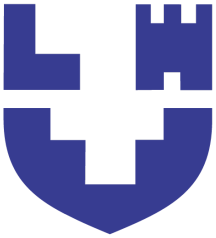
Міністерство освіти і науки України
Луцький національний технічний університет



КОДУВАННЯ ІНФОРМАЦІЇ ТА АРХІТЕКТУРА КОМП'ЮТЕРА

Методичні вказівки до лабораторних занять
для здобувачів першого (бакалаврського) рівня вищої освіти
освітньої програми «Професійна освіта (комп’ютерні технології)»
галузі знань А Освіта
спеціальності А5.39 Професійна освіта (Цифрові технології)
денної та заочної форм навчання

Луцьк 2026

УДК 004.2:004.3(07)
К57

До друку

Голова вченої ради факультету цифрових, освітніх та соціальних технологій ЛНТУ _____ Г. ГЕРАСИМЧУК

Затверджено вченою радою факультету цифрових, освітніх та соціальних технологій ЛНТУ, протокол № ____ від «____»_____ 2026 року.

Електронна копія друкованого видання передана для внесення в репозитарій ЛНТУ.
Директор бібліотеки _____ Н. ПОЛІЩУК

Рекомендовано до видання на засіданні кафедри цифрових освітніх технологій ЛНТУ, протокол № _____ від «____»_____ 2026 року.
Завідувач кафедри цифрових освітніх технологій _____ В. КАБАК

Укладач: _____ А. ЛОБАЦЬКИЙ, кандидат педагогічних наук, старший викладач кафедри цифрових освітніх технологій ЛНТУ.
Укладач: _____ П. САВАРИН, кандидат педагогічних наук, доцент кафедри цифрових освітніх технологій ЛНТУ.

Рецензент: _____ В. КАБАК, кандидат педагогічних наук, доцент кафедри цифрових освітніх технологій ЛНТУ.

Відповідальний за випуск:_____ В. КАБАК, кандидат педагогічних наук, доцент, завідувач кафедри цифрових освітніх технологій ЛНТУ.

К57 **Кодування інформації та архітектура комп'ютера** : Методичні вказівки до лабораторних занять для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Професійна освіта (комп'ютерні технології)» галузі знань А Освіта спеціальності А5.39 Професійна освіта (Цифрові технології) денної та заочної форм навчання / уклад. А. ЛОБАЦЬКИЙ, П. САВАРИН. Луцьк: ЛНТУ, 2026. 195 с.

Методичне видання складене відповідно до діючої програми курсу «Кодування інформації та архітектура комп'ютера» і містить необхідний теоретичний та практичний матеріал, який дасть можливість студентам виконати лабораторні роботи.

Зміст

Лабораторна робота №1

Системи представлення даних: азбука Морзе, шрифт Брайля

Теоретичні відомості

Азбука Морзе - телеграфна азбука, названа за ім’ям розробника Семюела Морзе, який запропонував її в 1838. Відтворення графічних знаків комбінації точок і тире - це її головна функція (рис. 1). За одиницю часу приймається тривалість однієї точки. Тривалість тире дорівнює трьом точкам. Пауза між елементами одного знака – одна точка, між знаками в слові – 3 точки, між словами – 7 точок.

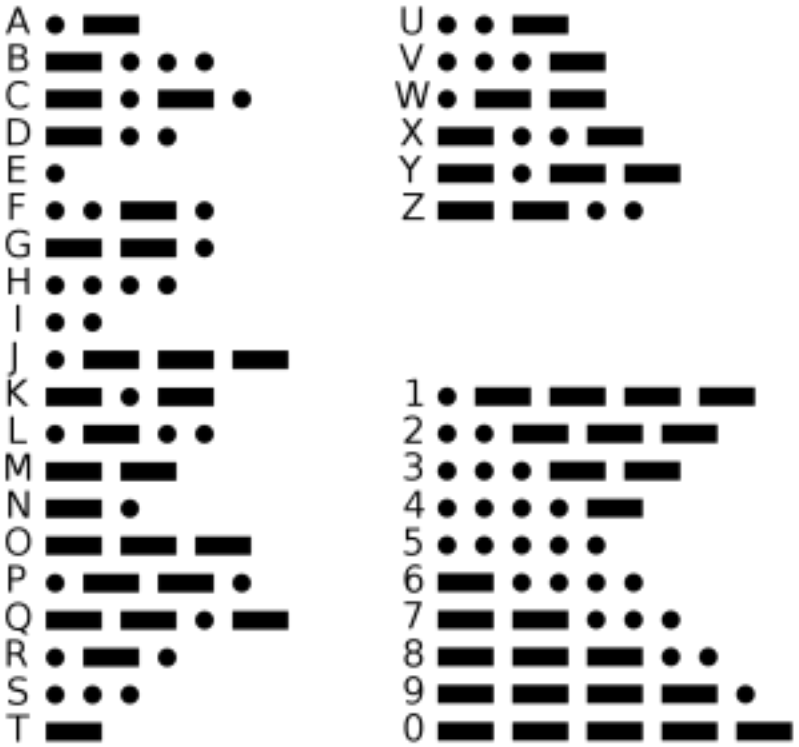


Рис. 1 - Таблиця кодів міжнародної азбуки Морзе

Принцип кодування абетки Морзе виходить з того, що літери, які найчастіше вживаються в англійській мові, кодуються простішими сполученнями крапок і тире (рис. 2). Це робить освоєння абетки Морзе простіше, а передачі – компактніше.

Абетка Морзе є першим цифровим способом передачі інформації. Телеграф та радіотелеграф спочатку використовували азбуку Морзе; пізніше стали

застосовуватися код Бодо та ASCII, які зручніші для автоматизації. Втім, зараз і для азбуки Морзе є засоби автоматичної генерації та розпізнавання. Крім того, радіоаматорами розроблена велика кількість апаратних декодерів абетки Морзе на базі мікроконтролерів.

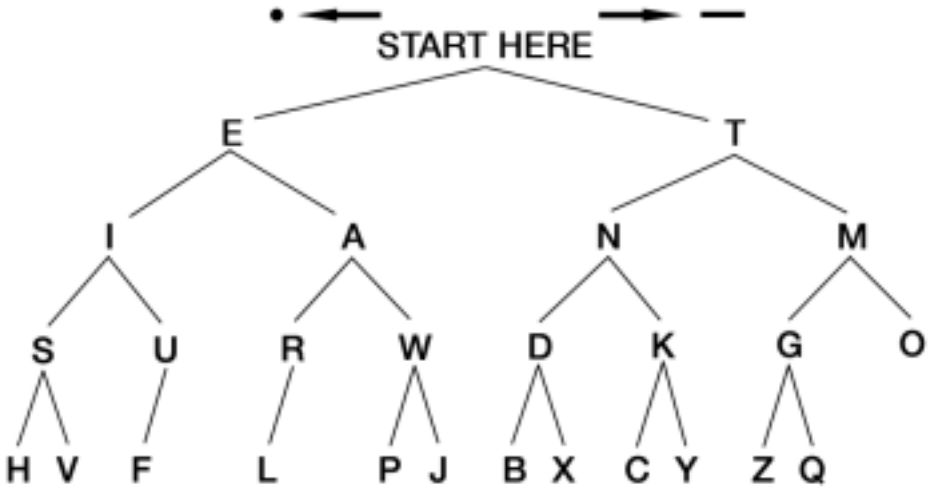


Рис. 2 Структура кодів азбуки Морзе за принципом частотності

Існує щонайменше два варіанти абетки Морзе українською мовою, які незначно відрізняються. Для передачі кирилиці використовувалися коди схожих латинських літер (таблиця 1).

Таблиця 1 - Кодування кириличних літер азбукою Морзе

Літери		Код Морзе
Латинські	Українські	
A	А	• –
B	Б	– • • •
C	Ц	– • – •
D	Д	– • •
E	Е	•
F	Ф	• • – •
G	Г, Ґ	– – •
H	Х	• • • •
I	І, Ї	• •
J	Й	• – – –
K	К	– • –
L	Л	• – • •
M	М	– –
N	Н	– •

O	О	– – –
P	П	• – – •
Q	Щ	– – • –
R	Р	• – •
S	С	• • •
T	Т	–
U	У	• • –
V	Ж	• • • –
W	В	• – –
X	Ь	– • • –
Y	И	– • – –
Z	З	– – • •
Đ, É, Ę	Є	• • – • •
Ó, Ö, Ø	Ч	– – – •
CH, Ĥ, Š	Ш	– – – –
Ü, Ŭ	Ю	• • – –
Ä, Æ, Ą	Я	• – • –

Коди цифр простіше запам’ятати, знаючи, що кожен з них складається з 5 сигналів (точок чи тире), цифри 0 – 5 мають стільки ж точок на початку, як значення цифри, а цифри 6 – 9 – стільки точок вкінці, скільки не вистачає до десяти, решта сигналів у коді – тире (таблиця 2).

Таблиця 2 - Кодування чисел азбукою Морзе

Цифра	Код
1	• – – – –
2	• • – – –
3	• • • – –
4	• • • • –
5	• • • • •
6	– • • • •
7	– – • • •
8	– – – • •
9	– – – – •
0	– – – – –

Переваги:

- висока перешкодозахищеність при прийомі на слух в умовах сильних радіоперешкод;
- можливість кодування вручну;

- запис і відтворення сигналів найпростішими пристроями.
- Недоліки:
- неекономічність, на передачу одного знака коду потрібно в середньому 9,5 елементарних посилок;
- мала придатність для букводрукуючого прийому;
- низька швидкість телеграфування.

Шрифт Брайля – рельєфно-точковий шрифт для написання і читання людьми з порушенням зору, розроблений французом Луїсом Брайлем. В основі шрифту лежить комбінація шести крапок. Брайль використав порядок літер латинського алфавіту. Для позначення перших літер алфавіту використовують верхні й середні точки. Для позначення наступних літер додається нижня точка ліворуч, потім ліворуч і праворуч, потім праворуч. Цими ж знаками позначаються і літери 8 українського алфавіту з додаванням спеціальних знаків. Різні комбінації шести точок дають можливість позначати також цифри, розділові знаки, математичні, хімічні й нотні знаки.

Завдання для виконання роботи

Завдання №1: Закодуй своє ім’я та прізвище латиницею за допомогою онлайн-ресурсу https://lucassedberg.com/tools/morse/simulator

Інструкція:

- Відкрий симулятор.
- Транслітеруйте своє ім’я та прізвище латиницею (наприклад, Шевченко → SHEVCHENKO).
- За допомогою клавіші пробіл “відстукай” своє ім’я та прізвище в азбуці Морзе.
    - коротке натискання = крапка (·)
    - довге натискання = тире (–)
    - пауза між літерами = коротка зупинка (~3 одиниці)
- Переконайся, що симулятор правильно розпізнав текст.

- Зроби скріншот результату для звіту.

Результат:

- скріншот з сайту де є ім’я та прізвище закодоване у Морзе.

Завдання №2: Розробити власну версію азбуки Морзе для українського алфавіту.

Вимоги:

- Використовуй принцип частотності: найчастіші літери → найкоротші коди;
- Побудуй таблицю виду:

Літера	Код	Обґрунтування

- Закодуй своє ім’я та прізвище за новоствореною системою.

Результат:

- Таблиця українського Морзе (твоя версія).
- Закодоване своє ім’я та прізвище, використовуючи створену азбуку.

Лабораторна робота №2

Системи числення

Теоретичні відомості

Системою числення називається сукупність правил і знаків, за допомогою яких можна відобразити (кодувати) будь-яке невід'ємне число. До систем числення висуваються певні вимоги, серед яких найбільш важливими є вимоги однозначного кодування чисел 0, 1,… з деякої їх скінченної множини Крім того, системи числення розв'язують задачу нумерації, тобто ефективного переходу від зображень чисел до номерів, які в даному випадку повинні мати мінімальну кількість цифр. Від вдалого чи невдалого вибору системи числення залежить ефективність розв'язання зазначених задач і її використання на практиці.

Розрізняють наступні типи систем числення:

- позиційні;
- непозиційні.

У позиційних системах числення одна і та ж цифра (числовий знак) у записі числа набуває різних значень залежно від своєї позиції. Накприклад, 1, 10, 100, 1000.

У непозиційних системах числення величина, яку позначає цифра, не залежить від позиції її у числі. При цьому система може накладати обмеження на позиції цифр, наприклад, щоб вони були розташовані по спаданню, чи згруповані за значенням. Проте це не є принциповою умовою для розуміння записаних такими системами чисел. Наприклад, I, II, V, X.

У інформаційних технологіях найчастіше застосовуються двійкова, десяткова, вісімкова, та шістнадцяткова системи.

Десяткова система числення – це позиційна система числення із основою 10, кожне число в якій записується за допомогою 10-ти символів, цифр – 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Запис числа формується за загальним принципом: на n-й позиції (справа наліво від 0) стоїть цифра, що відповідає кількості n-х степенів десяти в цьому числі.

Наприклад: $123456 = 1 \cdot 10^5 + 2 \cdot 10^4 + 3 \cdot 10^3 + 4 \cdot 10^2 + 5 \cdot 10^1 + 6 \cdot 10^0$

Дробова частина числа формується за таким самим принципом, тільки позиція цифри в дробовій частині відраховується від коми зліва направо починаючи з 1 і береться зі знаком «-».

Наприклад: $123{,}456 = 1\cdot10^{2} + 2\cdot10^{1} + 3\cdot10^{0} + 4\cdot10^{-1} + 5\cdot10^{-2} + 6\cdot10^{-3}$

Двійкова система числення – це позиційна система числення, база якої дорівнює двом та використовує для запису чисел тільки два символи: зазвичай 0 (нуль) та 1 (одиницю). Числа, представлені в цій системі часто називають двійковими або бінарними числами.

Для запису числа у двійковій системі числення використовується представлення цього числа за допомогою степенів числа 2.

Завдяки тому, що таку систему доволі просто використовувати в електричних схемах, двійкова система отримала широке розповсюдження у світі обчислювальних пристроїв.

Рахувати у двійковій системі не складніше, ніж у будь-якій іншій. Скажімо, у десятковій системі, коли число у поточному розряді сягає десяти, то розряд обнуляється і одиниця додається до старшого.

Наприклад: 9 + 1 = 10, 44 + 7 = 51;

Аналогічним чином у двійковій системі: коли число в розряді сягає двох – розряд обнуляється і одиниця додається до старшого розряду.

Тобто: 1 + 1 = 10.

Зверніть увагу, «10» у цьому записі – двійкове число, у десятковій системі це число записується як «2». А десяткове 9 + 1 = 10 у двійковій системі буде виглядати так: 1001 + 1 = 1010 (після додавання одиниці число в останньому розряді дорівнює двом, тож розряд обнуляється і одиниця додається до передостаннього (старшого) розряду).

Для перетворення з двійкової системи в десяткову використовують таку таблицю ступенів основи 2:

1024	512	256	128	64	32	16	8	4	2	1

Припустимо, дано двійкове число 110001_2. Для конвертування в десяткове запишіть його як суму за розрядами таким чином:

$$1 * 2^5 + 1 * 2^4 + 0 * 2^3 + 0 * 2^2 + 0 * 2^1 + 1 * 2^0 = 49$$

Теж саме але трохи по-іншому:

$$1 * 32 + 1 * 16 + 0 * 8 + 0 * 4 + 0 * 2 + 1 * 1 = 49$$

Можна записати у вигляді таблиці таким чином:

512	256	128	64	32	16	8	4	2	1
				1	1	0	0	0	1
				+32	+16	+0	+0	+0	+1

Вісімкова система числення – позиційна цілочисельна система числення з основою 8. Для представлення чисел в ній використовуються цифри від 0 до 7. Вісімкова система часто використовується в галузях, пов'язаних з цифровими пристроями. Характеризується легким переводом вісімкових чисел у двійкові і назад, шляхом заміни вісімкових чисел на триплети двійкових.

$0_8 = 000_2$

$1_8 = 001_2$

$2_8 = 010_2$

$3_8 = 011_2$

$4_8 = 100_2$

$5_8 = 101_2$

$6_8 = 110_2$

$7_8 = 111_2$

Для переведення вісімкового числа в двійкове необхідно замінити кожну цифру вісімкового на триплет двійкових цифр. Наприклад: 2541_8 = 010 101 100 001 = 010101100001_2.

Шістнадцяткова система числення – це позиційна система числення з основою 16. Тобто кожне число в ній записується за допомогою 16 символів. Арабські цифри від 0 до 9 відповідають значенням від нуля до дев'яти, а 6 літер латинської абетки A, B, C, D, E, F відповідають значенням від десяти до

п’ятнадцяти. Шістнадцяткова система числення широко використовується розробниками комп’ютерів та програмістами.

Цю систему часто називають також Hex (hexadecimal – шістнадцятковий).

Для переведення шістнадцяткового числа в десяткове необхідно це число представити у вигляді суми добутків ступенів основи шістнадцяткової системи числення на відповідні цифри в розрядах шістнадцяткового числа.

Наприклад, треба перевести шістнадцяткове число 5A3 в десяткове. В цьому числі 3 цифри. У відповідності з наданим правилом представимо його у вигляді суми ступенів з основою 16:

$$5A3_{16} = 3\cdot16^0+10\cdot16^1+5\cdot16^2 =$$

$$= 3\cdot1+10\cdot16+5\cdot256 = 3+160+1280 = 1443_{10}$$

Для переведення багатозначного числа у шістнадцяткову систему треба розбити його на тетради справа наліво та замінити кожну тетраду відповідною шістнадцятковою цифрою. Для переведення числа з шістнадцяткової системи у двійкову треба замінити кожну його цифру на відповідну тетраду з наведеної нижче таблиці переведення.

