. Повідомити відповідального інженера та викладача про закінчення роботи і звільнення свого робочого місця.

Лабораторна робота № 9. Організація циклів

Тема: програмування задач із циклами.

Мета роботи: надати навички програмування задач із циклами.

Устаткування: персональний комп'ютер.

Програмне забезпечення: Windows 7/8/10, DOSBoxPortable, MS MASM 6.x.

Хід роботи

1. Написати програму з даними відповідно до вашого варіанта
2. Одержати файл програми, що виконується, з файлом лістингу
3. Зберегти отримані файли в папку lab9.asm
4. Запустити програму й перевірити правильність рішення завдання
5. Записати всі дані, необхідні для оформлення звіту
6. Закрити всі програми

Арифметичні та логічні інструкції – це тип інструкцій на мові асемблера, які виконують математичні та логічні операції над даними, що зберігаються в регістрах і комірках пам'яті.

Арифметичні команди виконують прості арифметичні операції, такі як додавання, віднімання, множення та ділення. Ці команди працюють з числами та виконують обчислення. Приклади арифметичних команд:

ADD: Додає значення одного операнда до іншого

SUB: Віднімає значення одного операнда від іншого

MUL: Множить значення одного операнда на інший

DIV: Ділить значення одного операнда на інший

Логічні команди виконують операції над бітами, такі як логічне І (AND), логічне АБО (OR), виключне АБО (XOR) та заперечення (NOT). Логічні команди використовуються для маніпуляції бітами в регістрах та пам’яті.

Приклади логічних команд:

AND: Виконує логічне І над двома операндами

OR: Виконує логічне АБО над двома операндами

XOR: Виконує виключне АБО над двома операндами

NOT: Виконує заперечення одного операнда

Арифметичні команди працюють з числами та виконують обчислення, тоді як логічні команди працюють з бітами та виконують операції над ними. А також арифметичні команди можуть змінювати значення прапорів, такі як прапор

переповнення та прапор знаку, під час виконання операцій, тоді як логічні команди не впливають на ці прапори [1].

Завдання

Використовуючи керуючі структури While і Repeat/Until, створіть програму типу .EXE, що обчислює суму ряду 5(n+2)×n, де n змінюється від 1 до 10.

Використовуючи текстовий редактор, створіть файл lab9.asm у папці D:\PROJECT.

Введіть програмний код програми вашого варіанта. Для додавання у коментарі значень регістрів після виконання математичних операцій, скористайтеся відладчиком.

```
; ---------- Програмний код ----------
COMMENT !
Використовуючи керуючі структури While і Repeat/Until створіть програму типу .EXE, що обчислює суму ряду 5(n+2) n від 1 до 10
!

.model small
.stack 100h
.data
five    word  5

.code .386 main:


mov     ax, @data       ; Сегментна адреса
mov     ds,ax           ; початок визначення даних в DS

mov     bx, 0           ; Початкове значення регістру, в якому буде зберігається поточна сума
mov     cx, 0           ; Початкове становище лічильника

L1:
inc     cx              ; Збільшимо лічильник на 1
mov     ax, cx          ; Запишемо значення лічильника в AX (n=1)
add     ax, 2           ; Додамо до AX 2. (n+2)
mul     five            ; Помножимо суму на п'ять. 5(n+2)
add     bx, ax          ; Додамо поточний доданок до поточної суми
cmp     cx,10           ; Якщо поточне значення лічильника менше 10
jl      L1              ; перейти на мітку L1

mov     ax, 4C00h       ; Номер функції DOS:4C00h завершити програму в AX
int     21h             ; Виклик функції DOS з AX
end     main
```

1. Зробіть трансляцію й компонування програми.
   1.1. Використовуючи команди MSDOS, перейдіть у папку D:\PROJECT (cd D:\PROJECT)
   1.2. Ввести команди з урахуванням регістру символів:
   ml /c /Zi /Fl lab9.asm link /co lab9.obj

Питання до захисту роботи: опанування та використання, перелік арифметичних та логічних команд, сутність команд переходу.

Вказівки до змісту звіту

1. Заповнити бланк звіту.
2. Ім'я створюваного файла lab9.asm.
3. При написанні програми й оформленні звіту кожний рядок повинен супроводжуватися коментарями.
4. При написанні звіту скористайтеся відладчиком CodeView.

Варіанти завдань для лабораторної роботи № 9

1. $\sum_{n=1}^{10} 5(n+2)$	**2.** $\sum_{n=1}^{10} 2n^2$	**3.** $\sum_{n=1}^{10} 4n^2$
4. $\sum_{n=1}^{10} (12-n)^2$	**5.** $\sum_{n=1}^{10} (2+n)^2$	**6.** $\sum_{n=1}^{10} 8(12-n)$
7. $\sum_{n=1}^{10} 4(2n+1)$	**8.** $\sum_{n=1}^{10} (8n-1)$	**9.** $\sum_{n=1}^{10} (6n^2-5)$
10. $\sum_{n=1}^{10} (11-n)^2$	**11.** $\sum_{n=1}^{10} 3n^2$	**12.** $\sum_{n=1}^{10} 7(n+1)$
13. $\sum_{n=1}^{10} (110-n^2)$	**14.** $\sum_{n=1}^{10} (n^2-1)$	**15.** $\sum_{n=1}^{10} 5(21-n)$

Лабораторна робота № 10. Робота з рядками

Тема: використання команд обробки рядків і таблиць.

Мета роботи: надати навички використання команд обробки рядків і таблиць.

Устаткування: персональний комп'ютер.