Наприклад:

$$010110100011_2 = 0101\ 1010\ 0011 = 5A3_{16}$$

Завдання для виконання роботи

Завдання №1:Перетворити число у десяткову форму:

1	$101010_2 \rightarrow X_{10}$ $3257_8 \rightarrow X_{10}$ $A7B_{16} \rightarrow X_{10}$	1	$010110_2 \rightarrow X_{10}$ $2460_8 \rightarrow X_{10}$ $F34_{16} \rightarrow X_{10}$
2	$110011_2 \rightarrow X_{10}$ $1746_8 \rightarrow X_{10}$ $1D2_{16} \rightarrow X_{10}$	2	$101001_2 \rightarrow X_{10}$ $2173_8 \rightarrow X_{10}$ $8D9_{16} \rightarrow X_{10}$
3	$001100_2 \rightarrow X_{10}$ $6432_8 \rightarrow X_{10}$ $E4F_{16} \rightarrow X_{10}$	3	$001110_2 \rightarrow X_{10}$ $4371_8 \rightarrow X_{10}$ $71E_{16} \rightarrow X_{10}$
4	$111000_2 \rightarrow X_{10}$ $5071_8 \rightarrow X_{10}$ $C6A_{16} \rightarrow X_{10}$	4	$100111_2 \rightarrow X_{10}$ $6425_8 \rightarrow X_{10}$ $BFD_{16} \rightarrow X_{10}$
5	$010101_2 \rightarrow X_{10}$	5	$011100_2 \rightarrow X_{10}$

	$2165_8 \rightarrow X_{10}$ $B0F_{16} \rightarrow X_{10}$		$1076_8 \rightarrow X_{10}$ $C1E_{16} \rightarrow X_{10}$
6	$100100_2 \rightarrow X_{10}$ $4630_8 \rightarrow X_{10}$ $2D4_{16} \rightarrow X_{10}$	6	$110100_2 \rightarrow X_{10}$ $3150_8 \rightarrow X_{10}$ $85F_{16} \rightarrow X_{10}$
7	$011011_2 \rightarrow X_{10}$ $7524_8 \rightarrow X_{10}$ $B1A_{16} \rightarrow X_{10}$	7	$001010_2 \rightarrow X_{10}$ $7264_8 \rightarrow X_{10}$ $93B_{16} \rightarrow X_{10}$
8	$110110_2 \rightarrow X_{10}$ $3102_8 \rightarrow X_{10}$ $3E9_{16} \rightarrow X_{10}$	8	$100010_2 \rightarrow X_{10}$ $5013_8 \rightarrow X_{10}$ $7F5_{16} \rightarrow X_{10}$
9	$000111_2 \rightarrow X_{10}$ $6754_8 \rightarrow X_{10}$ $F8C_{16} \rightarrow X_{10}$	9	$101100_2 \rightarrow X_{10}$ $3627_8 \rightarrow X_{10}$ $6D2_{16} \rightarrow X_{10}$
10	$101101_2 \rightarrow X_{10}$ $2461_8 \rightarrow X_{10}$ $D2F_{16} \rightarrow X_{10}$	10	$111010_2 \rightarrow X_{10}$ $2706_8 \rightarrow X_{10}$ $8A4_{16} \rightarrow X_{10}$
11	$010010_2 \rightarrow X_{10}$ $1374_8 \rightarrow X_{10}$ $A9B_{16} \rightarrow X_{10}$	11	$100011_2 \rightarrow X_{10}$ $6453_8 \rightarrow X_{10}$ $B3E_{16} \rightarrow X_{10}$
12	$001001_2 \rightarrow X_{10}$ $5620_8 \rightarrow X_{10}$ $E0C_{16} \rightarrow X_{10}$		
13	$100001_2 \rightarrow X_{10}$ $4206_8 \rightarrow X_{10}$ $B5A_{16} \rightarrow X_{10}$		
14	$011010_2 \rightarrow X_{10}$ $7613_8 \rightarrow X_{10}$ $2F9_{16} \rightarrow X_{10}$		
15	$110001_2 \rightarrow X_{10}$ $1047_8 \rightarrow X_{10}$ $378_{16} \rightarrow X_{10}$		
16	$111010_2 \rightarrow X_{10}$ $5736_8 \rightarrow X_{10}$ $6AE_{16} \rightarrow X_{10}$		

Завдання №2:Перетворити десяткове число у вказану форму

1	$135 \rightarrow X_3$ $472 \rightarrow X_6$ $819 \rightarrow X_{12}$	1	$246 \rightarrow X_3$ $573 \rightarrow X_6$ $984 \rightarrow X_{12}$
2	$621 \rightarrow X_3$ $357 \rightarrow X_6$	2	$184 \rightarrow X_3$ $569 \rightarrow X_6$

	$708 \rightarrow X_{12}$		$293 \rightarrow X_{12}$
3	$846 \rightarrow X_3$ $712 \rightarrow X_6$ $430 \rightarrow X_{12}$	3	$986 \rightarrow X_3$ $547 \rightarrow X_6$ $321 \rightarrow X_{12}$
4	$765 \rightarrow X_3$ $198 \rightarrow X_6$ $634 \rightarrow X_{12}$	4	$257 \rightarrow X_3$ $840 \rightarrow X_6$ $916 \rightarrow X_{12}$
5	$372 \rightarrow X_3$ $605 \rightarrow X_6$ $128 \rightarrow X_{12}$	5	$479 \rightarrow X_3$ $743 \rightarrow X_6$ $356 \rightarrow X_{12}$
6	$918 \rightarrow X_3$ $284 \rightarrow X_6$ $691 \rightarrow X_{12}$	6	$507 \rightarrow X_3$ $362 \rightarrow X_6$ $749 \rightarrow X_{12}$
7	$128 \rightarrow X_3$ $934 \rightarrow X_6$ $672 \rightarrow X_{12}$	7	$519 \rightarrow X_3$ $403 \rightarrow X_6$ $276 \rightarrow X_{12}$
8	$865 \rightarrow X_3$ $531 \rightarrow X_6$ $917 \rightarrow X_{12}$	8	$648 \rightarrow X_3$ $372 \rightarrow X_6$ $194 \rightarrow X_{12}$
9	$586 \rightarrow X_3$ $341 \rightarrow X_6$ $729 \rightarrow X_{12}$	9	$812 \rightarrow X_3$ $465 \rightarrow X_6$ $293 \rightarrow X_{12}$
10	$670 \rightarrow X_3$ $158 \rightarrow X_6$ $426 \rightarrow X_{12}$	10	$795 \rightarrow X_3$ $301 \rightarrow X_6$ $874 \rightarrow X_{12}$
11	$652 \rightarrow X_3$ $719 \rightarrow X_6$ $348 \rightarrow X_{12}$	11	$506 \rightarrow X_3$ $927 \rightarrow X_6$ $214 \rightarrow X_{12}$
12	$683 \rightarrow X_3$ $579 \rightarrow X_6$ $431 \rightarrow X_{12}$		
13	$296 \rightarrow X_3$ $874 \rightarrow X_6$ $350 \rightarrow X_{12}$		
14	$967 \rightarrow X_3$ $218 \rightarrow X_6$ $465 \rightarrow X_{12}$		
15	$732 \rightarrow X_3$ $590 \rightarrow X_6$ $143 \rightarrow X_{12}$		
16	$687 \rightarrow X_3$ $275 \rightarrow X_6$ $864 \rightarrow X_{12}$		

Лабораторна робота №3:

Двійкове кодування даних. Додавання та віднімання двійкових чисел

Теоретичні відомості

Двійкова система числення є основою роботи будь-якого комп’ютера. Вона ґрунтується лише на двох станах: **0 і 1**. Ці два значення легко реалізувати на фізичному рівні: наявність або відсутність електричного струму, заряджений чи розряджений конденсатор, намагнічена чи намагнічена ділянка носія. Завдяки такій простоті й надійності саме двійкова система стала фундаментом сучасної цифрової техніки.

Біт і байт.

Найменшою одиницею інформації в комп’ютері є **біт** (від англ. *binary digit* - двійкова цифра). Один біт може зберігати лише два стани - 0 або 1. Для зручності біти групуються: 8 біт утворюють **байт**. Байти є основними «цеглинками» зберігання та опрацювання інформації: 1 кілобайт містить 1024 байти, 1 мегабайт - 1024 кілобайти тощо.

Подання реальних даних.

Дані, з якими ми працюємо щодня - числа, літери, зображення чи звук - також зводяться до набору нулів і одиниць. Наприклад:

- **Числа.** У двійковій системі десяткове число 13 виглядає як 1101 (рис. 1).

$$13_{10} = 1101_2$$

$$1101_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

$$= 8 + 4 + 0 + 1 = 13$$

Рис. 1 - Приклад перетворення десяткового числа 13 у двійковий код

- **Символи.** Літери та знаки кодуються спеціальними таблицями (ASCII, Unicode), де кожному символу відповідає певна двійкова комбінація (рис 2).

Символ	Десятковий код (ASCII)	Двійкове представлення
A	65	01000001
B	66	01000010
a	97	01100001
b	98	01100010
!	33	00100001

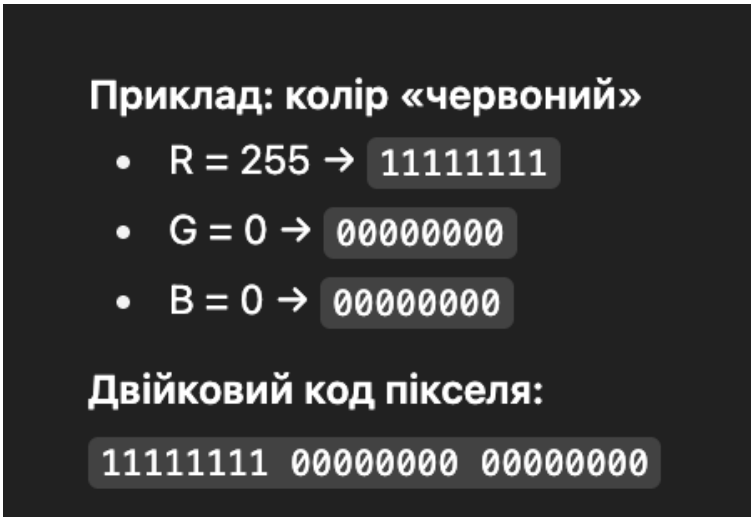
Рис. 2 - Відповідність символів десятковим та двійковим кодам

- **Зображення.** Будь-яке фото складається з пікселів, кожен з яких має колір, закодований у двійковому вигляді (наприклад, 24 біти для трьох складових RGB).

Приклад кодування зображення у двійковій системі

Будь-яке цифрове фото складається з **пікселів**. Кожен піксель має колір, який формується з трьох складових: **R (Red), G (Green), B (Blue)**. Для збереження кожної складової зазвичай використовується **8 біт** (один байт).

Отже, на колір одного пікселя потрібно реалізувати перетворення (рис 1):



- **Звук.** Аналогові звукові хвилі оцифровуються - їх амплітуди вимірюються через короткі проміжки часу й записуються як двійкові числа.

  Можна уявити це так:

1. **Реальні дані (буква, звук, колір пікселя)** → отримують **умовне числове значення** (у десятковій системі)
    - Літера **A** → код 65.
    - Амплітуда звуку → число від 0 до 255.
    - Колір → три числа для RGB (наприклад, 255, 0, 0 для червоного).
2. Потім ці числа переводяться у **двійкову систему** (бо комп'ютер працює тільки з нулями та одиницями).
    - 65 → 01000001.
    - 200 → 11001000.
    - 255 → 11111111.
3. Усі подальші обчислення, зберігання й передача відбуваються **всередині двійкової системи**.
4. Коли потрібно вивести результат для людини - двійковий код знову **перетворюється назад у «зрозумілий» вигляд**: букви, картинку, музику.

Тобто десяткова система тут використовується радше для зручності людини (нам легше працювати з 65, ніж з 01000001), але всередині комп'ютера все - лише двійкові комбінації.

Таким чином, двійковий код є універсальною мовою для комп'ютера. Незалежно від того, чи ми вводимо текст, слухаємо музику, переглядаємо відео чи працюємо з програмами, у пам'яті машини все це існує як комбінації нулів і одиниць.

Додавання двійкових чисел

У комп'ютері вся інформація - текст, числа, зображення, звук - зрештою зберігається у вигляді послідовностей нулів і одиниць (**бітів**). Саме над цими двійковими даними працює центральний процесор.

Арифметичні операції (додавання та віднімання) в двійковій системі є фундаментальними, так як:

- **Базові обчислення процесора.** Усі складніші математичні дії (множення, ділення, робота з числами з плаваючою комою) зводяться до багаторазових додавань і віднімань у двійковому вигляді.
- **Робота електронних схем.** Логічні елементи («І», «АБО», «НЕ») реалізують саме бінарні операції, і правила додавання/віднімання є прямим відображенням роботи цих схем.
- **Ефективність і швидкість.** Оскільки комп'ютер може оперувати мільярдами операцій на секунду, оптимізація навіть таких простих дій критично впливає на продуктивність.
- **Універсальність.** Розуміння двійкової арифметики дозволяє студентам бачити, як на низькому рівні відбувається робота програм, шифрування даних, передача сигналів у мережі.

Для нас у десятковій системі «+» і «–» здаються природними діями з числами. Для комп'ютера ж це - **послідовність елементарних маніпуляцій з бітами**: переносів, запозичень та зміни станів 0/1 у конкретних комірках пам'яті.

У комп'ютері всі обчислення виконуються у **двійковій системі числення**, де використовуються лише цифри 0 та 1. Кожна цифра у числі займає певну **позицію - розряд**. У двійковій системі розряди відповідають степеням числа 2:

- найправіший розряд = 2^0 (одиниці),
- наступний = 2^1 (двійки),
- далі = 2^2 (четвірки), 2^3 (вісімки) і т. д.

Біт (від *binary digit*) - це найменша одиниця інформації, що може бути 0 або 1. **Байт** = 8 біт (8 розрядів). Наприклад, число 01100100 містить 8 розрядів, тобто 1 байт, і відповідає десятковому числу 100.

1 байт = 8 біт

- Байти - це групування бітів для зручності.
- 8 біт (тобто 8 розрядів у двійковому числі) = 1 байт.

Наприклад:

- 00000000 → 0

- 00000001 → 1
- 11111111 → 255

Отже, 1 байт може зберігати значення від 0 до 255.

Правила двійкового додавання:

Додавання у двійковій системі виконується так само, як і у десятковій, але з урахуванням правил додавання тільки двох цифр – 0 та 1, що потребує врахування перенесення одиниці в старший розряд при сумі "1+1=10" (один та нуль) (рис. 3).

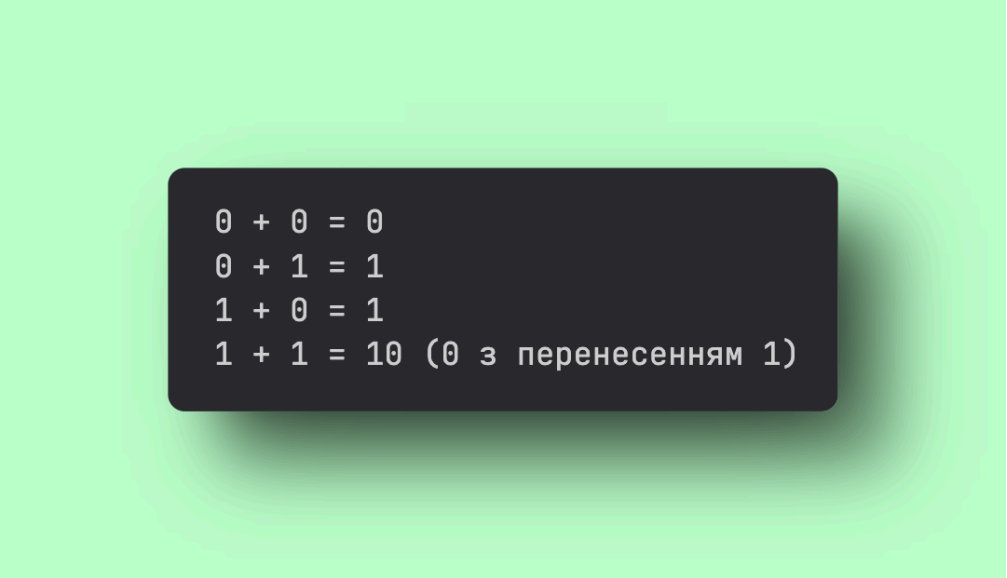


Рис. 3 - Приклад додавання в двійковій системі

Процес додавання:

1. **Вирівняйте числа:** Запишіть двійкові числа одне під одним, вирівнюючи їх за молодшим розрядом.
2. **Додавайте стовпчиком:** Починайте додавати справа наліво, розряд за розрядом.
3. **Враховуйте перенесення:** Якщо сума в стовпчику дорівнює 2 (тобто "1+1"), запишіть 0 у поточному розряді та перенесіть одиницю в наступний, лівий розряд.
4. **Продовжуйте процес:** Повторюйте кроки 2 і 3 доти, доки не додасте всі розряди (рис 4).

Рис. 4 - Приклад: Додамо двійкових чисел 101 та 11

- **Розряд 0 (найправіший):**1 + 1 = 10. Записуємо 0, переносимо 1 в наступний розряд.
- **Розряд 1:**0 + 1 + 1 (перенесена) = 10. Записуємо 0, переносимо 1 в наступний розряд.
- **Розряд 2:**1 + 1 (перенесена) = 10. Записуємо 0 і переносимо 1 в наступний розряд. Оскільки більше немає розрядів, записуємо перенесену 1.

Таким чином, результат додавання 101 та 11 становить 1000 у двійковій системі.

Віднімання двійкових чисел

Віднімання у двійковій системі числення виконується так само, як і в десятковій, шляхом позиційного віднімання, але зі специфікою двійкової системи: якщо цифра від'ємника в розряді більша за цифру зменшуваного, здійснюється **"позичання" одиниці** зі старшого розряду, яка дорівнює двом одиницям поточного розряду (*рис 5.*).

Кроки віднімання:

1. **Вирівнювання розрядів:** Допишіть нулі зліва до меншого числа, щоб обидва числа мали однакову кількість розрядів.

2. **Віднімання по розрядах:** Починайте віднімання з найправішого (молодшого) розряду.
    - **0 - 0 = 0**: Залиште 0.
    - **1 - 0 = 1**: Залиште 1.
    - **1 - 1 = 0**: Залиште 0.
    - **0 - 1**: Якщо ви зустріли таке, "позичте" одиницю з найближчого старшого розряду, що містить одиницю. Ця позичена одиниця перетворюється на дві одиниці в поточному розряді. Виконуйте віднімання: (2 + 0) - 1 = 1.
    - **Якщо старший розряд також містить 0:** Продовжуйте "позичати" одиницю, доки не знайдете розряд з одиницею.
3. **Результат:** Запишіть результат віднімання від молодшого розряду до старшого.

Рис 5. Приклад віднімання 10 від 15 в двійковій формі.

- **Молодший розряд:**1 - 0 = 1.
- **Другий розряд:** 1 - 1 = 0.
- **Третій розряд:** 1 - 0 = 1.
- **Старший розряд:** 1 - 1 = 0.

Результат: 101 (у двійковій системі).

Пряме двійкове віднімання може бути незручним при реалізації в апаратурі через необхідність обробки позик. **Комп'ютери часто уникають окремої операції віднімання, використовуючи інший підхід – додавання в спеціальному коді.**

Використання доповняльного коду при відніманні

У цифрових пристроях віднімання двійкових чисел найчастіше виконують через додавання з використанням **доповняльного коду** (two's complement). Доповняльний код – це спосіб представлення від'ємних чисел, який дозволяє виконувати віднімання як додавання, спрощуючи апаратну реалізацію операцій[7]. Іншими словами, замість окремої команди віднімання можна використовувати команду додавання навіть для роботи зі від'ємними числами – це зменшує вимоги до архітектури комп'ютера[8].