Програмне забезпечення: Windows 7/8/10, DOSBoxPortable, MS MASM 6.x.

Хід роботи

1. Створити програму відповідно до вашого варіанта.
2. При реалізації програми кожний рядок повинен супроводжуватися коментарями.
3. Зберегти програму в папку lab10.asm.
4. Записати всі дані, необхідні для оформлення звіту.
5. Закрити всі програми.

Команди обробки рядків у мові асемблера – це набір інструкцій, які дозволяють виконувати операції з текстовими даними, зокрема копіювати, порівнювати, шукати символи в рядках, а також виводити їх на екран. Ці інструкції особливо корисні, коли мова йде про маніпулювання великими обсягами даних або при необхідності виконати складніші операції, такі як пошук чи редагування частини рядка. У мові асемблера для процесорів архітектури x86 є кілька команд, які працюють з рядками або блоками пам'яті.

Можна зберігати рядки із завершальним символом-вартом для їх розмежування, замість того, щоб явно зберігати довжину рядка. Символ-варт має бути спеціальним символом, який не зустрічається у рядку.

Кожна рядкова інструкція може вимагати операнд джерела, операнд призначення або обидва. Для 32-бітних сегментів рядкові інструкції використовують регістри ESI та EDI, щоб вказати на операнди джерела та призначення відповідно. Однак для 16-бітних сегментів регістри SI та DI використовуються для вказівки на джерело та призначення відповідно.

Рядки символів являють собою звичайні одновимірні масиви, однак через деякі особливості подання їх виділяють в окрему категорію. Рядки символів розглядають як ланцюги байт або слів довільної довжини. Ознакою кінця ланцюга байт є нульовий байт (не символ «0» з ASCII-кодом 30h, а двійковий 0), а ознакою кінця ланцюга слів – нульове слово (два нульових байти). Однобайтові рядки містять коди символів у певному кодуванні, а рядки слів використовують

двобайтове кодування Unicode з різним порядком зберігання байтів у слові. Рядки символів можуть бути оброблені як звичайні одновимірні масиви байтів або слів, однак для розв'язання більшості типових задач зручніше використання команд обробки ланцюжків у поєднанні з префіксами повторення [3].

Завдання

Надано рядок символів: «Assembler language programming is the fastest in the world.». Написати програму, яка формує новий рядок, виділивши з початкового слово "Assembler".

1. Використовуючи текстовий редактор, створіть файл **lab10.asm** у папці **D:\PROJECT.**
2. Введіть програмний код програми вашого варіанта.

```
; ---------- Програмний код ----------
COMMENT !
Лабораторна робота № 10
Дано рядок символів: «Assembler language programming is the fastest in the world.»  Написати програму, що формує новий рядок, виділивши з початкового слово "Assembler".
!
.model small
.stack 100h
.data
Original_string db 'Assembler language programming is the fastest in the world.', 13, 10, '$'; Початковий рядок - рядок джерело.
Formed_line db 255 DUP(?)              ;Сформований рядок.
.code  .386  main:



mov ax, @data                          ; Сегментна адреса
mov ds,ax                              ; початок визначення даних в DS
mov es,ax                              ; Ініціалізуємо сегментний регістр ES

; ========= Вивід початкового рядка. ============
mov dx, offset Original_string  mov ah, 9
int 21h
; ================================
cld                                    ; Скинемо прапор DF і встановимо прямий напрямок
mov ecx, LENGTHOF Original_string  ; Задамо значення лічильника
sub ecx, 10                            ; Віднімемо від довжини рядка довжину слова 'Assembler'
mov esi, [OFFSET Original_string]+10  ; Задамо адресу джерела + довжина рядка 'Assembler'
mov edi, OFFSET Formed_line  ; Задамо адресу одержувача даних  rep movsb
          ; Сформуємо потрібний рядок.

; ====== Вивід сформованого рядка. ============
mov dx, offset Formed_line  mov ah, 9
int 21h  ;   mov ax, 4C00h                ; Номер функції DOS:4C00h завершити програму в AX
int 21h                                ; Виклик функції DOS із AX  end main
```

3. Зробіть трансляцію й компонування програми.
   3.1. Використовуючи команди MSDOS, перейдіть у папку D:\PROJECT (cd D:\PROJECT)
   3.2. Ввести команди з урахуванням регістру символів:
   ml /c /Zi /Fl lab10.asm
   link /co lab10.obj;

Питання до захисту роботи: яким чином працюють команди обробки рядків і таблиць.

Вказівки до змісту звіту

1. Текст програми. Кожний оператор супроводити коментарями.
2. Результат роботи програми.

Варіанти завдань для лабораторної роботи № 10

№ вар.	**Дано рядок символів: «Assembler language programming is the fastest in the world. »**
1.	Написати програму, що формує новий рядок, додаючи по два пробіли між словами замість одного.
2.	Написати програму, що формує новий рядок, заміняючи пробіл символом «-».
3.	Написати програму, що формує новий рядок, що складається із символів початкового рядка записаних навпаки.
4.	Написати програму, що формує новий рядок, що формує новий рядок з початкового замінивши слово «Assembler» словом «What» а символ «.» символом «?».
5.	Написати програму, що формує новий рядок, що складається із символів початкового у верхньому регістрі.
6.	Написати програму, що формує новий рядок з початкового, роблячи перші букви слів заголовними.
7.	Написати програму, що формує новий рядок з початкового, роблячи останні букви слів заголовними.
8.	Написати програму, що формує новий рядок, видаливши з початкового слова «in the world».
9.	Написати програму, що формує новий рядок, додавши символи переведення рядка й каретки після кожного слова.

10.	Написати програму, що формує новий рядок, видаливши з початкового слова «language».
11.	Написати програму, що формує новий рядок, заміняючи всі символи «i» на «I».
12.	Написати програму, що формує новий рядок, у якому записані тільки перші 30 символів.
13.	Написати програму, що формує новий рядок, заміняючи всі символи «m» на «M».
14.	Написати програму, що формує новий рядок, у якому записані тільки 20 останніх символів.
15.	Написати програму, що формує новий рядок, видаливши з початкового слово «Assembler» і замінивши його пробілами.

Лабораторна робота № 11. Робота зі стеком

Тема: підпрограми й стек.

Мета роботи: надати навички роботи із процедурами.

Устаткування: персональний комп'ютер.

Програмне забезпечення: Windows 7/8/10, DOSBoxPortable, MS MASM 6.x.

Хід роботи

1. Створити програму відповідно до вашого варіанта.
2. При реалізації програми кожний рядок повинен супроводжуватися коментарями.
3. При виконанні програми значення змінних вибрати самостійно. Перевірку програми виконати для різних комбінацій змінних.
4. Зберегти програму в папку lab11.asm.
5. Записати всі дані, необхідні для оформлення звіту (для аналізу програми скористайтеся відладчиком CodeView).
6. Закрити всі програми.

Стек – це структура даних типу LIFO (Last In, First Out), яка використовується для зберігання тимчасових даних, таких як адреси повернення, параметри функцій і локальні змінні. У процесі виклику підпрограми стек використовується для:

- збереження адреси повернення, щоб після завершення підпрограми програма могла продовжити виконання з правильного місця;
- передачі параметрів до підпрограми;
- збереження значень регістрів, які можуть бути змінені в підпрограмі, щоб після повернення відновити початкові значення [2].

Завдання

Використовуючи процедури, створіть програму, яка обчислює вираз $r2 = a - b * 10 - c$ з 32 розрядними цілими змінними. Для передачі параметрів використовуйте регістри, результат зберегти в пам'ять.

1. Використовуючи текстовий редактор, створіть файл **lab11.asm** у папці **D:\PROJECT.**
2. Введіть програмний код програми, що використовує процедури.

```
; Програмний код
 .model small
.stack 100h
.data

;Оформіть змінні, необхідні для реалізації завдань
a DWORD 100  b DWORD 4  d DWORD 10  r2 DWORD ?                ; Змінна для збереження результату.
.code
.386
main PROC                        ; Це головна процедура програми, керуюча всіма діями.
; Викликає: EvaluationExpression.
mov ax, @data                    ; Сегментна адреса mov ds,ax                ; початок визначення даних в DS  mov ebx, a                ; Підготовляємо параметри (регістри):
mov ecx, b                       ; EBX, ECX, EDX для  mov edx, d             ; передачі в процедуру.  call Evaluation Expression  ; Викликаємо процедуру для обчислення виразу.  mov r2, eax                Зберігаємо відповідь у змінній: r2.
mov ax, 4C00h                    ; Номер функції DOS:4C00h завершити програму в AX
int 21h                          ; Виклик функції DOS з AX  main ENDP
; Обчислює вираз r2 = a - b * 10 - c
; Параметри для обчислення передаються в регістрах: EBX, ECX, EDX, які після виконання процедури відновлюються.
; Результат після виконання буде знаходитися в регістрі EAX.

EvaluationExpression PROC USES ebx ecx edx
; r2 = a - b * 10 - c
CLC                              ; Скидаємо прапор переносу.
mov eax, ecx   mov ecx,          ; Обчислюємо добуток
10  ;  b  *  10.    mul  ecx
mov ecx, eax  mov eax,
ebx


sub eax, ecx                     ; Обчислюємо різницю a - (b * 10).
sub eax, edx                     ; Обчислюємо різницю (a - b * 10) - c.
ret                              ; Вихід із процедури.
EvaluationExpression
 ENDP
end main
```

3. Зробіть трансляцію й компонування програми.
   ml /c /Zi /Fl lab11.asm link /co lab11.obj; Завдання

Питання до захисту роботи:

1. Які команди враховують знак при виконанні обчислень?
2. Наведіть класифікацію команд передачі управління.
3. Охарактеризуйте команди умовного переходу.

Вказівки до змісту звіту

1. Текст програми. Кожний оператор супроводити коментарями.
2. Результат роботи програми.

Варіанти завдань для лабораторної роботи № 11

№	**Вираз**
1.	rl=5*(a+b)-c
2.	r2=a-b*10-c
3.	r3=b+c*a^2
4.	r4=a^2+b^2+c^2
5.	r5 =a^2+c-b
6.	r6=10*(a*b-c)
7.	r7=10*a+20*b+c
8.	r8=c^4-a^3-c^2
9.	r9=a^2+b^2-c
10.	r10=(a-b)*(b-c)*(c-
11.	r11=$(a+b+c)^2$
12.	r12=(a-b)*$(b-c)^2$
13.	r13=a-b+c^2
14.	r14=14(200-a+b)-
15.	r15=10*(a+b+c)-1

Лабораторна робота № 12. Консольний ввід/вивід даних

Тема: створення програм з можливістю вводу з клавіатури й виводу на екран засобами MS DOS і BIOS.

Мета роботи: навчити виконувати програмний ввід-вивід засобами переривань MS DOS і BIOS.

Устаткування: персональний комп'ютер.

Програмне забезпечення: Windows 7/8/10, DOSBoxPortable, MS MASM 6.x.