Принцип: щоб відняти число *B* від *A* (тобто обчислити A – B), можна взяти двійкове представлення числа *B*, отримати його доповняльний код (що еквівалентно -B), і додати його до *A*. Формально: $A - B = A + (-B)$. При такому додаванні, якщо виходить перенос за межі старшого розряду, він просто відкидається (не впливає на результат), оскільки в доповняльному коді розряду переносу немає значущої інтерпретації[9][10].

Кроки віднімання за допомогою доповняльного коду:

1. **Перевести обидва числа в двійковий формат.** Всі числа мають бути представлені в двійковому форматі, зазвичай з однаковою кількістю бітів, заповнюючи прогалини нулями зліва, якщо необхідно.
2. **Знайти доповняльний код від'ємника.** Щоб отримати доповняльний код від'ємного числа:
    - Інвертуйте всі біти числа (замініть 0 на 1 і 1 на 0). Це називається "перше доповнення" або "доповнення до одиниці".
    - Додайте 1 до отриманого числа.

3. **Виконати двійкове додавання.** Додайте зменшуване до отриманого доповняльного коду від'ємника.
4. **Інтерпретувати результат.**
    - Якщо відбулося переповнення (одиниця, що "виходить" за найстарший біт), її слід відкинути.
    - Результат додавання буде в доповняльному коді. Якщо результат додатний, він буде представлений своїм двійковим значенням. Якщо результат від'ємний, він буде представлений у вигляді доповняльного коду.

Приклад:

Відняти 5 від 10 (10 - 5).

1. **Двійковий формат:**
    - 10 = 1010
    - 5 = 0101 (використовуємо 4 біти для прикладу)
2. **Знайти доповняльний код -5:**
    - Візьмемо 5 (0101)
    - Інвертуємо біти: 1010 (перше доповнення)
    - Додаємо 1: 1010 + 1 = 1011. Отже, -5 в 4-бітному доповняльному коді це 1011.
3. **Виконати додавання:**
    - 1010 (10) + 1011 (-5) = 10101.
4. **Інтерпретувати результат:**
    - Оскільки ми використовуємо 4 біти, "викидаємо" найстаршу одиницю переповнення.
    - Результат: 0101 (5). 10 - 5 = 5.

Переваги доповняльного коду: Two’s complement (доповняльний код) – найпоширеніший спосіб представлення цілих від'ємних чисел у комп'ютерах[7]. Його ключова перевага – уніфікація операцій: додавання і віднімання виконуються одним і тим же блоком (суматором), що спрощує апаратну реалізацію і підвищує швидкодію системи[8]. При використанні доповняльного коду процесор може

замість операції *A – B* виконати *A + (-B)*, і апаратно це виглядає так само, як додавання (лише з інвертором і керованим переносом на вході). Це зменшує кількість необхідних логічних елементів і спрощує архітектуру. Фактично, **додавання та віднімання у two's complement реалізуються одним і тим же апаратним вузлом**, що знижує апаратні витрати (кількість вентилів), а отже, і підвищує надійність та ефективність[14][15].

Практичний підсумок: У доповняльному коді **віднімання** зводиться до **додавання** з інвертованим від'ємним числом. Цей метод широко використовується, бо усуває потребу мати окремий віднімач – достатньо модифікувати вхід другого операнду і початковий перенос у суматорі.

Завдання для виконання роботи

Завдання №1: Виконайте додавання та віднімання двійкових чисел згідно варіантом

Варіант	Приклади
1	Додавання: 101 + 11, 1101 + 111 Віднімання: 1001 – 101, 1111 – 1010
2	Додавання: 1001 + 101, 1110 + 1011 Віднімання: 1010 – 111, 1100 – 1010
3	Додавання: 111 + 11, 1010 + 1111 Віднімання: 1101 – 101, 1111 – 110
4	Додавання: 110 + 101, 1111 + 111 Віднімання: 10000 – 1111, 10101 – 1001
5	Додавання: 1011 + 1001, 1100 + 1010 Віднімання: 1110 – 1001, 1101 – 1011
6	Додавання: 11101 + 1011, 10011 + 1110 Віднімання: 10110 – 1101, 11111 – 1010
7	Додавання: 10110 + 1101, 11101 + 1111 Віднімання: 11011 – 10101, 10001 – 111
8	Додавання: 10001 + 10111, 11111 + 11010 Віднімання: 10101 – 10011, 11100 – 1101
9	Додавання: 101010 + 11101, 110011 + 11110 Віднімання: 11111 – 10101, 11010 – 1111
10	Додавання: 111000 + 1111, 101111 + 11001 Віднімання: 100111 – 1011, 111100 – 11010

Завдання №2: Виконайте операції з байтами: подання чисел у двійковій системі та виконання додавання/віднімання (варіантні значення A і B) згідно варіантом.

- Подайте цифри у двійковій системі (1 байт = 8 біт):
- Виконайте операції: додавання та віднімання
- Виконайте додавання A+B у двійковій системі
- Виконайте просте віднімання A−B у двійковій системі
- Переведіть обидва результати (сума та різниця) назад у десяткову систему.

Варіант	Дані для розрахунку
1	A=25, B=64; A=127, B=25
2	A=64, B=27; A=150, B=75
3	A=127, B=56; A=95, B=42
4	A=150, B=75; A=200, B=40
5	A=200, B=40; A=220, B=30
6	A=95, B=42; A=180, B=60
7	A=140, B=100; A=127, B=50
8	A=75, B=25; A=64, B=13
9	A=220, B=30; A=140, B=100
10	A=180, B=60; A=75, B=25

Завдання №3: Виконайте віднімання у 8-бітному доповняльному коді:

- Запишіть у 8-бітному форматі число (–B) у доповняльному коді.
- Виконайте віднімання за допомогою доповняльного коду.
- Переведіть обидва назад у десяткову систему.

Варіант	Дані для розрахунку
1	A=12, B=5; A=10, B=13
2	A=25, B=17; A=20, B=27
3	A=50, B=27; A=40, B=55
4	A=30, B=12; A=15, B=22
5	A=120, B=75; A=95, B=105
6	A=127, B=64; A=100, B=115
7	A=64, B=33; A=45, B=60

8	A=80, B=35; A=70, B=85
9	A=110, B=60; A=85, B=127
10	A=90, B=48; A=100, B=120

Завдання №4: Закодуйте смайл у двійковій системі

Виконайте побітне кодування, Записавши *матрицю смайлика (рис. 6) у вигляді двійкових рядків:*

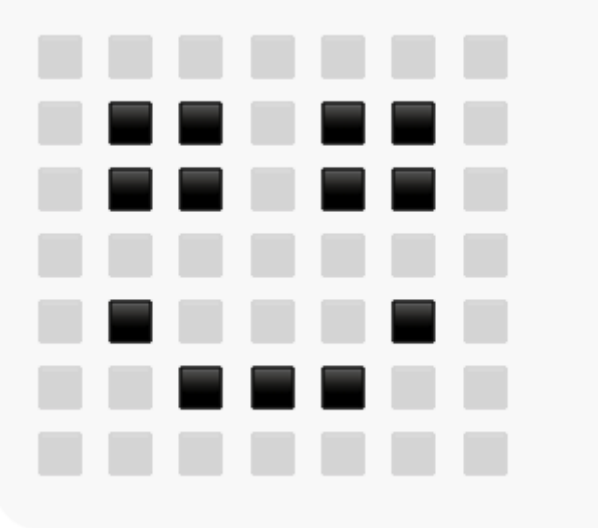


Рис. 6 - Зображення матриці смайлика

- Щоб створити чорно-біле зображення, переведіть кожен рядок у 8-бітний код (доповніть зліва нулями).
    - Наприклад, 0110010 → 00110010.
    - Побудуйте чорно-біле зображення:
        - 0 → білий (255 255 255)
        - 1 → чорний (0 0 0)
    - Запишіть зображення рядки у вигляді послідовності байтів.
        - Перший рядок → лише «білі» байти.
        - У другому рядку мають з’явитись «чорні» байти на позиціях очей.

Лабораторне заняття №4

Двійкова арифметика з дробовими і від’ємними числами

Теоретичні відомості

Позиційна система числення з комою

У цифровій техніці та математиці позиційна система числення - це фундаментальний метод запису чисел. Головна його особливість полягає у тому, що кількісний еквівалент (або «вага») кожної цифри безпосередньо залежить від її місця - позиції (розряду) відносно розділової коми. Це означає, що одна й та сама цифра, наприклад «5», може позначати зовсім різні величини: 50, 5, 0.5 або 0.005. Все вирішує те, де саме вона розташована в структурі числа.

Щоб зрозуміти принцип, розглянемо звичне нам десяткове число **347.52** (основа системи - 10). Будь-яке дійсне число можна розкласти на суму добутків його цифр на основу системи у відповідному ступені.

Розкладемо число 347.52 за розрядами:

- Цифра 3 (сотні): знаходиться у другій позиції ліворуч від одиниць → $3 \times 10^2 = 300$
- Цифра 4 (десятки): перша позиція ліворуч → $4 \times 10^1 = 40$
- Цифра 7 (одиниці): нульова позиція → $7 \times 10^0 = 7$
- Цифра 5 (десяті): перша позиція праворуч від коми → $5 \times 10^{-1} = 0.5$
- Цифра 2 (соті): друга позиція праворуч від коми → $2 \times 10^{-2} = 0.02$

  У підсумку маємо: $300 + 40 + 7 + 0.5 + 0.02 = 347.52$

  Розділова кома є точкою відліку.
- Рух вліво (Ціла частина): Кожен крок вліво збільшує вагу розряду в 10 разів (для десяткової системи) або в 2 рази (для двійкової). Тому показники степеня зростають і є додатними (0, 1, 2…).
- Рух вправо (Дробова частина): Кожен крок вправо зменшує вагу розряду. Ми ділимо на основу системи: 1/10, 1/100, 1/1000 і так далі. Математично ділення записується через від’ємний степінь (10^{-1}, 10^{-2}, 10^{-3}…).

Цей принцип універсальний. Для двійкової системи (яку використовує комп'ютер) схема ідентична, змінюється лише основа:

- Зліва: степені двійки зростають (вага 1, 2, 4, 8…), відбувається подвоєння.
- Справа: степені двійки стають від'ємними, відбувається поділ навпіл (вага 1/2, 1/4, 1/8…).

Правила перетворення дробів з десяткової системи у двійкову

Процес переведення дійсного числа складається з двох окремих етапів, оскільки ціла та дробова частини обробляються за різними алгоритмами.

1. **Ціла частина:** переводиться методом послідовного **ділення** на 2 (записуємо остачі у зворотному порядку).
2. **Дробова частина:** переводиться методом послідовного **множення** на 2.

Детальний алгоритм для дробової частини:

1. Множимо дробову частину числа на 2.
2. Аналізуємо отриманий результат:
   - Якщо результат ≥ 1, то у двійковий запис (біти після коми) заносимо **1**, а від результату віднімаємо цілу частину.
   - Якщо результат < 1, то у двійковий запис заносимо **0**.
3. Продовжуємо множити отриману дробову частину (залишок) на 2.
4. **Умова зупинки:** процес завершується, коли дробова частина стає рівною нулю, АБО коли ми отримали необхідну кількість бітів точності (якщо дріб нескінченний).

Практичні приклади перетворення

Приклад 1: Переведемо число 12.375_{10}

- *Крок 1 (Ціла частина):* $12_{10} = 1100_2$ (це стандартне переведення).
- *Крок 2 (Дробова частина 0.375):*
  - $0.375 \times 2 = 0.75 \rightarrow$ ціла частина 0
  - $0.75 \times 2 = 1.5 \rightarrow$ ціла частина 1 (залишаємо 0.5)
  - $0.5 \times 2 = 1.0 \rightarrow$ ціла частина 1 (залишок 0, стоп)
- *Результат:* Збираємо отримані біти згори вниз: .011
- Відповідь: $12.375_{10} = 1100.011_2$

Приклад 2: Проблема нескінченних дробів (число 0.1_{10})

Спробуємо перевести, здавалося б, просте число 0.1:

- $0.1 \times 2 = 0.2 \rightarrow 0$
- $0.2 \times 2 = 0.4 \rightarrow 0$
- $0.4 \times 2 = 0.8 \rightarrow 0$
- $0.8 \times 2 = 1.6 \rightarrow 1$ (залишок 0.6)
- $0.6 \times 2 = 1.2 \rightarrow 1$ (залишок 0.2)
- $0.2 \times 2 = 0.4 \rightarrow 0$ … *(ми повернулися до того, що вже було!)*

Процес зациклився. Це означає, що ми отримали нескінченний періодичний дріб: $0.1_{10} = 0.0001100110011\ldots_2$

Важливе зауваження: Не всі дроби, які є скінченними у десятковій системі, будуть скінченними у двійковій.

- Дріб буде скінченним у двійковій системі тільки тоді, коли його знаменник є степенем двійки (1/2, 1/4, 3/8 тощо).
- Якщо у знаменнику є прості множники, відмінні від 2 (наприклад, 5 у числі 1/10), такий дріб у двійковій системі стане нескінченним періодичним.

Будь-який десятковий дріб можна представити у двійковому коді, але результат може бути різним:

1. Точним (наприклад, $0.5 = 0.1_2$).
2. Нескінченним/наближеним (наприклад, $0.1 = 0.00011\ldots_2$).

Крім того, одне й те саме число математично можна записати у безлічі форм (наприклад, $0.0357 = 357 \times 10^{-4} = 3.57 \times 10^{-2}$). Така варіативність створює проблему для комп'ютерів, адже апаратура потребує чіткого, уніфікованого стандарту для зберігання даних.

Саме тому в цифровій техніці були розроблені спеціальні формати представлення дійсних чисел, які ми будемо вивчати далі:

- Числа з фіксованою комою (Fixed Point): де місце коми жорстко визначене заздалегідь.

- Числа з плаваючою комою (Floating Point): де кома може "зміщуватися" завдяки використанню порядкового степеня (експоненти), що дозволяє охопити значно ширший діапазон величин.

Робота з числами з фіксованою точкою

Числа з фіксованою комою (Fixed-Point) - це спосіб комп'ютерного представлення дійсних чисел, при якому позиція розділової коми (крапки) є строго визначеною та незмінною для всієї програми або конкретного типу даних.

Механізм зберігання:

- Розрядна сітка числа умовно поділяється на дві частини:
  - Ціла частина: старші біти.
  - Дробова частина: молодші біти.
- Фізично у пам'яті комп'ютера кома не зберігається. Процесор обробляє таке число як звичайне ціле, а «місце коми» існує лише як умовна домовленість програміста або архітектури системи.

Зв'язок з позиційною системою

Формат з фіксованою комою базується на принципах двійкової позиційної системи. Вага кожного біта визначається його відстанню від умовної коми:

- Зліва від коми (ціла частина): вага бітів зростає за степенями двійки (2^0, 2^1, 2^2…).
- Справа від коми (дробова частина): вага бітів зменшується (2^{-1}, 2^{-2}, 2^{-3}…).

Таким чином, формат Fixed-Point - це, по суті, звичайне ціле число, помножене на масштабний коефіцієнт 2^{-n} (де n - кількість дробових бітів).

Нотація Qm.n

Для опису формату використовується позначення **Qm.n**, де:

- **Q** - вказує на дробовий тип (від англ. *Quotient*).
- **m** - кількість бітів для цілої частини (включно зі знаком).
- **n** - кількість бітів для дробової частини.

Загальна довжина слова: L = m + n.

Приклади конфігурацій:

- Q5.3: 5 бітів цілих + 3 біти дробових = 8-бітове число.
- Q7.8: 7 бітів цілих + 8 бітів дробових = 15-бітове число.
- Q1.7: 1 біт цілий + 7 бітів дробових. Такий формат забезпечує високу точність, але має дуже малий діапазон значень (від -1 до +1).

Алгоритм перетворення числа у формат Fixed-Point

Розглянемо процес кодування на прикладі формату Q5.3 (5 цілих, 3 дробових біти). Завдання: Записати число 12.375 у двійковому коді.

1. Перетворення цілої частини (12): Переводимо у двійкову систему: 12 → 01100_2 (використано 5 бітів).
2. Перетворення дробової частини (0.375): Виконуємо послідовне множення на 2:
    - $0.375 \times 2 = 0.75$ → ціла частина 0
    - $0.75 \times 2 = 1.5$ → ціла частина 1 (залишок 0.5)
    - $0.5 \times 2 = 1.0$ → ціла частина 1 (кінець) Результат дробової частини: $.011_2$.
3. Об’єднання (Конкатенація): З’єднуємо цілу та дробову частини: 01100 + 011 = 01100011_2

Результат: У пам’яті число зберігається як суцільна послідовність бітів 01100011, але програма інтерпретує його як 12.375.

Точність та дискретність

Головна особливість фіксованої коми - постійний крок дискретизації. Мінімальна зміна числа (вага молодшого біта) дорівнює 2^{-n}.

Для формату Q5.3:

- Крок (точність) = $2^{-3} = 0.125$.
- Це означає, що ми можемо записати лише числа, кратні 0.125 (наприклад: 12.125, 12.250, 12.375).

Проблема округлення: Якщо число не потрапляє у сітку дискретизації, воно округлюється до найближчого можливого значення.

- Число 11.376 буде збережено як 11.375 (втрата точності).
- Число 5.1 буде збережено як 5.125 (найближче представлення).

Арифметика у Fixed-Point

- Додавання та віднімання: Виконуються процесором як звичайні цілочисельні операції. Це головна перевага формату - висока швидкодія.
    - *Приклад (Q3.2):* 5.25 (0101.01) + 2.75 (0010.11) = 8.00 (1000.00).
- Множення та ділення: Потребують додаткових дій (масштабування), оскільки при множенні кількість дробових розрядів подвоюється, і результат може вийти за межі формату.

Обмеження: Ділема «Діапазон - Точність»

Використання фіксованої коми накладає суттєві обмеження на діапазон чисел.

Уявімо 32-бітове число (формат Q16.16):

- Ми можемо записати числа в діапазоні від -32768 до +32767.
- Точність становить 1/65536 (≈ 0.000015).

Проблема:

1. Якщо потрібно записати велике число (наприклад, 1 000 000), нам не вистачить бітів цілої частини (переповнення).
2. Якщо потрібно записати надмале число (наприклад, 0.0000001), нам не вистачить бітів дробової частини (втрата значущості, запишеться як 0).

Фіксована точка ефективна для вузького діапазону значень. Для універсальних обчислень, де зустрічаються і дуже великі, і дуже малі величини, використовують формат із плаваючою комою (Floating Point, стандарт IEEE 754), де позиція коми динамічно змінюється.

Концепція «плаваючої коми»

Термін «плаваюча кома» (floating point) описує спосіб запису дуже великих або дуже малих чисел, який є комп'ютерним аналогом **наукової нотації** в математиці.