Хід роботи

1. Створити програму.
2. При реалізації програми кожний рядок повинен супроводжуватися коментарями.
3. При виконанні програми значення змінних вибрати самостійно. Перевірку програми виконати для різних комбінацій змінних.
4. Зберегти програму в папку lab12.asm.
5. Записати всі дані, необхідні для оформлення звіту.
6. Закрити всі програми.

У середовищі MS-DOS консольний ввід/вивід (I/O) даних є ключовою складовою для взаємодії користувача з програмою. У програмах на мові асемблера такі операції зазвичай реалізуються за допомогою переривань, зокрема int 21h, який викликає функції DOS для доступу до пристроїв введення та виведення, таких як клавіатура та екран. Кожна функція DOS, що викликається через int 21h, визначається значенням, яке поміщається в регістр AH.

Для виведення тексту на екран широко використовується функція 09h, яка виводить строку, що закінчується символом $. У регістр AH записується значення 09h, а в DX – адреса рядка. Цей спосіб є зручним для повідомлень, інструкцій користувачу та відображення результатів обчислень.

Щодо введення – часто застосовується функція 0Ah – "Buffered Input". Вона дозволяє зчитувати рядок символів з клавіатури до буфера, структура якого передбачає перший байт – максимальну довжину, другий байт – фактичну довжину введеного рядка, а далі самі введені символи. Такий підхід зручний для подальшої обробки введених чисел або тексту. Після зчитування символів їх можна перетворити з ASCII у числове представлення, щоб виконати арифметичні або логічні операції.

Для виводу одного символу або очікування натискання клавіші можна використовувати функції 02h і 08h відповідно. Функція 02h дозволяє виводити один символ (який подається у DL), тоді як функція 08h чекає натискання клавіші та зчитує її код.

Усі ці функції є прикладом низькорівневої роботи з апаратними засобами комп'ютера через BIOS і DOS. Такий рівень контролю дозволяє програмісту точно визначати, як саме повинна працювати взаємодія з користувачем. Це особливо важливо в системному програмуванні або при навчанні принципам роботи комп'ютера на найнижчому рівні.

Загалом, консольний ввід/вивід у DOS на асемблері базується на чітких викликах через int 21h, з урахуванням правильного завдання значень у регістрах і дотримання формату буферів. Такий підхід вимагає глибокого розуміння архітектури і принципів роботи DOS-системи [3].

Завдання

Створити програму, яка обчислює суму трьох позитивних чисел введених з клавіатури і виводить результат (позитивне число) на екран в десятковому вигляді.

1. Використовуючи текстовий редактор, створіть файл lab12.asm у папці D:\PROJECT.
2. Введіть програмний код програми, що обчислює значення виразу. із трьох позитивних чисел введених із клавіатури й виводить результат (позитивне число) на екран у десятковому вигляді.

```
; ---------- Програмний код ----------  .model small

.stack 100h

.data val_A dw ? val_B dw ? val_C dw ?
ConditionProblem db 'The program finds the sum of three numbers.', 13, 10, '$'
FirstEnMsg db 'Enter the first number: $'
SecondEnMsg db 'Enter the second numbers $'
ThirdEnMsg db 'Enter the third number: $'
MsgSumme db 'The summe of input numbers is:', '$'
MessageUpExit db 10, 13, 10, 'Press any key to exit ...', 13, 10, '$' err_msg db 'Input error', 13, 10, '$' crlf db
0Dh, 0Ah, '$' buffer LABEL BYTE max db 6 act db ? field db 6 dup(?)
L dw 6
OutBuf db 6 dup(?), '$'

.code

.386 main:
mov AX,DGROUP mov DS,AX
mov DX,offset ConditionProblem call writeStr mov DX,offset FirstEnMsg call writeStr call readString call
AsciiToBin mov val_A, AX mov DX,offset crlf
```

```
call writeStr
mov DX, offset SecondEnMsg
call writeStr call readString call AsciiToBin mov val_B, AX mov DX,offset crlf call writeStr
mov DX, offset ThirdEnMsg      ; Вивід запрошення до введення
call writeStr                  ; Вивід рядка.
call readString                ; Читаємо рядок.
call AsciiToBin                ; Переведення числа у двійковий формат.
mov val_C, AX                  ; Запис числа в пам'ять.
mov DX,offset crlf             ; Переведення курсора на наступний рядок
call writeStr                  ; Вивід рядка.
mov ax, val_A
add ax, val_B
add ax, val_C
mov DX,offset OutBuf           ; Адреса буфера для виводу в DX.
mov bx, dx                     ; Задаємо параметр для проц.
call BinToAscii                ; Переводимо двійкове число в рядок.
mov DX,offset MsgSumme         ; Адреса рядка підказки користувачеві в DX.
call writeStr                  ; Вивід рядка.
mov DX,offset OutBuf           ; Адреса рядка з буфером виводу в DX.
call writeStr                  ; Вивід рядка.
mov DX,offset MessageUpExit
call writeStr                  ; Вивід рядка.
call waitPressing              ; Затримує вихід із програми.
call ExitProgram               ; Вихід із програми.
BinToAscii PROC
mov cx, L                      ; Лічильник циклу = довжені рядка виводу.
Fill_Buf:                      ; Очищення буфера виводу:
mov Byte ptr [bx], ' '         ; заповнимо пробілами буфер для виводу.
Inc bx
loop Fill_Buf
mov si, 10                     ; Дільник! SI = 10.
Cld_Dvd:                       ; Початок циклу послідовного ділення позитивного числа.
xor dx, dx
div si
add dx, '0'
dec bx                         ; Заносимо символьне представлення
mov Byte PTR [bx], dl          ; у буфер виводу.
inc cx                         ; Збільшуємо CX на одиницю.
  or ax,ax                     ; Перевіряємо частку на рівність її нулю,
  jnz Cld_Dvd                  ; якщо не дорівнює нулю, продовжимо цикл.
  BinToAscii ENDP
  AsciiToBin PROC
  xor di,di                    ; DI = 0 - номер байта в буфері
  xor ax,ax                    ; AX = 0 - поточне значення результату
  mov cl,act xor ch,ch xor bx,bx
  mov si,cx                    ; SI - довжина буфера.
  mov cl,10                    ; CL = 10, множник для MUL.
  asc2hex:
  mov bl,byte ptr field[di]
  sub bl,'0'                   ; Цифра = код цифри - код символу '0'.
  jb asc_error                 ; Якщо код символу був менше, ніж код "0",
  cmp bl, '9'                  ; або більше, ніж "9",
  ja asc_error                 ; вийти із програми з повідомленням про помилку,
  mul cx                       ; інакше: помножити поточний результат на 10,
  add ax,bx                    ; додати до нього нову цифру,
  inc di                       ; збільшити лічильник
```

```
  cmp di,si                    ; якщо лічильник+1 менше числа символів -
  jb asc2hex                   ; продовжити (лічильник вважається від 0).
ret asc_error: mov dx,offset err_msg call writeStr
mov DX,offset MessageUpExit
call writeStr call waitPressing call ExitProgram AsciiToBin End readString PROC push ax mov dx, offset
Buffer mov ah, 0Ch mov al, 0Ah int 21h xor ch, ch mov cl, act add dx, 2 pop ax ret
readString ENDP writeStr PROC push ax mov ah, 09h int 21h pop ax ret
writeStr ENDP waitPressing PROC push ax mov ah,8 int 21h pop ax
ret
waitPressing ENDP ExitProgram PROC
mov AX,4C00h
int 21h
ret
ExitProgram ENDP
end main
```