Згадайте, як ми записуємо числа у фізиці:

- Замість **300 000 000 м/с** ми пишемо $\mathbf{3 \times 10^{8}}$.
- Замість **0.0000005 м** ми пишемо $\mathbf{5 \times 10^{-7}}$.

Тут десяткова кома не зафіксована на одному місці, а «плаває» (зміщується), щоб перед нею залишалася лише одна значуща цифра. Показник степеня (експонента) компенсує це зміщення.

У двійковій системі комп'ютера принцип аналогічний, тільки основою степеня є **2**, а не 10.

- Приклад: Число $\mathbf{118.35}_{10}$ у двійковій системі виглядає як $\mathbf{1110110.010110...}_{2}$
- У нормалізованому (науковому) вигляді це записується як: $\mathbf{1.110110010110...}_{2} \times \mathbf{2}^{6}$

Ми зсунули кому на 6 позицій вліво, і степінь двійки це зафіксував.

Структура формату IEEE 754 (32 біти / Single Precision)

Стандарт IEEE 754 визначає, як саме зберігати такі числа у 32 бітах пам'яті. Число кодується за формулою:

$$\text{Число} = (-1)^{sign} \times 1.\text{мантиса} \times 2^{(\text{експонента} - \text{зсув})}$$

Бітовий рядок складається з трьох полів:

1. **Sign (Знак, 1 біт):** Визначає полярність числа.
   - **0** → число додатне (+).
   - **1** → число від'ємне (−).
2. **Exponent (Експонента, 8 біт):** Визначає масштаб числа (на скільки зсунуто кому).

- Використовується зміщення (bias) **+127**. Це зроблено для того, щоб зберігати як додатні, так і від'ємні степені у вигляді беззнакового цілого числа (від 0 до 255).
- Якщо реальний степінь **2**, то записуємо: 2 + 127 = **129**.
- Якщо реальний степінь **-3**, то записуємо: -3 + 127 = **124**.

3. **Mantissa (Мантиса, 23 біти):** Зберігає значущі цифри числа (дробову частину після коми).
    - **Важливо:** У нормалізованому двійковому числі перша цифра завжди **1** (бо це єдина цифра, відмінна від 0). Тому цю одиницю **не записують** у пам'ять (вона мається на увазі). Це називається *прихована одиниця* (hidden bit), що дозволяє збільшити точність.

Приклади перетворення чисел у формат IEEE 754 (32 біти):

Приклад 1: Число 5.75

1. Переведення у двійкову систему: $5 = 101_2$ $0.75 = 0.11_2$ Результат: 101.11_2
2. Нормалізація: Зсуваємо кому вліво, щоб залишилась одна одиниця перед нею: $101.11_2 = 1.0111_2 \times 2^2$
3. Розрахунок полів:
    - Знак: Число додатне → 0.
    - Експонента: Степінь 2. Додаємо зміщення: 2 + 127 = 129. $129_{10} = 10000001_2$.
    - Мантиса: Беремо частину після коми (0111) і доповнюємо нулями до 23 біт. 01110000000000000000000
4. Результат (32 біти): 0 10000001 01110000000000000000000

Знак (1)	Експонента (8)	Мантиса (23)
0	10000001	01110000000000000000000

Приклад 2: Десяткове число 0.15625

1. Переведення: $0.15625_{10} = 0.00101_2$
2. Нормалізація: Зсуваємо кому вправо на 3 позиції, щоб отримати «1.» на початку: $0.00101_2 = 1.01_2 \times 2^{-3}$
3. Розрахунок полів:
    - Знак: 0 (+).
    - Експонента: Степінь -3. Зміщення: -3 + 127 = 124. $124_{10} = 01111100_2$.
    - Мантиса: Частина після коми (01). 01000000000000000000000
4. Результат: 0 01111100 01000000000000000000000

Знак (1 біт)	Експонента (8 біт)	Мантиса (23 біти)
1	01111110	10000000000000000000000

Приклад 3: Велике число 1 000 000

1. Переведення: $1\ 000\ 000_{10} \approx 11110100001001000000_2$ (20 біт).
2. Нормалізація: Зсуваємо кому вліво на 19 позицій: $1.11101000010010000000_2 \times 2^{19}$
3. Розрахунок полів:
    - Знак: 0.
    - Експонента: Степінь 19. Зміщення: 19 + 127 = 146. $146_{10} = 10010010_2$.
    - Мантиса: Відкидаємо першу одиницю, записуємо решту: 11101000010010000000000
4. Результат: 0 10010010 11101000010010000000000

Знак (1 біт)	Експонента (8 біт)	Мантиса (23 біти)
0	10010010	11101000010010000000000

Приклад 4: Від’ємне число −0.75

1. Переведення модуля числа: $0.75_{10} = 0.11_2$
2. Нормалізація: Зсуваємо кому вправо на 1 позицію: $0.11_2 = 1.1_2 \times 2^{-1}$
3. Розрахунок полів:

- Знак: Число від’ємне → 1.
- Експонента: Степінь -1. Зміщення: -1 + 127 = 126. $126_{10} = 01111110_2$.
- Мантиса: Частина після коми (1). 10000000000000000000000

4. Результат: 1 01111110 10000000000000000000000

Знак (1 біт)	Експонента (8 біт)	Мантиса (23 біти)
0	01111100	01000000000000000000000

Приклад 5: Від’ємне велике число −500 000

1. Переведення: $500\,000_{10} = 1111010000100100000_2$ (19 біт).
2. Нормалізація: Зсуваємо кому вліво на 18 позицій: $1.11101000010010000_2 \times 2^{18}$
3. Розрахунок полів:
   - Знак: Число від’ємне → 1.
   - Експонента: Степінь 18. Зміщення: 18 + 127 = 145. $145_{10} = 10010001_2$.
   - Мантиса: Записуємо дробову частину: 11101000010010000000000 (доповнюємо нулями).
4. Результат: 1 10010001 11101000010010000000000

Знак (1 біт)	Експонента (8 біт)	Мантиса (23 біти)
1	10010001	11101000010010000000000

Додавання та віднімання чисел з плаваючою комою (IEEE 754)

Операції додавання та віднімання для чисел із плаваючою комою значно складніші, ніж для цілих чисел. Головна проблема полягає в тому, що числа можуть мати **різні порядки** (експоненти).

Аналогія з фізикою: Ми не можемо додати **5 метрів** (5×10^0) до **2 кілометрів** (2×10^3) напряму, отримавши «7». Спершу ми повинні привести їх до однієї одиниці виміру (наприклад, перетворити метри в кілометри або навпаки).

У форматі IEEE 754 роль «одиниці виміру» відіграє експонента. Перед виконанням арифметичної дії над мантисами необхідно **вирівняти експоненти**.

Алгоритм виконання операцій

Процес складається з п'яти послідовних кроків:

1. **Розпакування (Виділення полів):**
    - Зчитуємо знаки, експоненти та мантиси обох чисел.
    - Відновлюємо приховану одиницю перед мантисою (дописуємо «1.» на початку), оскільки в пам'яті зберігається лише дробова частина.
2. **Вирівнювання експонент (Масштабування):**
    - Порівнюємо експоненти двох чисел.
    - Якщо вони різні, знаходимо різницю: $\Delta = Exp\_max - Exp\_min$.
    - Мантису числа з меншою експонентою зсуваємо вправо на Δ бітів. При цьому експонента цього числа збільшується до рівня більшої.
    - *Чому зсуваємо менше число?* Це дозволяє зберегти значущі цифри більшого числа. Втрата молодших бітів меншого числа (при зсуві вправо) менш критична для загальної точності.
3. **Арифметична дія над мантисами:**
    - Якщо знаки чисел однакові → додаємо мантиси.
    - Якщо знаки різні → віднімаємо меншу мантису від більшої (враховуючи приховану одиницю).
    - Знак результату визначається знаком числа, більшого за модулем.
4. **Нормалізація результату:** Отриманий результат може порушувати стандартний вигляд 1.xxxxx:
    - Переповнення (результат ≥ 2): Зсуваємо мантису вправо на 1 біт, до експоненти додаємо 1.
    - Втрата нормалізації (результат < 1): Зсуваємо мантису вліво, поки перша цифра не стане одиницею. На кількість зсувів зменшуємо експоненту.
5. **Формування фінального запису:**
    - Відкидаємо «1.» перед комою (приховану одиницю).
    - Округлюємо мантису до 23 бітів.
    - Записуємо результат у поля IEEE 754.

Приклад 1: Додавання чисел

Завдання: Знайти суму двох чисел: 2.5 + 0.5.

Крок 1: Представлення у двійковій формі та виділення полів

- Число A (2.5): $10.1_2 = 1.01_2 \times 2^1$.
    - Експонента: 1.
    - Мантиса (з прихованою одиницею): 1.0100...
- Число B (0.5): $0.1_2 = 1.0_2 \times 2^{-1}$.
    - Експонента: -1.
    - Мантиса (з прихованою одиницею): 1.0000…

Крок 2: Вирівнювання експонент

- Різниця порядків: $1 - (-1) = 2$.
- Число A має більшу експоненту (1), тому число B (0.5) треба привести до порядку 2^1.
- Зсуваємо мантису числа B вправо на 2 позиції: $1.0 \times 2^{-1} \rightarrow 0.1 \times 2^0 \rightarrow 0.01 \times 2^1$.

Крок 3: Додавання мантис Тепер експоненти однакові (2^1), додаємо мантиси: 1.0100 (від 2.5) + 0.0100 (від 0.5, зсунуте) = 1.1000

Крок 4: Нормалізація Результат 1.1000 вже знаходиться в межах [1, 2), тому додаткова нормалізація не потрібна. Експонента залишається 2^1.

Крок 5: Формування IEEE 754 (float32)

- Знак: 0 (плюс).
- Експонента: 1 + 127 = 128 → 10000000.
- Мантиса: Відкидаємо першу одиницю (1.1000 → 1000...).

Результат (структурний запис):

Знак (1)	Експонента (8)	Мантиса (23)
0	10000000	10000000000000000000000

Приклад 2: Віднімання чисел

Завдання: Знайти різницю: 3.5 − 1.0.

Крок 1: Представлення та поля

- Число A (3.5): $11.1_2 = 1.11_2 \times 2^1$. (Exp = 1).
- Число B (1.0): $1.0_2 = 1.00_2 \times 2^0$. (Exp = 0).

Крок 2: Вирівнювання експонент

- Різниця порядків: 1 − 0 = 1.
- Зсуваємо мантису меншого числа (B) вправо на 1 позицію, щоб привести до порядку 2^1: $1.00 \times 2^0 \rightarrow 0.10 \times 2^1$.

Крок 3: Віднімання мантис Віднімаємо вирівняну мантису B від мантиси A: 1.11 (від 3.5) − 0.10 (від 1.0, зсунуте) = 1.01

Крок 4: Нормалізація Результат 1.01 є нормалізованим (перша цифра 1).

Експонента результату дорівнює спільній експоненті: 1.

Крок 5: Формування IEEE 754 (float32)

- Знак: 0 (результат додатний).
- Експонента: 1 + 127 = 128 → 10000000.
- Мантиса: Відкидаємо приховану одиницю (1.01 → 01...).

Результат (структурний запис):

Знак (1)	Експонента (8)	Мантиса (23)
0	10000000	01000000000000000000000

Завдання для виконання

Завдання №1: Переведення дробових чисел з десяткової системи у двійкову.

Варіант	Дріб A	Дріб B
1	0.5	0.2
2	0.25	0.3
3	0.75	0.1
4	0.125	0.6
5	0.375	0.7
6	0.875	0.9
7	0.625	0.35
8	0.0625	0.15
9	0.3125	0.45
10	0.9375	0.55

Завдання №2: робота з числами з фіксованою комою

Варіант	Завдання 1 (перевести з A_{10} → Q5.3)	Завдання 2 (Q5.3 → A_{10})	Завдання 3 (сума у Q5.3)
1	12.375 → Q5.3	10101.011 → ?	7.25 + 3.5
2	9.5 → Q5.3	01011.101 → ?	5.125 + 6.75
3	15.125 → Q5.3	01110.010 → ?	8.25 + 2.625
4	7.875 → Q5.3	00111.111 → ?	9.5 + 4.125
5	10.25 → Q5.3	01010.100 → ?	6.375 + 5.5
6	13.75 → Q5.3	01101.011 → ?	11.125 + 3.875
7	5.625 → Q5.3	00101.101 → ?	7.75 + 6.25
8	8.5 → Q5.3	01000.100 → ?	12.375 + 2.75
9	11.875 → Q5.3	01011.111 → ?	9.25 + 7.125
10	14.5 → Q5.3	01110.100 → ?	8.875 + 5.5
11	6.75 → Q5.3	00110.110 → ?	10.25 + 6.125

12	9.875 → Q5.3	01001.111 → ?	11.5 + 4.375
13	12.625 → Q5.3	01100.101 → ?	7.375 + 5.75
14	4.25 → Q5.3	00100.010 → ?	6.875 + 9.125
15	13.125 → Q5.3	01101.001 → ?	8.25 + 10.75
16	7.5 → Q5.3	00111.100 → ?	12.5 + 3.625
17	10.75 → Q5.3	01010.110 → ?	11.875 + 2.25
18	6.125 → Q5.3	00110.001 → ?	9.5 + 4.75
19	14.875 → Q5.3	01110.111 → ?	7.625 + 5.125
20	11.25 → Q5.3	01011.010 → ?	10.75 + 6.375

Завдання №3: Робота з числами з плаваючою комою (формат IEEE 754)

Варіант	Завдання А: Перевести числа у формат IEEE 754	Завдання Б: Виконати арифметичні дії (у двійковому коді)
1	0.375; –12.5; 1 000 000; –0.1; 3.14	(+) 0.5 + 0.25 (−) 5.75 – 2.5
2	0.5; –7.25; 10 000; –0.25; 2.71	(+) 1.5 + 2.25 (−) 7.5 – 3.25
3	0.125; –15.5; 500 000; –0.05; 6.28	(+) 2.5 + 1.25 (−) 10.75 – 4.5
4	0.625; –3.5; 250 000; –0.125; 1.41	(+) 3.5 + 0.75 (−) 12.5 – 6.25
5	0.875; –25.75; 2 000 000; –0.03125; 9.81	(+) 4.25 + 1.75 (−) 15.5 – 7.75

Лабораторна робота №5

Операції множення та ділення над двійковими числами

Теоретичні відомості

Арифметика двійкового множення є значно простішою за десяткову, оскільки двійкова цифра може набувати лише двох значень: **0** або **1**. Це означає, що процес зводиться не до заучування таблиці множення, а до серії операцій **зсуву** та **додавання**.

Існує два основні підходи до реалізації множення, які дають однаковий результат, але відрізняються послідовністю дій.

Спосіб 1. Множення починаючи з молодших розрядів (LSB)

Це класичний метод, який ми використовуємо в «стовпчик» у десятковій системі. Ми беремо останню цифру множника, множимо на неї множене, записуємо результат. Потім беремо передостанню, множимо, зсуваємо результат вліво на одну позицію і так далі.

Приклад: 1001 × 1010

1. Множник закінчується на 0 → записуємо 0000.
2. Наступна цифра 1 → записуємо саме число (1001) зі зсувом вліво на 1 розряд.
3. Наступна цифра 0 → записуємо 0000 зі зсувом на 2 розряди.
4. Старша цифра 1 → записуємо 1001 зі зсувом на 3 розряди.
5. Сумуємо: Результат 1011010.

Спосіб 2. Множення починаючи зі старших розрядів (MSB)

У цьому випадку процес починається з крайньої лівої цифри множника. Кожен наступний частинний добуток зсувається вправо відносно попереднього. Цей метод часто використовується в апаратурі, оскільки дозволяє почати обробку даних ще до отримання всіх розрядів множника.

Алгоритмічна сутність множення

Незалежно від обраного способу, операція множення складається з циклічного повторення двох дій:

1. Аналіз розряду множника.
2. Додавання та зсув (Shift and Add).

Логіка процесу:

- Якщо поточний розряд множника дорівнює 1: до суми частинних добутків додається множене, після чого виконується зсув.
- Якщо поточний розряд множника дорівнює 0: додавання не виконується (адже множення на нуль дає нуль). Процесор просто виконує зсув (вліво або вправо залежно від алгоритму), переходячи до наступного розряду.

Економія операцій:

Якщо в множнику зустрічається серія нулів (наприклад, число 1001), алгоритм не виконує додавання для проміжних нулів. Він лише фіксує кількість необхідних зсувів.

Приклад: Якщо після одиниці йдуть два нулі, а потім знову одиниця, то множене буде зсунуто одразу на 3 позиції перед наступним додаванням.

Ділення двійкових чисел

Операція ділення в комп’ютерах зазвичай реалізується через алгоритм, аналогічний шкільному діленню «в стовпчик». Суть методу полягає у послідовному відніманні дільника від діленого (або його частини) для отримання цифр частки.

Загальний алгоритм («Шкільний метод»):

1. Виділяємо частину діленого (починаючи зі старших розрядів), яка є більшою або рівною дільнику.
2. Віднімаємо дільник.
    - Якщо віднімання можливе (результат ≥ 0) → у частку записуємо **1**.

- Якщо дільник більший за поточну частину діленого → у частку записуємо **0**.

3. До залишку зносимо наступний біт діленого і повторюємо процес.

Приклад: Поділимо 11001100 (204) на 1100 (12).

1. Беремо перші 4 біти: 1100. Вони дорівнюють дільнику 1100.
   - 1100 − 1100 = 0.
   - Записуємо у частку 1.
2. Зносимо наступний біт (1) → маємо 1. Це менше за 1100.
   - Записуємо у частку 0.
3. Зносимо наступний біт (1) → маємо 11. Це менше за 1100.
   - Записуємо у частку 0.
4. Зносимо 0 → маємо 110. Це менше за 1100.
   - Записуємо у частку 0.
5. Зносимо останній 0 → маємо 1100. Дорівнює дільнику.
   - 1100 − 1100 = 0.
   - Записуємо у частку 1.

     Результат: 10001 (17 у десятковій системі).

Машинна реалізація: Ділення з відновленням залишку

Для комп'ютера визначити «на око», чи вміщується дільник у ділене, складно. Тому АЛП (арифметико-логічний пристрій) використовує формальний метод - ділення з відновленням залишку.

Чому потрібне відновлення?

Комп'ютер спочатку «наосліп» віднімає дільник від поточного залишку.

- Якщо результат виявився додатним (або нулем) - все добре, у частку пишемо 1, а залишок зсуваємо для наступного кроку.
- Якщо результат виявився від'ємним - це означає, що ми відняли забагато (дільник не вмістився). У частку пишемо 0, але тепер нам потрібно скасувати помилкову дію. Для цього ми додаємо дільник назад до отриманого від'ємного числа. Це і називається відновленням залишку.