3. Зробіть трансляцію й компонування програми.
   ml /c /Zi /Fl lab12.asm link /co lab12.obj;

Питання до захисту роботи

1. 1.З якими числами може працювати процесор?
2. Наведіть класифікацію цілочислових команд процесора?
3. Наведіть класифікацію команд передачі управління.

Вказівки до змісту звіту

1. Текст програми. Кожний оператор супроводити коментарями.
2. Результат роботи програми.

Варіанти завдань для лабораторної роботи № 12

№ вар.	**Вираз**
1.	r1 = 24 * a * b^2 + c^4.
2.	r2 = 5 * (a + b) – c.
3.	r3 = a – b * 10 – c.
4.	r4 = b + c * a^2.
5.	r5 = a^2 + b^2 + c^2
6.	r6 = a^2 + c – b.
7.	r7 = c^4 – a^3 – b.
8.	r8 = a^2 + b^2 – c.
9.	r9 = (a – b) * (b – c) * (c – a)
10.	r10 = $(a+ b+ c)^2$
11.	r11 = (a- b) * $(b-c)^2$
12.	r12 = a- b + c^2
13.	r13 = 14* (200-a+b)+ c^2
14.	r14 = 10 *(a+ b+ c)-1
15.	r15 = 12* $(a+ b)^2$ * c - 2

Лабораторна робота № 13. Робота з текстовими файлами

Тема: вивчення будови й принципів роботи дискових накопичувачів. Функції підтримки дисків і файлів переривання INT 21h.

Мета роботи: ознайомити студентів з основними принципами роботи з дисковими накопичувачами.

Устаткування: персональний комп'ютер.

Програмне забезпечення: Windows 7/8/10, DOSBoxPortable, MS MASM 6.x.

Хід роботи

1. Створити текстовий файл стандартними засобами ОС D:\Project\zfile.txt, у який вписати своє прізвище, ім'я та по батькові на латиниці.
2. Створити програму, що зі створеного раніше текстового файлу (zfile.txt), зчитує ім'я в створений буфер. Після програмно створит новий файл (zname.txt), у який записати вміст буфера.
3. При реалізації програми кожний рядок повинен супроводжуватися коментарями.
4. Зберегти програму й файли в папку для lab13.asm
5. Записати всі дані, необхідні для оформлення звіту.
6. Закрити всі програми.

Завдання

Створити програму з текстового файлу, який містить прізвище, ім'я та по батькові здобувача, записані латиницею. Після цього створити новий файл, в який записати зміст буфера.

1. Використовуючи текстовий редактор, створіть файл lab13.asm у папці D:\PROJECT.
2. Введіть програмний код програми, що буде з текстового файлу (zfile.txt), що містить ваше прізвище, ім'я та по батькові; записані на латиниці, зчитувати ім'я в створений буфер. Після створити новий файл (zname.txt), у який записати вміст буфера.

```
; ---------- Програмний код ----------
TITLE ==== Програма для роботи з файлами. ====

.model small
.stack 100h .data
```

```
Create_File_id      dw      ?                                   ; Ідентифікатор створюваного файлу.
Create_File_name    db      'D:\Project\zname.txt', 0           ; Шлях і ім'я створюваного файлу.
File_id             dw      ?                                   ; Ідентифікатор зчитувального файлу.
File_name           db      'D:\Project \zfile.txt', 0          ; Шлях і ім'я зчитувального файлу.
ErrorFileRead       db      'Error file read.'
MessageUpExit       db      10, 13, 10, 'Press any key to exit ...', 13, 10, '$'
buffer              db      255 dup(?), 13, 10, '$'
. code
 .386 main:
        mov     AX,DGROUP
        mov     DS,AX

; 1) Відкриємо для читання й запису заздалегідь створений файл " D:\Project\zfile.txt"
; у якому записаний рядок, що містить ПІБ (наприклад "Pupkin Vasiliy").

; Відкриття файлу для читання й запису:
; вивід AX = ідентифікатор файлу (якщо не відбулася помилка)
mov     ax, 3D02h                       ; Номер функції для читання й запису файлу в AX. mov         dx,
offset File_name                    ; Ім'я файлу і його розташування в DX. int      21h
        jc      Exit_If_Error_File_Read         ; Якщо відбулася помилка при відкритті файлу –
                                                ; вивести повідомлення.
        mov     File_id, ax     ; Збережемо ідентифікатор файлу в File_id

; 2) Встановимо вказівник для читання ім'я.
; Установка вказівника в початок файлу (DOS 42h: Перемістити вказівник читання/запису).
        mov     ah, 42h                 ; AH = номер функції.
        mov     bx, File_id             ; BX = ідентифікатор файлу.
        mov     cx, 0                   ; CX:DX = відстань, на яку треба перемістити
        mov     dx, 8                   ; вказівник (зі знаком), 8 - довжина прізвища.
        mov     al, 0                   ; AL = переміщення:
                                        ;   0 - від початку файлу; ;   1 - від поточної
                                        позиції; ;   2 - від кінця файлу.


        int     21h

; 3) Зчитаємо рядок символів з файлу й запишемо його в буфер. ; Функція DOS 3Fh:  Читання з файлу
або пристрою.
        mov     ah, 3Fh                 ; AH = номер функції.
        mov     bx, File_id             ; BX = ідентифікатор файлу.
        mov     cx,  6                  ; CX = число байт. (6 - довжина ім'я)
        mov     dx, offset buffer       ; DS:DX = адреса буфера для прийняття даних.
        int     21h

; 4) Закриємо відкритий файл.
; Закриття файлу.
        mov     bx, File_id             ; Ідентифікатор файлу у BX.
mov  ah, 3Eh   int  21h                 ; Номер функції для закриття файлу в AH.


; 5) Створимо новий файл ("D:\Project\zname.txt")для запису ім'я з буфера.
        mov     ah, 5Bh
        mov     cx, 0
        mov     dx, offset Create_File_name
        mov     Create_File_id, ax
```

```
        int         21h

; 6) Закриємо створений файл.
; Закриття файлу.
        mov         bx, Create_File_id      ; Ідентифікатор файлу у BX.
        mov         ah, 3Eh                 ; Номер функції для закриття файлу в AH.
        int         21h

; 7) Відкриємо тільки що створений файл і запишемо в нього вміст буфера.
; Відкриття файлу для читання й запису:
; вихід AX = ідентифікатор файлу (якщо не відбулася помилка)
        mov         ax, 3D02h                   ; Номер функції для читання й запису файлу в AX.
        mov         dx, offset Create_File_name ; Ім'я файлу і його розташування в DX.
         int        21h
          jc        Exit_If_Error_File_Read     ; Якщо відбулася помилка при
                                                ; відкритті файлу - вивести повідомлення.
        mov         Create_File_id, ax          ; Збережемо ідентифікатор файлу в Create_File_id

; 8) Запишемо зміст буфера у файл.
; Функція DOS 40h: Запис у файл або пристрій.
        mov         ah, 40h                     ; AH = 40h.
        mov         bx, Create_File_id          ; BX = ідентифікатор.
        mov         cx,  6                      ; CX = число байт(у моєму ім'ї 6 літер).
        mov         dx, offset buffer           ; DS:DX = адреса буфера з даними
        int         21h

; 9) Закриємо файл після запису даних.
; Закриття файлу.
mov     bx, Create_File_id              ; Ідентифікатор файлу у BX. mov         ah, 3Eh
          ; Номер функції для закриття файлу в AH.
        int         21h

;=== Вихід із програми.===
        call        ExitProgram                 ; Вихід із програми.
; Мітка для аварійного виходу із програми у випадку помилки читання/запису. Exit_If_Error_File_Read:
        mov         DX,offset ErrorFileRead
        call        writeStr                    ; Вивід рядка.
        mov         DX, offset MessageUpExit
        call        writeStr                    ; Вивід рядка.
        call        waitPressing                ; Затримує вихід із програми.
        call        ExitProgram                 ; Вихід із програми.

;=== writeStr ===
; Процедура. Виводить рядок символів на екран.
; Вхід: DX = <адреса рядка в пам'яті> (символ закінчення рядка '$'). ; Вихід: немає.
 writeStr       PROC
        push        ax                          ; Збереження регістру AX.
        mov         AH,09h                      ; Параметр функції 21h - вивід рядка.
        int 21h                                 ; Виклик функції DOS 21h.
        pop         ax                          ; Відновлення регістру AX.
 ret
 writeStr       ENDP
;=== End writeStr ===

;=== waitPressing ===
```

```
;Процедура. Затримує вихід із програми для перегляду результату.
;Вхід: немає
;Вихід: AL - введений символ waitPressing       PROC
        push    ax              ; Збереження регістру AX.
        mov     ah,8            ; Параметр функції 21h - читання символу.
        int     21h             ; Виклик функції DOS 21h.
        pop     ax              ; Відновлення регістру AX.
ret
waitPressing    ENDP
 ;=== End waitPressing ===

ExitProgram   PROC mov      AX,4C00h int    21h ret         ExitProgram ENDP end        main
```

3. Зробіть трансляцію й компонування програми.
   ml /c /Zi /Fl lab13.asm link /co lab13.obj;

Питання до захисту роботи: сутність використання переривання MS DOS і BIOS.