Алгоритм по кроках:

1. Зсув залишку вліво на 1 розряд.
2. Спроба віднімання дільника.
3. Аналіз знака результату:
    - (+) Плюс: записуємо 1 у молодший розряд частки. Переходимо до кроку 1.
    - (–) Мінус: записуємо 0 у частку. Виконуємо додавання дільника до поточного результату (відновлюємо попередній стан). Переходимо до кроку 1.

Цей процес повторюється стільки разів, скільки розрядів має число.

Завдання для виконання

Завдання №1: Згідно з вказаним варіантом завдання здійснити операцію множення десяткових чисел A_{10} та B_{10} у двійковій системі числення.

Варіант	1	2	3	4	5	6	7	8	9	10
A_{10} (Множене)	58	61	65	81	55	91	173	86	61	37
B_{10} (Множник)	34	56	48	79	42	59	45	54	44	53
Варіант	11	12	13	14	15	16	17	18	19	20
B_{10} (Множник)	61	77	83	97	62	32	99	57	64	78
B_{10} (Множник)	29	78	24	58	42	65	66	71	87	93

Завдання №2: Згідно з вказаним варіантом завдання здійснити операцію ділення десяткових чисел A_{10} та B_{10} у двійковій системі числення.

Варіант	1	2	3	4	5	6	7	8	9	10
A_{10} (Множене)	578	616	672	790	546	885	720	864	616	371
B_{10} (Множник)	34	56	48	79	42	59	45	54	44	53
Варіант	11	12	13	14	15	16	17	18	19	20
B_{10} (Множник)	609	780	816	928	630	325	990	568	609	744
B_{10} (Множник)	29	78	24	58	42	65	66	71	87	93

Лабораторна робота №6

Операції інверсії, кон’юнкції та диз’юнкції над двійковими числами

Теоретичні відомості

Алгебра логіки (двійкова логіка) – розділ математичної логіки, що вивчає систему логічних операцій над висловлюваннями. В алгебрі логіки значенням змінних є значення істинності істина або хиба, які зазвичай визначаються як 1 і 0 відповідно.

На відміну від елементарної алгебри, у якій значеннями змінних є числа, а основними операціями є додавання і множення, основними операціями булевої алгебри є кон'юнкція (операція І, AND, позначається як $\wedge$), диз'юнкція (АБО, OR, позначається як $\vee$), і заперечення або інверсії (НІ, NOT, позначається як ¬).

Базовими елементами, якими оперує алгебра логіки, є висловлювання.

Як уже зазначалося, базовими операціями булевої алгебри (алгебри логіки) є такі логічні операції:

- І (кон'юнкція), позначається $x \wedge y$, задовольняє $x \wedge y = 1$, якщо $x = y = 1$ та $x \wedge y = 0$ в інших випадках.
- АБО (диз'юнкція), позначається $x \vee y$, задовольняє $x \vee y = 0$, якщо $x = y = 0$ та $x \vee y = 1$ в інших випадках.
- НІ (заперечення), позначається $\neg x$ (!x), задовольняє $\neg x = 0$, якщо $x = 1$ та $\neg x = 1$, якщо $x = 0$.

Альтернативним способом значення $x \wedge y$, $x \vee y$, та $\neg x$ можна представити в табличному вигляді для всіх можливих значень за допомогою таблиць істинності таким способом:

x	y	$x \wedge y$	$x \vee y$
0	**0**	0	0
1	**0**	0	1
0	**1**	0	1
1	**1**	1	1

x	$\neg x$
0	1
1	0

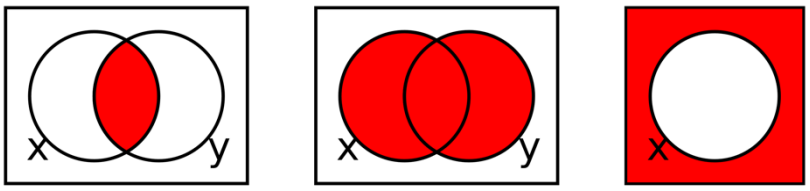
Якщо значення істинності 0 та 1 інтерпретувати як цілі числа, ці операції можна було задати як звичайні операції арифметики (де x + y використовує додавання, а xy використовує множення), або як функції мінімуму/максимуму:

$$x \wedge y = xy = \min(x, y)$$
$$x \vee y = x + y - xy = \max(x, y)$$
$$\neg x = 1 - x$$

Можна вважати, що лише заперечення і одна з двох операцій, що залишилися, є базовими, оскільки наступні рівняння дозволяють визначити кон'юнкцію через операції заперечення та диз'юнкцію, і навпаки:

$$x \wedge y = \neg(\neg x \vee \neg y)$$
$$x \vee y = \neg(\neg x \wedge \neg y)$$

Діаграми Венна на малюнку знизу показують операції кон'юнкції x∧y, диз'юнкції x∨y, та доповнення ¬x:



x∧y x∨y ¬x

Три булеві операції описані вище називають базовими або первинними, що означає, що вони можуть бути базисом для всіх інших булевих операцій, оскільки всі інші операції можна виразити через них за допомогою композиції. Серед операцій, які можна побудувати з базових операцій є такі:

$$x \to y = \neg x \vee y$$
$$x \oplus y = (x \vee y) \wedge \neg(x \wedge y) = (x \wedge \neg y) \vee (\neg x \wedge y)$$
$$x \equiv y = \neg(x \oplus y) = (x \wedge y) \vee (\neg x \wedge \neg y)$$

Ці визначення дають такі таблиці істинності, у яких показані значення цих операцій для всіх можливих вхідних значень:

x	y	$x \to y$	$x \oplus y$	$x \equiv y$
0	**0**	1	0	1
1	**0**	0	1	0
0	**1**	1	1	0
1	**1**	1	0	1

Перша операція, x → y, називається імплікацією. Якщо x є істинним, тоді за значення виразу x → y приймається значення y. Але якщо x приймає хибне значення, то значення y можна було б ігнорувати; однак ця операція повинна повернути деяке логічне значення, і існує лише два варіанти вибору. То ж за визначенням, x → y є істиною коли x є хибним.

Друга операція, x ⊕ y, називається виключною диз'юнкцією (часто задається як абревіатура XOR), аби відрізнити її від диз'юнкції. Вона є істиною, лише коли x та y різні.

Третя операція, обернена до виключної диз'юнкції, називається еквівалентність або булева рівність: x ≡ y буде істиною, лише коли x та y мають однакове значення.

Для двох даних операндів, кожен з яких має $2^2 = 4$ можливих комбінації входів. Оскільки кожен вихід може мати два можливих значення, існує загалом $2^4 = 16$ можливих булевих операцій.

Діаграма Венна – це графічне представлення булевих операцій за допомогою зафарбованих областей, що перекриваються. По одній області для кожної змінної, всі області круглі, як показано на прикладах. Внутрішня й зовнішня частина області x відповідає значенням 1 (істина) та 0 (хиба) відповідно для змінної x. Зафарбована область показує значення операції для кожної комбінації цих областей, де зафарбована область означає 1, а не зафарбована – 0 (але в деяких книжках може зустрітися і навпаки).

Завдання для виконання роботи

Завдання №1: Скласти таблицю істинності для функції (A ∧ B ⊕ B ∧ C) ∨ (¬C → A)

Лабораторне заняття №7

Практичне знайомство з Tinkercad: робота з компонентами та симуляція

Теоретичні відомості

Tinkercad - це безкоштовне онлайн-середовище для створення 3D-моделей, електронних схем і програмування мікроконтролерів. Розроблений компанією **Autodesk**, він поєднує у собі три основні модулі:

1. **Tinkercad 3D Design** – моделювання простих тривимірних об’єктів;
2. **Tinkercad Circuits** – симулятор електронних схем;
3. **Tinkercad Codeblocks** – візуальне програмування для створення алгоритмів.

Tinkercad призначений для навчання основам електроніки, програмування та цифрового мислення. Він є популярним у школах і закладах професійної освіти, оскільки не потребує встановлення програмного забезпечення та дозволяє моделювати все безпосередньо у браузері.

Призначення і можливості середовища Tinkercad

Основні можливості:

- **Моделювання електронних схем** із реалістичними компонентами: резисторами, діодами, транзисторами, світлодіодами, кнопками, перемикачами, логічними елементами, сенсорами, двигунами тощо.
- **Симуляція роботи схем** - спостереження, як змінюються сигнали, напруга, струм при натисканні кнопок або зміні параметрів.
- **Програмування мікроконтролерів Arduino** у двох режимах:
    - блокове програмування (візуальне, для початківців);
    - текстове програмування на мові **C/C++**.
- **Вимірювання параметрів** через віртуальні прилади (вольтметр, осцилограф, мультиметр).
- **3D-моделювання корпусів та пристроїв**, у яких можна розмістити створену електроніку.

Огляд **Tinkercad Circuits** - це онлайн-середовище для **створення, тестування й симуляції електронних схем**.

Його головне призначення - навчити користувачів розуміти, **як працює електроніка та цифрова логіка** без потреби у фізичних компонентах.

Основні можливості:

1. **Моделювання електричних схем**
    - Створення з'єднань між джерелами живлення, резисторами, світлодіодами, кнопками, транзисторами, логічними елементами тощо.
    - Робота як з **аналоговими**, так і з **цифровими сигналами**.
2. **Симуляція роботи схем**
    - Перевірка працездатності схеми без фізичної збірки.
    - Відображення рівнів напруги, напрямку струму, станів 0 (LOW) і 1 (HIGH).
3. **Програмування мікроконтролерів Arduino**
    - Блокове програмування (візуальне середовище).
    - Текстове програмування на **C/C++**.
    - Можливість комбінувати електроніку та програму в одній симуляції.
4. **Вимірювання параметрів у режимі симуляції**
    - Використання цифрових приладів (вольтметр, мультиметр, осцилограф).
5. **3D-інтеграція з іншими модулями Tinkercad**
    - Можна створити 3D-корпус пристрою і вставити у нього власну електронну плату.

Використання Tinkercad Circuits у навчанні

Tinkercad - це базовий інструмент для розвитку **компетентностей з кодування, архітектури комп'ютера та цифрової грамотності**.

У навчальному процесі він використовується для:

- моделювання схем, які пояснюють принцип **сигналів 0 і 1**;

- демонстрації логічних операцій **(AND, OR, NOT, XOR)**;
- побудови **простих цифрових пристроїв** - лічильників, таймерів, індикаторів, регуляторів яскравості;
- симуляції **роботи Arduino** (наприклад, керування світлодіодом через код);
- підготовки студентів до **реального конструювання** електронних систем.

Перевага Tinkercad - у тому, що він

- не потребує складного обладнання;
- працює у будь-якому браузері;
- дозволяє без ризику помилок «погратися» з електронікою, щоб зрозуміти, як вона мислить.

Використання в житті - практичні реальні кейси

1. Освітні проєкти

- **STEM-лабораторії у школах:** учні створюють макети світлофорів, сигналізацій, систем поливу чи охорони.
- **Курси з робототехніки:** студенти вчаться керувати двигунами, сервоприводами, сенсорами.
- **Професійна освіта:** імітація архітектури комп'ютера через прості логічні схеми.

2. Прототипування

- Викладач або розробник може **спроектувати плату**, перевірити її логіку та функції перед реальним виготовленням.
- Малі стартапи використовують Tinkercad для **тестування ідей IoT-пристроїв** (наприклад, датчик температури, освітлення, руху).

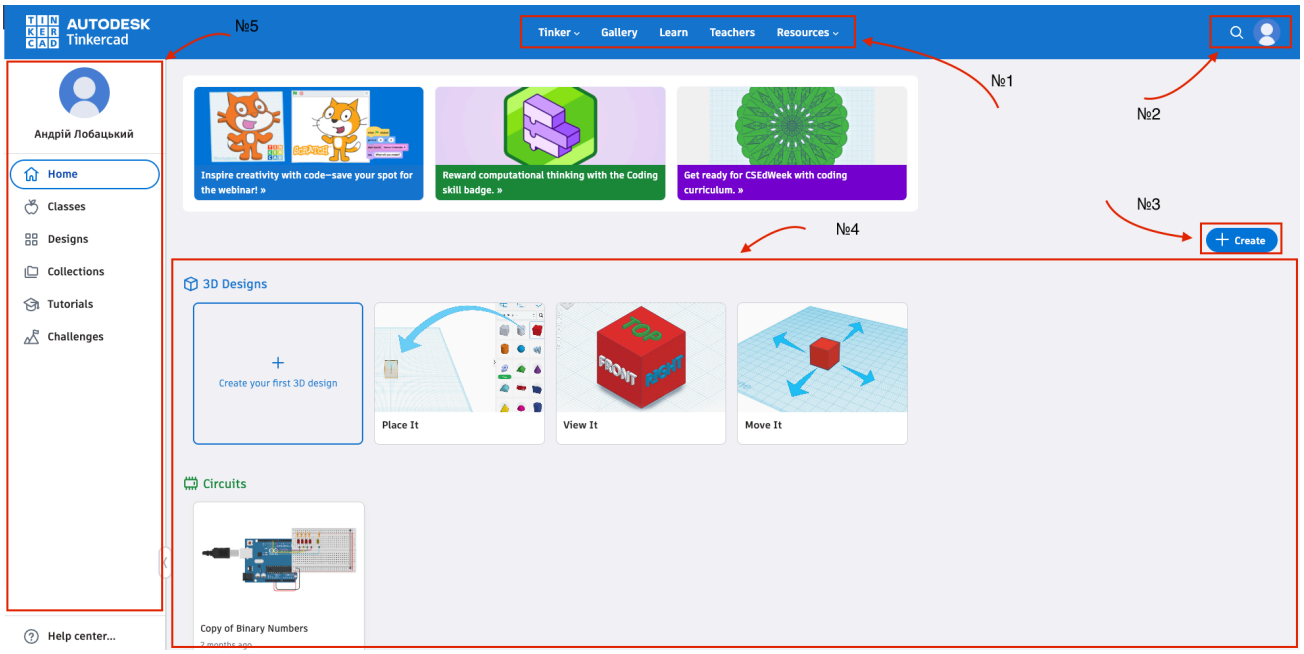
3. Хобі та творчість

- Моделювання **розумного дому** - керування світлом або вентиляцією через Arduino.
- Створення **інтерактивних іграшок** із датчиками руху або світла.
- 3D-друк корпусів для саморобних пристроїв.

4. Початковий етап професійного проектування

- Для майбутніх інженерів і педагогів Tinkercad - це перший крок до професійних систем на кшталт:
    - **Fritzing, Proteus, KiCad, Multisim**,
    - або CAD-платформ (Autodesk Fusion 360, Altium Designer).

3. Огляд основного інтерфейсу Tinkercad



№1 - Головна панель навігації

Розташована у верхній частині сторінки.

Містить основні розділи сайту:

- **Tinker** - перехід до роботи з 3D-дизайнами, схемами чи кодом.
- **Gallery** - перегляд публічних проєктів користувачів.
- **Learn** - навчальні матеріали, уроки, приклади.
- **Teachers** - розділ для педагогів із методичними ресурсами.
- **Resources** - додаткові ресурси, довідка, документація, матеріали для навчання.

№2 - Кнопки пошуку та профілю

У правому верхньому куті:

- **Пошук** - дозволяє шукати проєкти, схеми, користувачів.
- **Профіль** - доступ до налаштувань акаунта, виходу, персональних даних.

№3 - Кнопка «Create»

Синя кнопка **“+ Create”** використовується для створення нового проєкту:

- **3D Design** - 3D-моделі;
- **Circuits** - електронні схеми;
- **Codeblocks** - програмування за допомогою блоків;
- **Lesson** - створення навчального матеріалу (для викладачів).

№4 - Робоча область

Основна частина сторінки, де відображаються:

- Створені користувачем 3D-дизайни або схеми;
- Банери з новинами, вебінарами, порадами;
- Приклади або шаблони для початку роботи.

№5 - Бокова панель користувача

Зліва розташоване меню з основними розділами:

- **Home** - головна сторінка;
- **Classes** - класи та курси (для викладачів);
- **Designs** - усі створені проєкти (3D, Circuits, Codeblocks);
- **Collections** - збережені або згруповані проєкти;
- **Tutorials** - інтерактивні навчальні уроки;
- **Challenges** - завдання для розвитку навичок проєктування.

У нижній частині є кнопка **Help center** - довідка та підтримка користувача.

4. Огляд інтерфейсу середовища Tinkercad Circuits

№1 - Панель інструментів

Розташована у верхній частині вікна.

Містить основні кнопки для роботи з елементами схеми:

- **Створити / Зберегти / Копіювати / Видалити** - керування файлами та об’єктами;
- **Undo / Redo** - скасування та повтор дії;
- **Коментарі** - додавання приміток до проєкту;
- **Кольори та товщина ліній** - зміна вигляду з’єднань;
- **Інструменти вирівнювання** - точне розташування елементів;
- **Інші функції** - прив’язка, масштабування, режим редагування.

Це основна панель керування під час побудови електричної схеми.

№2 - Назва проєкту та статус збереження

У верхній частині відображається:

- **Назва схеми** (наприклад, *Epic Trug*) - її можна змінити, натиснувши на неї;
- **“All changes saved”** - автоматичне збереження змін у хмарі.

Tinkercad автоматично зберігає проєкт після кожної дії, тож дані не губляться.

№3 - Панель симуляції

Містить три основні кнопки:

- **Code** - відкриває вкладку для програмування (наприклад, Arduino-код або блокове кодування).

- **Start Simulation / Stop Simulation** - запуск або зупинка моделювання схеми в реальному часі.
- **Send To** - експорт проєкту, публікація або передача іншим користувачам.

Ця панель використовується для тестування створеної електричної схеми.

№4 - Панель компонентів

Розташована праворуч.

Тут містяться **електронні компоненти**, які можна перетягнути на робочу область.

Складові панелі:

- **Search** - пошук потрібного елемента (наприклад, “LED”, “Resistor”, “Arduino”).
- **Components** - випадаючий список категорій:
    - *Basic* - базові елементи (резистори, LED, кнопки, батареї);
    - *Power* - джерела живлення;
    - *Input/Output* - сенсори, двигуни, серводвигуни;
    - *Microcontrollers* - Arduino, micro:bit тощо.

Перетягнувши елемент у центральну зону, ви додаєте його до своєї схеми.

№5 - Робоче поле (Workspace)

Центральна частина інтерфейсу, де створюється сама схема.

Тут користувач може:

- розташовувати компоненти (LED, резистори, батареї, плати);
- з’єднувати їх дротами;
- переміщувати, обертати, видаляти елементи;
- переглядати результат у симуляції.

Порада: використовуйте масштабування коліщатком миші та “руку” для зручного огляду схеми.