Вказівки до змісту звіту

1. Текст програми. Кожний оператор супроводити коментарями.
2. Результат роботи програми.

Варіанти лабораторної роботи № 13

Як варіанти завдань використовуйте свої особисті дані на латиниці:
ПРІЗВИЩЕ ІМ'Я ПО БАТЬКОВІ.

Лабораторна робота № 14. Використання команд співпроцесора

Тема: використання команд співпроцесора.

Мета роботи: навчити здобувачів використовувати команди співпроцесора.

Устаткування: персональний комп'ютер.

Програмне забезпечення: Windows 7/8/10, DOSBoxPortable, MS MASM 6.x.

Хід роботи

1. Створити програму .
2. При реалізації програми кожний рядок повинен супроводжуватися коментарями.
3. При виконанні програми значення змінних вибрати самостійно.
4. Перевірку програми виконати для різних комбінацій змінних.
5. Зберегти програму в папку для lab14.asm.
6. Записати всі дані, необхідні для оформлення звіту.
7. Закрити всі програми.

Завдання

Завдання: створити програму, яка обчислює значення виразу **(c*(a+b)) / (d-e).**

Результат обчислення записати в змінну Rezultat_Expression

1. Використовуючи текстовий редактор, створіть файл lab14.asm у папці D:\PROJECT.
2. Створити програму, що обчислює вираз, що складається з дійсних чисел, за допомогою команд математичного співпроцесора.

```
TITLE ==== Програма. Обчислення виразу за допомогою співпроцесора. ==== .model small
.stack 100h .data
; Константи для обчислення виразу.
val_A dd 25.056 val_B dd 128.16 val_C dd 4.7 val_D dd 228.47 val_E dd 226.47
Rezultat_Expression dd ?                    ; Змінна для зберігання результату.
ConditionProblem db 'The program calculates the value of the expression:',13,10,'$'
Expression db '(4,7*(25,056 + 128,16))/(228, 47-226,47)',13,10,'$'
RezInfo db 'To view the results, use a debugger CV.EXE',13,10,'$'
MessageUpExit db 10,13,10, 'Press any key to exit ...',13,10,'$'
.code .386 main:
mov AX,@data
mov DS,AX

;==================== Вивід умови завдання
```

```
mov DX,offset ConditionProblem call     ; Адреса рядка з умовою в DX.
writeStr ; Вивід рядка.
mov DX,offset Expression        call    ; Адреса рядка з умовою в DX.
writeStr ; Вивід рядка.
mov DX,offset RezInfo                   ; Адреса рядка з умовою в DX.
call writeStr                           ; Вивід рядка.

;****************** ОБЧИСЛЕННЯ ВИРАЗУ *********************
finit                                   ; Приведення співпроцесора в початковий стан. fld val_A
                        ; Завантаження числа з пам'яті на
; вершину стеку співпроцесора. ST(0)=25.056
fadd val_B                              ; Додавання значень регістру ST(0) і змінної val_B.
; ST(0)=ST(0)+val_B=25.056+128.16=153.216
fld val_C                               ; Завантаження речовинного числа з пам'яті на
; вершину стеку співпроцесора. ST(0)=4.7; ST(1)=153.216
fmul                                    ; Множимо регістри ST(0) і ST(1):
                                ; ST(0)= ST(0)* ST(l)=4.7*153.216=720.1152
fld val_D                               ; Завантаження речовинного числа з пам'яті на
                                    ; вершину стеку співпроцесора. ST(0)=228.47; ST(1)=720.1152
fsub val_E                              ; Вирахування значень регістру ST(0) і змінної val_E.
; ST(0)=ST(0)-val_E=228.47-226.47=2; ST(1)=720.1152
Fdiv                                    ; Ділення ST(1) на ST(0):
; ST(0)= ST(1)\ST(0)=720.1152\2=360.0576  ; Запис результату з регістру ST(0) у змінну
Rezultat_Expression.
fst Rezultat_Expression

;************************************************************
;=========================Вивід підказки користувачу й вихід із програми. mov DX,offset
MessageUpExit call writeStr                     ; Вивід рядка. call waitPressing                ;
Затримує вихід із програми. call ExitProgram                    ; Вихід із програми.

;================== writeStr ============== ; Процедура. Виводить рядок символів на екран.
; Вхід: DX = <адреса рядка в пам'яті> (символ закінчення рядка '$').
; Вихід: немає.

writeStr PROC push ax                           ; Збереження регістру AX. mov AH,09h
                        ; Параметр функції 21h - вивід рядка. int 21h                   ; Виклик
функції DOS 21h. pop ax                         ; Відновлення регістру AX. ret
writeStr ENDP
;================== End writeStr ==============

;================== waitPressing ==============
;Процедура. Затримує вихід із програми для перегляду результату.
;Вхід: немає
;Вихід: AL - введений символ

waitPressing PROC push ax                       ; Збереження регістру AX. mov ah,8
                ; Параметр функції 21h - читання символу. int 21h                       ; Виклик
функції DOS 21h. pop ax                         ; Відновлення регістру AX. ret
waitPressing ENDP

;=============== End waitPressing ==============
ExitProgram PROC
mov AX,4C00h
int 21h
```

```
ret
ExitProgram ENDP end main
```

3. Зробіть трансляцію й компонування програми.
   ml /c /Zi /Fl lab12.asm link /co lab12.obj;

Результат роботи програми:

C:\Masm615>C:\Masm615\p.exe;
The program calculates the value of the expression:
(4,7» (25,056*128,16)1/(228, 47-226,47)
To view the results, use a debugger TD.EXE Press any key to exit...

Результат роботи відладчика

Результати обчислень можна подивитися за допомогою відладчика Turbo Debugger (TD.EXE). Для цього необхідно загрузити в нього програму (lab14.exe), прогнати її. Після чого можна подивитися результати обчислень: меню View Numeric processor.

Питання до захисту роботи: команди пересилання даних математичного співпроцесора – перелік, сутність, методи використання, арифметичні команди математичного співпроцесора.

Вказівки до змісту звіту

1. Текст програми. Кожний оператор супроводити коментарями.
2. Результат роботи програми.

Варіанти завдань для лабораторної роботи № 14

№ вар.	
1.	Rezultat_Expression=14*(200– a+b)/c^2
2.	Rezultat_Expression=10/(a+b+c)–1
3.	Rezultat_Expression=(24*a)/(b^2+c^4)
4.	Rezultat_Expression=5/(a*b)–c
5.	Rezultat_Expression=(a–b)*(10–c)
6.	Rezultat_Expression=(b+c)/a^2
7.	Rezultat_Expression=a^2/(b^2–c)
8.	Rezultat_Expression=(a^2+)/(b–1)
9.	Rezultat_Expression=(a–b)/(c–a)

10.	Rezultat_Expression=$(c^2+1)/(a^3–b)$
11.	Rezultat_Expression=$(a^2+2)/(b^2–c)$
12.	Rezultat_Expression=(a–2*b)/(c–a)
13.	Rezultat_Expression=$(a/b+c)^2$
14.	Rezultat_Expression=$(a+b)/(b–c)^2$
15.	Rezultat_Expression=$(a–b)/(c^2+1)$

МЕТОДИ ОЦІНЮВАННЯ

Результати захисту лабораторних робіт здобувачів виставляються у вигляді оцінки за національною шкалою.

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

Основна література

1. Мікропроцесорна техніка: Лабораторний практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності 171 «Електроніка» / КПІ ім. Ігоря Сікорського; уклад.: Т. О. Терещенко, Л. М. Батрак, Ю. С. Ямненко. – Електронні текстові дані (1 файл: 1,51 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2021. – 76 с.
2. Левченко Л. О. Операційні системи [Електронний ресурс] : навч. посіб. для здобувачів ступеня бакалавра за освіт. програмою «Цифрові технології в енергетиці» спец. 122 «Комп'ютерні науки» / Л. О. Левченко, Ю. А. Тарнавський; КПІ ім. Ігоря Сікорського. – Електрон. текст. дані (1 файл. – Київ: КПІ ім. Ігоря Сікорського, 2023. – 256 с.
3. Зілінський Ю. В., Перекрест А. Л., Юдіна А. Л. Системне програмування. Програмування на ассеблері: навч. посібник. Кременчук: Кременчуцький національний університет імені Михайла Остроградського, 2023. – 258 с.
4. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Чинний від 2016-07-01. Вид. офіц. Київ : УкрНДНЦ, 2016. –16 с.

Додаткова література

1. Основи комп'ютерної техніки та програмування мікропроцесорів: навч. посіб. / Д. О. Гололобов. – К.: Редакційно-видавничий центр Державного університету телекомунікацій, 2019. – 58 с.
2. Мельник А. О. Архітектура комп'ютера: підручник для студентів вузів. 3-вид., – Луцьк : Волинська обласна друкарня, 2018. – 470 с.
3. Тонкошкур О. С., Гниленко О. Б., Матвєєва Н. О., Морозов О. С. Архітектура комп'ютерів. Машинні команди та програмування на асемблері. Навчальний посібник, – Дніпро: «Нова Ідеологія», 2018. – 179 с.
4. Advanced Micro Devices, Inc. AMD64 Architecture Programmer's Manual Volume 1: Application Programming. Publication No. 24592. Revision Date 3.22. December 2017. – Режим доступу: https://developer.amd.com/wordpress/media/2012/10/24592_APM_v11.pdf
5. Інтернет ресурс: The MASM Forum http://masm32.com/board/index.php
6. Інтернет ресурс: https://stackoverflow.com/questions/14782902/assembly-i-o-programming
7. Nadavlevy. Functions in assembly & the stack. Medium. URL: https://medium.com/@nadavlevy1000/functions-in-assembly-the-stack-34f5c1917a81

Навчальне видання

АРХІТЕКТУРА КОМП'ЮТЕРІВ ТА НИЗЬКОРІВНЕВЕ ПРОГРАМУВАННЯ

Частина ІІ

ЕЛЕКТРОННІ МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт
для студентів першого (бакалаврського) рівня вищої освіти
спеціальності F7 Комп'ютерна інженерія

Електронне практичне видання

Укладачі:

Шаріпова Ільнара Вільївна

Берков Юрій Миколайович

В авторській редакції

Затв. авт. 19.05.2026. Шрифт Times New Roman.
Системні вимоги: операційна система сумісна з програмним забезпеченням
для читання файлів формату PDF.
Обсяг 1,1 МБ. Зам. № 3141.

Видавець:
Одеський національний університет імені І. І. Мечникова
вул. Змієнка Всеволода, буд. 2, м. Одеса, 65001, Україна
Свідоцтво суб'єкта видавничої справи ДК № 8592 від 23.03.2026 р.
Тел.: (048) 723 28 39, e-mail: druk@onu.edu.ua