Завдання для виконання

Завдання №1: Ознайомлення з середовищем Tinkercad Circuits

1. Увійдіть у середовище Tinkercad Circuits.
2. Ознайомтеся з ключовими елементами інтерфейсу:

- панель компонентів;
- робоче поле;
- інструменти з’єднання (проводи);
- панель/кнопки запуску симуляції.

3. Зробіть скріншот інтерфейсу та додайте його до звіту.
4. На скріншоті (або під ним) коротко позначте й підпишіть основні елементи інтерфейсу (наприклад: «панель компонентів», «робоча зона», «інструменти», «симуляція»).

Лабораторна робота №8

Робота з перемикачами: подання 0 і 1 у схемах

Теоретичні відомості

Світлодіод – це напівпровідниковий прилад, що випромінює світло під час проходження через нього електричного струму. Світлодіоди мають два виводи: анод (+) і катод (–). Струм може проходити тільки від анода до катода, тому для підключення **анод світлодіода слід з'єднувати з позитивним полюсом живлення, а катод – з негативним**. (рис 1.)

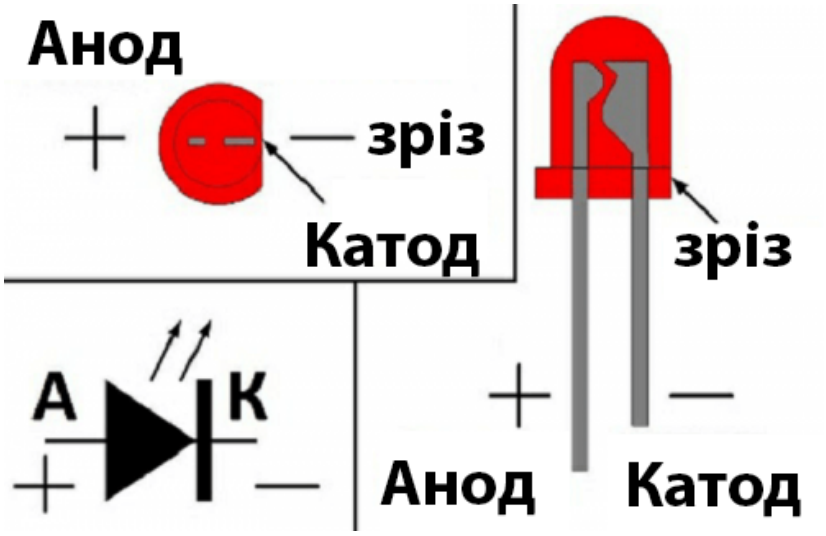


Рис. 1 - Зображення світлодіод

Якщо підключити LED неправильно або безпосередньо до джерела, він не світитиметься або може згоріти. Для нормальної роботи світлодіода необхідно обмежувати струм через нього за допомогою резистора – інакше пряме підключення до батареї може спалити світлодіод від надмірного струму. Світлодіоди випускаються різних кольорів і розмірів, часто використовуються як **індикатори стану**: світіння LED може позначати логічну "1" або сигнал увімкнення, а відсутність світіння – "0" або вимкнення.

Світлодіод (LED) у середовищі Tinkercad Circuits

На зображені нижче подано інтерфейс Tinkercad Circuits

з компонентом LED, який використовується для побудови електричних схем у *Tinkercad Circuits*. (рис. 2)

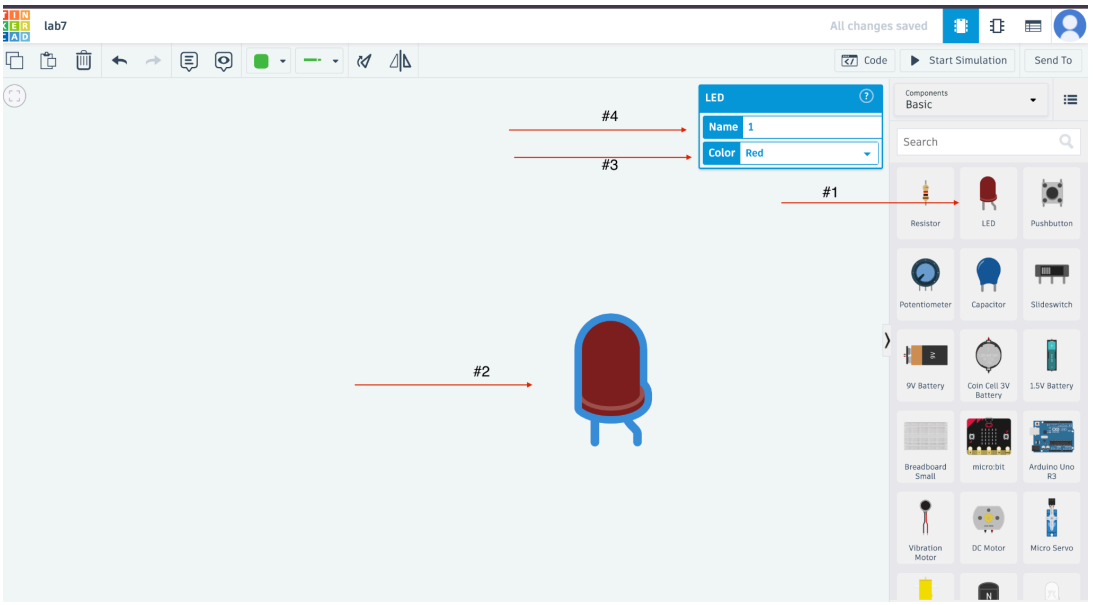


Рис. 2 - Інтерфейс компонента LED.

Розглянемо елементи позначені на ілюстраціях:

#1 Відображення компонента LED в бібліотеці компонентів *Tinkercad*. Після додавання на робоче поле ви побачите LED з двома ніжками.

#2 - Будова LED

Анод (Anode) - під'єднується до **позитивного полюсу (+)** джерела живлення або через перемикач.

Катод (Cathode) - під'єднується до **негативного полюсу (–)**, або **GND** (землі).

Якщо підключити навпаки - світлодіод не світитиметься, бо струм не зможе пройти через діод у зворотному напрямку.

У Tinkercad при наведенні курсора з'являється підказка, яка допомагає відрізнити контакти. (рис 3)

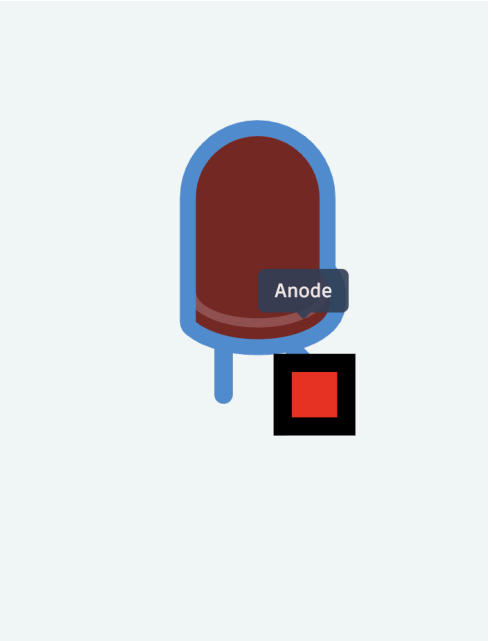


Рис. 3 - Вигляд підказки при наведенні курсора

#3 - Параметр вибору кольору світіння LED. Доступні варіанти: *Red, Green, Blue, Yellow, тощо*. Колір не впливає на схему, але допомагає візуально розрізняти елементи (наприклад, різні сигнали).

#4 - Поле Name - це ім'я компонента на схемі. За замовчуванням - LED1, LED2 і т. д. Його можна змінити для зручності (наприклад, LED_Red або Indicator), щоб не плутати між собою кілька світлодіодів.

Принцип роботи у Tinkercad.

У *Tinkercad Circuits* LED моделює реальну поведінку діода:

- Світиться лише при прямому струмі (анод → катод).
- Не світиться при зворотному.
- Якщо подати надто великий струм без резистора - у симуляції LED «перегорає» (з'являється помилка).

Приклад підключення:

- +9V → Switch → LED → Resistor → GND
- +9V → Switch → Resistor → LED → GND

У Tinkercad ці два варіанти еквівалентні. Головне - послідовно з'єднати LED і резистор.

Світлодіод LED RGB

LED RGB (мультиколірний світлодіод) - світлодіод із трьома кольоровими каналами (*Red, Green, Blue*). (рис 4.)

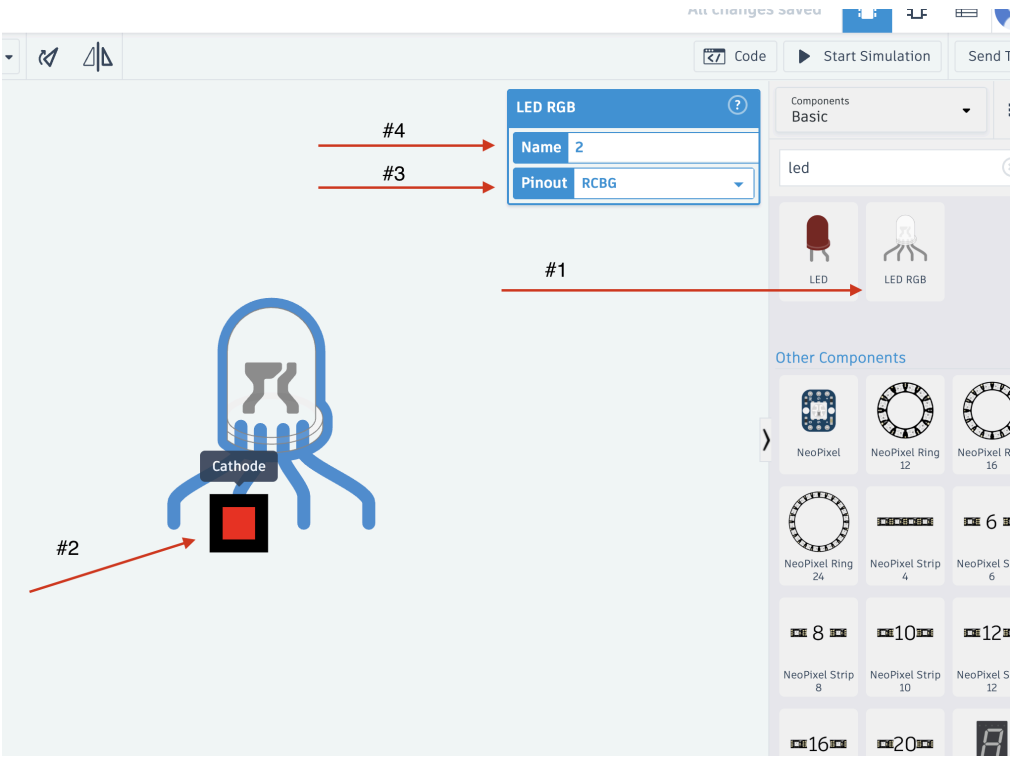


Рис. 4 - LED RGB (мультиколірний світлодіод)

Особливості LED RGB:

- У параметрі Pinout можна обрати схему підключення: RCBG або GRBC.
- Має спільний катод (–).
- RGB LED дозволяє створювати будь-який колір світіння, змішуючи три основних кольори через різні опори.

У середовищі Tinkercad Circuits світлодіод - це зручний навчальний інструмент для демонстрації:

- напрямку струму в електричному колі;
- роботи двійкових станів (світиться = 1, вимкнено = 0);
- необхідності обмежувального резистора.

Перемикачі (Switches)

Перемикач - це електромеханічний компонент, який замкнувши або розімкнувши електричне коло, дозволяє керувати потоком струму. Його головна функція - керування подачею сигналу.

- У замкнутому стані (ON) струм проходить - це відповідає логічній “1”.
- У розімкнутому стані (OFF) коло розірване, струм не тече - логічна “0”.

Такі елементи широко використовуються в цифровій техніці для передачі двійкових сигналів (HIGH/LOW).

Основні типи перемикачів:

1. **SPST** (Single Pole Single Throw) - найпростіший тип із двома клемами. Діє як звичайний вимикач світла: або з’єднує ланцюг, або розриває його. (рис. 5)

Рис. 5 - Однополюсний вимикач (SPST): конструкція та схема комутації.

2. **SPDT** (Single Pole Double Throw) - має три контакти: Common (спільний) і два виходи. Перемикає спільний контакт між двома іншими, створюючи вибір одного з двох шляхів для струму. (рис. 6)

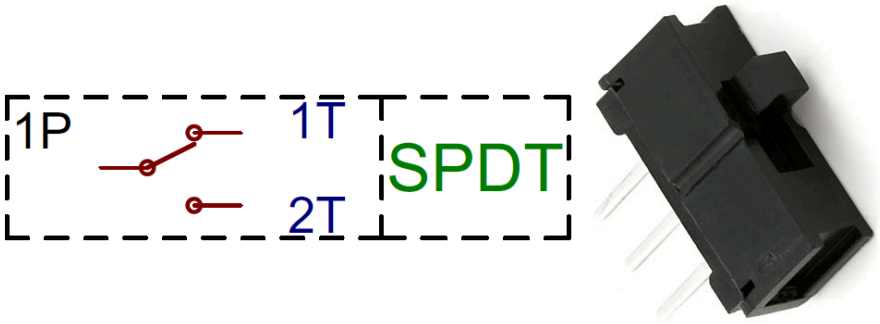


Рис. 6 - Однополюсний перемикач на два напрямки (SPDT): конструкція та схема комутації.

3. **DPDT** (Double Pole Double Throw) - фактично два SPDT, з'єднаних паралельно; дозволяє одночасно перемикати два незалежні кола. (рис. 7)

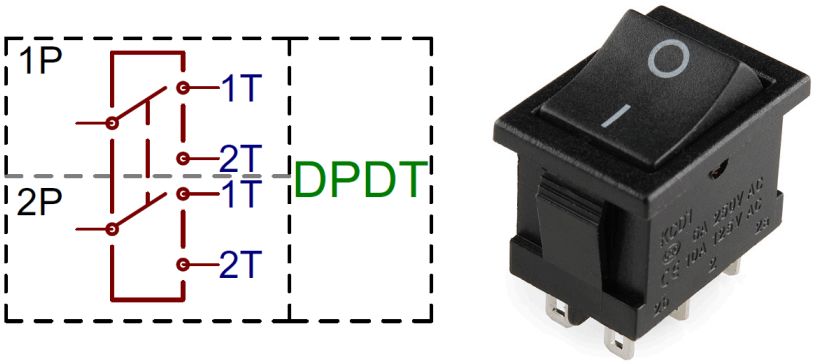


Рис. 7 - Двополюсний перемикач на два напрямки (DPDT): схема комутації двох незалежних кіл.

Перемикачі у Tinkercad Circuits

У середовищі Tinkercad Circuits доступно кілька видів перемикачів (рис. 8) у категорії Basic components → Switches. Найчастіше використовуються:

- **Pushbutton** (Кнопковий перемикач) - активується лише під час натискання. Після відпускання повертається у вихідний стан.
- **Slideswitch** (Повзунковий перемикач) - має два або три контакти, і фіксується в одному з двох положень: ON/OFF.

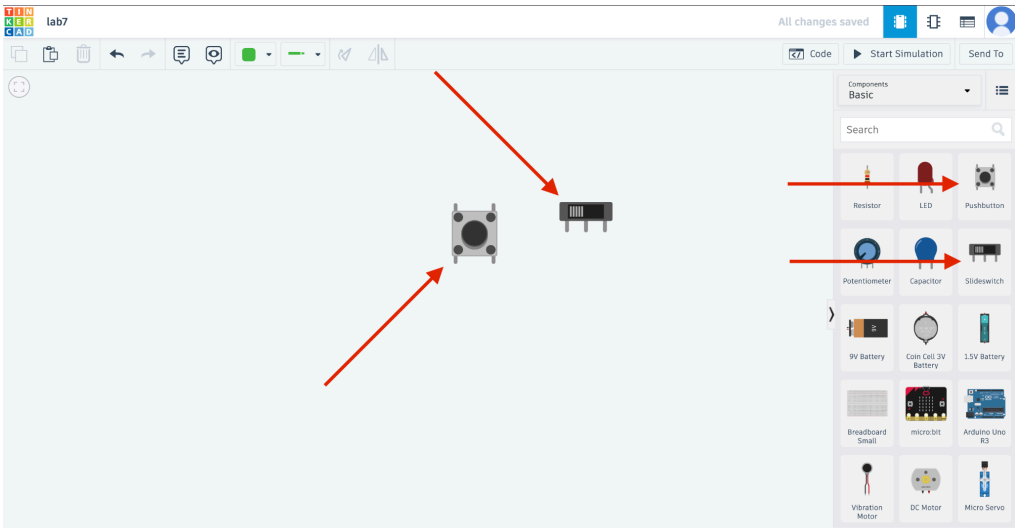


Рис. 8 - Перемикачі Tinkercad Circuits

Робота з **Pushbutton** у Tinkercad

Pushbutton кнопка має чотири термінали - по два з кожного боку:

- №1 Terminal 1a
- №2 Terminal 1b
- №3 Terminal 2a
- №4 Terminal 2b

Пари контактів (1a–1b і 2a–2b) електрично з'єднані попарно. Коли кнопку натиснуто, між цими парами замикається ланцюг.

Тобто кнопка працює як миттєвий вимикач (momentary switch) - струм проходить лише під час натискання.

Правила користування (рис. 9) :

1. Додайте компонент Pushbutton у робочу область (п. №1).
2. При наведенні курсора з'являються підказки з номерами терміналів (п. №2–3).
3. В полі Name (п. №4) задайте ім'я, наприклад PB1 - це полегшить ідентифікацію в схемі.

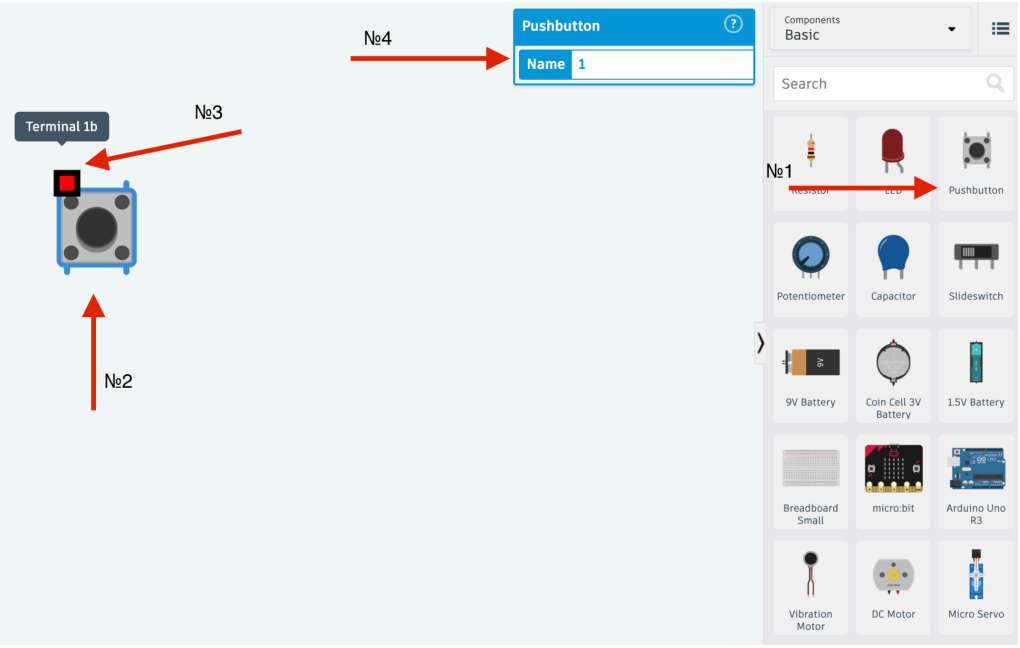


Рис. 9 - Ілюстрація в інтерфейсі Tinkercad

Робота зі **Slideswitch** у Tinkercad. Slideswitch має три термінали:

- Terminal 1 - крайній контакт;
- *Common* - центральний;

- Terminal 3 - інший крайній контакт.

Коли повзунок переміщується, *Common* з'єднується або з Terminal 1, або з Terminal 3. Це імітує простий SPDT-перемикач, який можна використовувати для увімкнення або вибору лінії живлення.

Правила користування (рис. 10):

1. Оберіть Slideswitch (п. №1).
2. Наведіть на контакт, щоб побачити підказку - наприклад, *Terminal 2* (п. №2).
3. У правому полі (п. №3) можна перейменувати компонент, наприклад SW1.
4. У схемі використовуйте лише два з трьох терміналів, якщо потрібно просте вмикання/вимикання (SPST-режим).

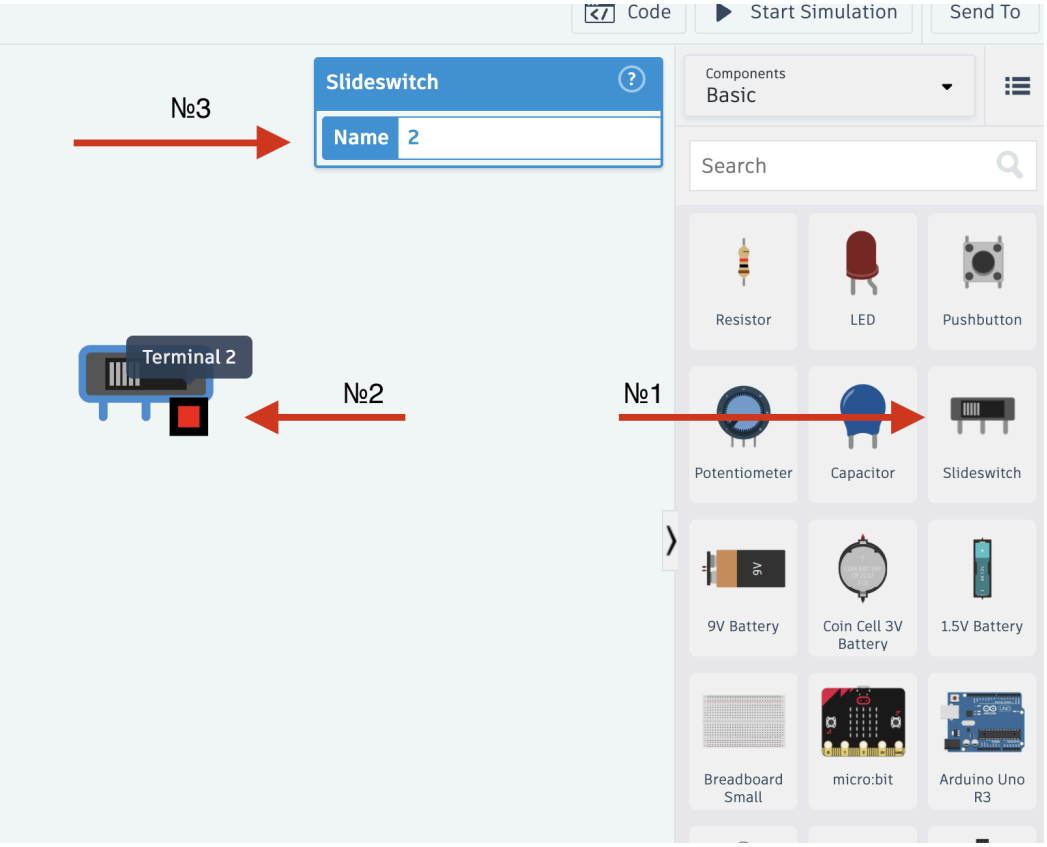


Рис. 10 - Інтерфейс середовища Tinkercad: панель властивостей та комутаційні контакти компонента Slide Switch.

Приклади використання в схемах

Pushbutton + LED:

Коли натискаєш кнопку - ланцюг замикається, LED засвічується. +9V → Pushbutton → Resistor → LED → GND

Відпускаєш кнопку - LED гасне. Це приклад реалізації тимчасового сигналу (імпульсу).

Slideswitch + LED:

Перемикач фіксує положення, і LED залишається світитися, доки слайдер не перемкнути назад. +9V → Slideswitch → LED → Resistor → GND. Це приклад стабільного двійкового стану (0/1).

Поради при роботі в симуляції Tinkercad

- Не забудьте підключити GND (землю) до джерела живлення - інакше коло не замкнеться.
- Для LED завжди додавайте резистор у послідовність.
- У полі Start Simulation можна натискати на кнопку або перемикач, щоб перевірити зміну станів 0/1.
- Використовуйте кольорове маркування дротів (червоний - +V, чорний - GND) для зручності читання схеми.

Резистори

Резистор - це пасивний електронний компонент, який обмежує силу струму та створює падіння напруги у колі відповідно до закону Ома ($V = I \times R$). Основна роль резистора - контроль електричного струму і захист чутливих компонентів, таких як світлодіоди (LED), транзистори чи мікросхеми. (рис. 11)

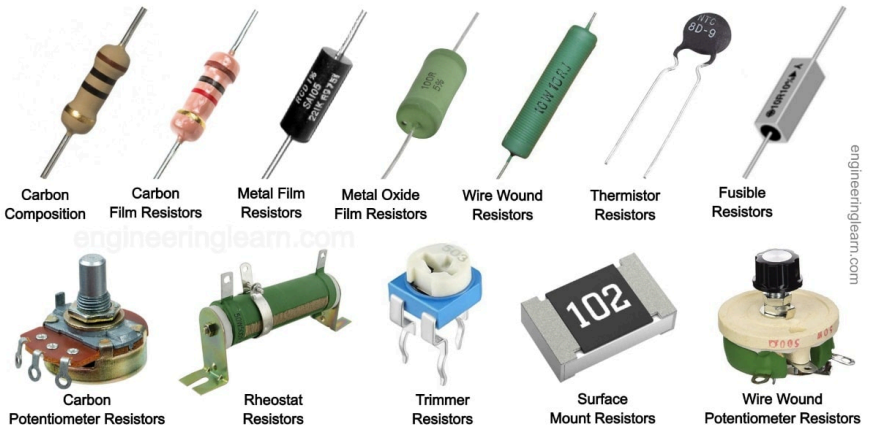


Рис. 11 - Зовнішній вигляд основних типів резисторів.

У середовищі Tinkercad Circuits використовується базовий тип резистора (Resistor) з можливістю змінювати параметр *Resistance* (опір) у меню праворуч.

Порядок роботи:

1. Виберіть компонент Resistor із категорії Basic Components.
2. Додайте його на плату або Breadboard.
3. У вікні параметрів задайте:
    - Name - ім’я компонента (наприклад, R1),
    - Resistance - значення опору (наприклад, 220 Ω, 1 kΩ, 10 kΩ).
4. Розмістіть резистор послідовно із світлодіодом або іншим елементом для обмеження струму.

Рекомендовані значення для LED:

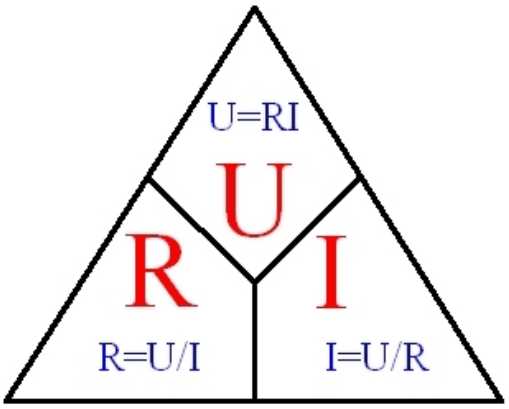
- 220 Ω → яскраве світіння
- 330 Ω → помірне (оптимальне)
- 1 kΩ → тьмяне, без перегріву

Практичне використання

У Tinkercad можна експериментувати зі зміною опору:

- зменшуйте опір - струм і яскравість зростають;
- збільшувати - яскравість зменшується.

Це наочно демонструє закон Ома і дозволяє зрозуміти вплив опору на струм у колі.



Резистори - це фундаментальний компонент електроніки, який використовується для:

- обмеження струму,
- поділу напруги,
- формування сигналів,
- захисту елементів схеми,
- калібрування та регулювання параметрів.

У середовищі Tinkercad Circuits резистор - це базовий компонент, який ви можете знайти у списку Basic components → Resistor. Він відображається як циліндричний елемент із кольоровими смугами, що позначають його номінал. Після розміщення на платі можна натиснути на компонент і у вікні праворуч задати значення опору у полі Resistance - наприклад, 220 Ω, 330 Ω або 1 kΩ.

На зображенні показано інтерфейс роботи з резистором у середовищі Tinkercad Circuits (рис. 12). Розглянемо ключові елементи:

- **#1** на зображенні - Компонент Resistor - обирається з панелі Basic components праворуч. Саме цей елемент додається на робочу область.
- **#2** на зображенні - Поле Resistance - тут задається номінал опору (значення у Омах, кілоомах, мегаомах). У Tinkercad значення можна ввести вручну, наприклад 220, 330, 1000, а праворуч вибрати одиницю виміру (Ω, kΩ, MΩ).
- **#3** на зображенні - Поле Name - ідентифікатор резистора на схемі. За замовчуванням позначається як R1, R2, тощо. Його можна перейменувати для зручності (наприклад, R_LED).
- **#4** на зображенні - Графічне позначення - на схемі резистор зображено як циліндр з кольоровими смугами, які відповідають його реальному номіналу. Колірні коди моделюють стандартну резисторну шкалу (наприклад, коричневий-чорний-червоний ≈ 1 кΩ).

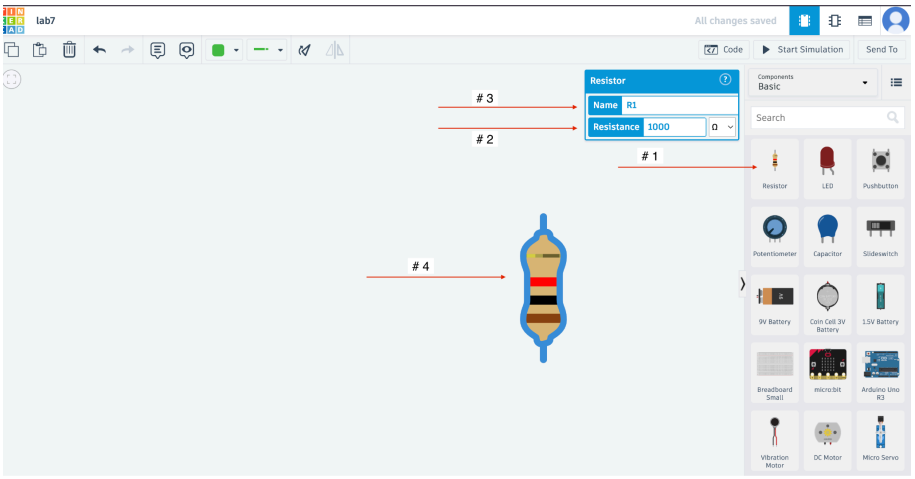


Рис. 12 - Редагування властивостей та зовнішній вигляд резистора.

Джерело живлення (Power Supply)

Джерело живлення - це компонент, який забезпечує електричною енергією усі інші елементи схеми. Воно створює різницю потенціалів між двома точками кола, завдяки чому через провідники протікає струм.

Роль у цифровій електроніці

У цифрових схемах **рівні напруги** відповідають **логічним станам**:

- **Логічна “1” (HIGH)** - висока напруга (наприклад, +5 В у TTL або +3.3 В у CMOS-системах).
- **Логічний “0” (LOW)** - нульова напруга (0 В або GND).

Таким чином, подання напруги на вхід світлодіода або мікросхеми означає передачу сигналу “1”, а її відсутність - “0”.

У середовищі **Tinkercad Circuits** користувач може вибирати різні типи джерел живлення (Power Supply) для створення електричних кіл. Кожне з них має свою напругу, тип елементів живлення та поведінку в симуляції.

Види джерел живлення:

Тип батареї	Напруга	Особливості
9V Battery	9 В	Найпоширеніша для живлення навчальних схем, Arduino, LED-ланцюгів. Має два контакти - “+” (червоний) і “–” (чорний).

1.5V Battery Pack	1.5 В × кількість елементів (зазвичай 3 = 4.5 В)	Можна змінювати кількість батарейок (**Count**) і тип (**AA, AAA, C, D**). Дає змогу моделювати ланцюги з нижчою напругою.
Coin Cell 3V Battery	3 В	Маленьке джерело живлення для простих схем - індикаторів, годинників, мікроконтролерів.
Power Supply (DC)	(налаштовується вручну)	Використовується для точного задання напруги, наприклад 5 В або 12 В.

Правила користування джерелами живлення у Tinkercad

1. **Додавання компонента**

   У панелі **Components → Basic** оберіть потрібний тип батареї:

   - *9V Battery*
   - *1.5V Battery*
   - *Coin Cell 3V Battery*

2. **Редагування параметрів**

   Натисніть на компонент, щоб відкрити параметри справа:

   - **Name** - задайте ім’я (наприклад, B1 або Battery_9V).
   - **Count** (для 1.5V Battery) - кількість батарейок у блоці (1–4).
   - **Built-in Switch** - чи має батарея вбудований вимикач (*yes/no*).

3. **Полярність**

   - Червоний контакт **“+”** - позитивний полюс.
   - Чорний контакт **“–”** - негативний полюс (GND).
   - Приєднуйте червоний дріт до навантаження (через перемикач, LED, резистор), а чорний - до **землі (GND)**.

4. **Підключення у схемі**

   - +9V → Switch → Resistor → LED → GND
   - або для меншої напруги: +3V (Coin Battery) → LED → GND

     Це дозволяє візуально демонструвати логічні стани **“1” (ON)** і **“0” (OFF)**.

Види батарей у Tinkercad (з прикладу)

9V Battery

- Доступна в **Basic components**.
- Використовується для живлення схем із більшим струмом (декілька LED, Arduino).

Після розміщення можна задати ім’я (Name: 1).

1.5V Battery (AA, AAA) Дає змогу налаштувати кількість елементів:

- Count: 3 batteries → 4.5 V
- Type: AA → стандартна
- Built-in Switch: no

Використовується для схем із меншою напругою.

Coin Cell 3V Battery

- Маленьке джерело на 3 В (CR2032).
- Підходить для тестів або мікросхем із низьким споживанням.
- У параметрах змінюється лише **Name**.

Поради при побудові схем

- **Не перевищуйте допустиму напругу** для LED (більше 3 В - тільки через резистор).
- **Пам’ятайте про полярність**: якщо підключити навпаки - елементи не працюватимуть.
- **Завжди додавайте резистор** у коло з LED.
- **Використовуйте різні батареї**, щоб спостерігати, як змінюється яскравість світлодіода при різній напрузі.
- У **Start Simulation** можна перевіряти поведінку кола в реальному часі.

Завдання для виконання

Завдання №1: Побудувати три електричні схеми в середовищі Tinkercad Circuits, де за допомогою перемикача (Pushbutton або Slideswitch) подається сигнал на світлодіод. Для кожної схеми використати різні джерела живлення:

1. **Coin Cell 3V Battery**

2. **1.5V Battery (3 шт.) → 4.5V**
3. **9V Battery**

Кінцева мета - продемонструвати подання логічних станів:

- **1 (ON)** - коли коло замкнуте, LED світиться.
- **0 (OFF)** - коли коло розімкнуте, LED не світиться.

Кроки реалізації

1. **Вибір компонентів (Basic components):**
    - Battery (обрати тип: 3V, 4.5V або 9V)
    - Slideswitch або Pushbutton
    - Resistor (220–330 Ω)
    - LED
    - Провідники (Wires)
2. **Побудова схеми:**
    - З’єднайте **позитивний полюс (+)** батареї з перемикачем.
    - Вихід перемикача під’єднайте до одного боку **резистора**,
    - Інший кінець резистора - до **анода LED**.
    - **Катод LED** з’єднайте із **GND (–)** батареї.
    - Для перевірки логічних станів запустіть **Start Simulation**.
3. **Спостереження результатів:**
    - У положенні **ON** - LED світиться (логічна 1).
    - У положенні **OFF** - LED не світиться (логічна 0).
    - Порівняйте яскравість LED при різних напругах.

Додаткові кроки для поглибленого розуміння роботи з електронними компонентами у Tinkercad Circuits:

1. **Додайте другий LED** - для демонстрації інверсії сигналу (один світиться, інший гасне).
2. **Підключіть мультиметр** - виміряйте напругу на LED у станах 0 і 1.
3. **Замість звичайного перемикача використайте Pushbutton** - щоб світлодіод світився лише при натисканні.
4. **Створіть логічну схему “АБО” або “І”** - додавши два перемикачі.

Лабораторна робота №9

Побудова логічних схем із вентилів AND, OR, NOT, XOR

Теоретичні відомості

Логічні вентилі та комбінаторні схеми

Логічний вентиль – це базовий елемент цифрового пристрою, який реалізує елементарну логічну операцію над вхідними бінарними сигналами і формує на виході бінарний результат. Кожен вентиль оперує двома рівнями сигналів: логічний 0 (**низький рівень напруги**, наприклад 0 В) та логічна 1 (високий рівень, наприклад +5 В). Вентилі можна з'єднувати між собою, підключаючи вихід одного до входу іншого, утворюючи таким чином більш складні схеми. Сукупність вентилів, з'єднаних без утворення петлі зворотного зв'язку, утворює комбінаторну логіку – в таких схемах вихідні сигнали повністю визначаються поточними значеннями вхідних сигналів і не залежать від попередніх станів системи (тобто не мають пам'яті).

Основні види логічних вентилів: У цифровій електроніці виділяють кілька фундаментальних логічних операцій. Розглянемо чотири з них, які складають базис для побудови довільних булевих функцій:

AND (логічне «І»),

OR (логічне «АБО»),

NOT («НЕ», інвертор) та

XOR («виключне АБО»).

Нижче наведено визначення цих операцій та відповідні **таблиці істинності** – всі можливі комбінації вхідних змінних і очікуваний вихід.

- **Логічне "І" (AND)** – вентиль, що реалізує **кон'юнкцію**. Вихід **A AND B** дорівнює 1 лише тоді, коли обоє вхідних сигналів **A** і **B** рівні 1. В усіх інших випадках (якщо хоча б один з входів 0) на виході буде 0. Таким чином, AND виконує функцію логічного множення (добутку): наприклад, $1 \cdot 1 = 1$, а $1 \cdot 0 = 0$ (так само $0 \cdot 0 = 0$). Таблиця істинності для AND-вентиля:

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

- **Логічне "АБО" (OR)** – вентиль, що реалізує **диз'юнкцію**. Вихід **A OR B** дорівнює 1, якщо **хоча б один** із вхідних сигналів дорівнює 1. Тобто **1** при входах (1,0), (0,1) або (1,1); лише коли обидва входи 0 – результат 0. OR виконує функцію логічного додавання: 1 + 0 = 1, 1 + 1 = 1. Таблиця істинності OR-вентиля:

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

- **Логічне "НЕ" (NOT)** – вентиль, відомий як **інвертор**. Виконує операцію логічного заперечення: вихідний сигнал **NOT A** є протилежним до вхідного **A**. Якщо на вхід подано 1, інвертор видає 0; якщо на вхід 0 – на виході отримуємо 1. Цей вентиль має лише один вхід і один вихід. У булевій нотації операція заперечення позначається рискою зверху **¬A**. Таблиця істинності інвертора:

A	NOT A
0	1
1	0

- **Виключне АБО (XOR)** – вентиль, що реалізує **операцію виключного або**. Вихід **A XOR B** дорівнює 1 **лише тоді, коли рівно один** з вхідних сигналів

дорівнює 1 (тобто, коли входи різняться). Якщо ж обидва входи однакові – обидва 0 або обидва 1 – на виході буде 0. Іншими словами, XOR виконує складання двох одно-бітових чисел без переносу: $1 \oplus 1 = 0$ (сума 0, перенос 1 який відкидається), $1 \oplus 0 = 1$, $0 \oplus 0 = 0$. Таблиця істинності XOR-вентиля:

A	**B**	**A XOR B**
0	0	0
0	1	1
1	0	1
1	1	0

З наведеної теорії можна зробити висновок, що комбінацією вентилів AND, OR, NOT можна реалізувати будь-яку логічну функцію від набора вхідних змінних. Набір {AND, OR, NOT} утворює так званий функціонально повний базис булевої алгебри.

Транзистор як електронний перемикач

Уявіть, що **транзистор** - це **розумний вимикач**, який може **вмикатись або вимикатись сам**, коли до нього приходить маленький сигнал. Якщо звичайний вимикач натискає людина, то транзистор “натискає” електричний сигнал. Тобто: Транзистор - це електронний керований перемикач. Він може сам керувати струмом у колі - без людини.

Концептуальна схема **транзистор** наступна, він має **три ніжки**:

1. **База (B)** - це “кнопка”, яка керує всім.
2. **Колектор (C)** - сюди “приходить” основний струм.
3. **Емітер (E)** - звідси струм “виходить”.

Принцип роботи доволі простий:

- Якщо до **бази** подати **маленький струм** - транзистор відкривається, і через нього може пройти **великий струм** між колектором і емітером.

- Якщо струму на базі **немає** - транзистор закритий, і струм **не тече**.

Для чого потрібен транзистор

1. Як перемикач. Він може вмикати чи вимикати частини схеми - наприклад, лампочку або мотор. Але робить це сам, без натискання кнопки. Наприклад, коли надходить сигнал із датчика або мікроконтролера.
2. Як підсилювач. Маленький сигнал (наприклад, від мікрофона) стає більшим - так працюють радіо, колонки, телефони. Транзистор бере на “базу” слабенький сигнал, та підсилює його за рахунок великого струму, який проходить між колектором і емітером.

Чому не можна просто замінити транзистор на звичайний вимикач?

Транзистор вимикач працює тільки тоді, коли людина його натисне. А транзистор вмикається сам - коли приходить електричний сигнал. Він реагує швидше, ніж будь-яка людина, і може працювати мільйони разів за секунду. Тому саме транзистори дозволяють створювати комп’ютери, телефони, логічні схеми, процесори - усе, що працює автоматично.

Де студенти можуть побачити транзистор у житті

- у комп’ютері - у процесорі мільярди крихітних транзисторів;
- у підсилювачах звуку - вони збільшують слабкий сигнал від мікрофона;
- у сенсорних лампах - вмикають світло, коли ви торкаєтесь корпусу;
- у радіо, телевізорах, зарядках, іграшках - майже всюди, де є електроніка.

У Tinkercad

У симуляторі Tinkercad Circuits транзистор працює точно так само: ви можете підати сигнал на базу, і побачите, як вмикається лампочка або запускається мотор. Якщо прибрати сигнал - лампочка гасне. Це показує, що транзистор перетворює логічний сигнал “1” (вмикай) у справжню дію в схемі (рис. 1).

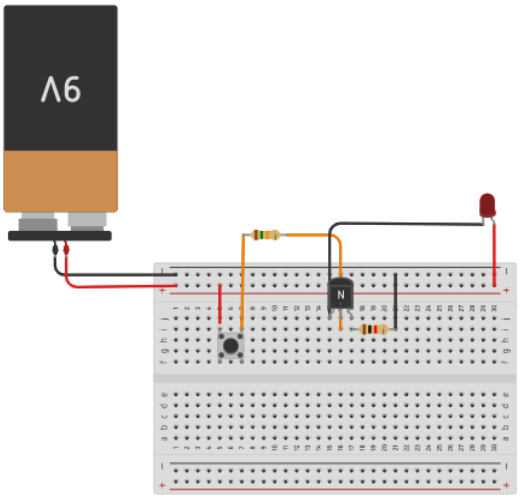


Рис. 1 - Приклад схеми керування навантаженням за допомогою транзистора.

Побудова логічних вентилів на транзисторах і базових компонентах

Маючи у розпорядженні **транзистори**, а також пасивні компоненти (**резистори**, **світлодіоди**) та **перемикачі** (для подачі вхідних сигналів 0/1), можна побудувати будь-який логічний вентиль. Основний принцип такий:

Вентиль NOT

Вентиль **NOT** або інвертор - це найпростіший приклад того, як транзистор може самостійно керувати струмом (рис. 2).

Він перевертає сигнал: якщо на вході «1» (струм є), то на виході буде «0» (струму немає). Якщо ж на вході «0», то на виході з’явиться «1». Тому він і називається *інвертором* - бо робить навпаки.

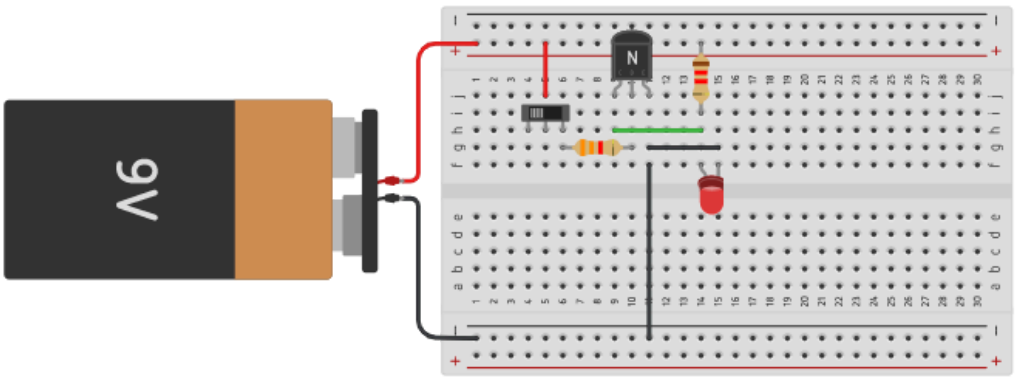


Рис. 2 - Реалізація логічного вентиля NOT у середовищі Tinkercad.

Як улаштована схема

На макетній платі зображено:

- **Транзистор NPN** - «серце» схеми.
    - Емітер (E) - до **землі (GND)**.
    - Колектор (C) - до **виходу (Output)**.
    - База (B) - до **входу (Input)** через **резистор**.
- **Резистор підтягування** - стоїть між виходом і **+ живленням (наприклад +5 В)**.
- **LED із резистором** - показує, коли на виході є сигнал (можна замінити на індикатор або логічний вузол).
- **Батарея 9 В** - джерело живлення для всієї схеми.

Як це працює

1. **Коли на вході є “1” (струм подається):**
    - Через резистор на базу надходить невеликий струм.
    - Транзистор відкривається - між колектором і емітером утворюється шлях для струму.
    - Транзистор “замикає” вихід на землю.
    - Вихід опускається до **0 В** (логічний нуль). *LED гасне* - бо вихід тепер з’єднано із землею.
2. **Коли на вході “0” (струму немає):**
    - Транзистор закритий, струм через нього не проходить.
    - Ланцюг колектора до землі розірвано.
    - Вихід через резистор підтягування підтягується до **+5 В** (логічна одиниця). *LED світиться* - бо на виході є напруга.

Чому це важливо

Ця схема показує головний принцип електроніки: **малий сигнал може керувати великим**. Тут маленький струм на базі керує великим струмом між

колектором і емітером. Саме на цьому принципі працюють усі комп’ютери - мільярди таких транзисторів формують логіку, пам’ять і процесори.

Вентиль AND

Вентиль AND - це логічний елемент, який видає «1» тільки тоді, коли обидва входи також рівні 1. Інакше кажучи: Умова виконується лише тоді, коли всі входи активні. Щоб реалізувати це фізично, потрібно використати два транзистори, з’єднані послідовно - як два вимикачі в одному ланцюгу (рис 3).

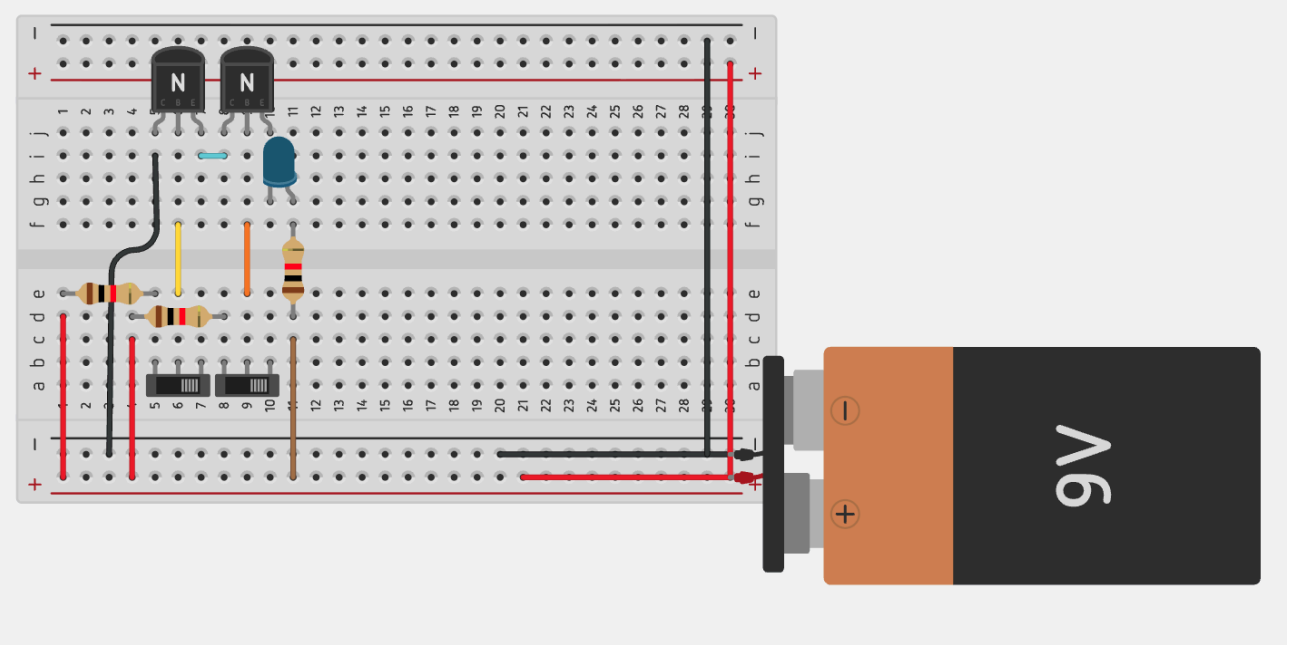


Рис. 3 - Реалізація логічного вентиля AND *у середовищі Tinkercad.*

Як улаштована схема

На макетній платі бачимо:

Два NPN-транзистори, з’єднані послідовно:

- Колектор верхнього транзистора - це вихід.
- Емітер верхнього підключено до колектора нижнього.
- Емітер нижнього йде до землі (GND).

Два входи (A і B) - кожен через окремий резистор подає сигнал на базу відповідного транзистора. Резистор підтягування (pull-up) - з’єднує вихід із + живленням (+5 В). Завдяки йому, якщо транзистори закриті, вихід утримується у

стані «1». LED із резистором - підключений до виходу для візуалізації результату. Живлення - батарея 9 В (або адаптоване до 5 В джерело).

Як це працює

Випадок 1: обидва входи = 1

- Обидва транзистори відкриті, струм проходить крізь них до землі.
- Вихід з'єднується із землею → низький рівень (0).
- Якщо після цього вихід інвертувати (через окремий NOT), отримаємо AND = 1.

  LED засвічується у схемі з інвертором.

Випадок 2: хоча б один із входів = 0

- Один із транзисторів закритий → ланцюг розірвано.
- Струм не проходить, вихід через підтягування стає високим (1).

  LED вимикається (у прямій схемі AND).

Підсумок логіки

Вхід A	**Вхід B**	**Вихід AND**	**Стан LED**
0	0	0	Вимкнений
0	1	0	Вимкнений
1	0	0	Вимкнений
1	1	1	Світиться

Як це пов'язано з комп'ютером

Вентиль AND - один із базових логічних елементів, на яких побудовані всі цифрові пристрої.

Він використовується для перевірки одночасного виконання кількох умов, наприклад:

- У процесорі: команда виконується лише якщо одночасно активні сигнали "готовність пам'яті" *і* "дозвіл на запис".

- У клавіатурі: спрацьовує комбінація клавіш, коли натиснуті дві кнопки одночасно.
- У мікроконтролерах: керування подіями, які мають виконуватися лише при збігу кількох сигналів.

Вентиль OR (логічне «АБО»)

Вентиль OR - це логічний елемент, який видає «1», якщо хоча б один із входів = 1. Інакше кажучи: Якщо хоча б один сигнал активний, то результат також активний (рис. 4).

Використання: якщо або кнопка A, або кнопка B натиснути → лампочка світиться. Лише якщо жодна не натиснута → темно.

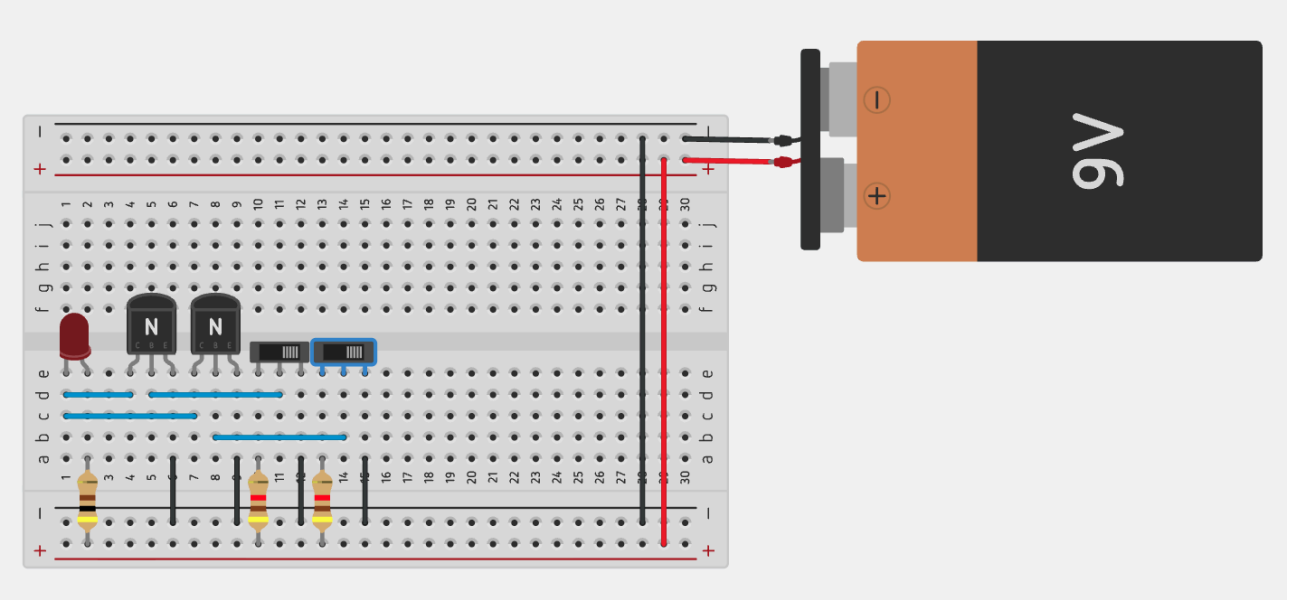


Рис. 4 - Реалізація логічного вентиля OR у середовищі Tinkercad.

Як улаштована схема

На макетній платі зображено:

- **Два NPN-транзистори** - вони з'єднані **паралельно** між виходом і землею.
  - Колектори обох транзисторів → спільна точка виходу (**Output**).
  - Емітери обох → до землі (**GND**).
  - Кожна база → свій вхід (**Input A** і **Input B**) через окремий резистор.

- **Резистор підтягування** - стоїть між виходом і **+ живленням** (+5 В). Коли транзистори закриті, він «підтягує» вихід догори.
- **LED із резистором** - показує, коли на виході є сигнал.
- **Батарея 9 В** - джерело живлення.

Як це працює

1. **Коли обидва входи = 0 (жоден транзистор не відкритий):**
    - Транзистори не проводять струм.
    - Вихід через резистор підтягування підтягується до **+5 В** → логічна 1. *LED світиться.*
2. **Коли хоча б один із входів = 1:**
    - Відповідний транзистор відкривається й підключає вихід до землі.
    - Напруга на виході падає до 0 В → логічний 0. *LED гасне.*

Підсумок логіки роботи

Вхід A	Вхід B	Вихід NOR	Після інвертора (OR)	Стан LED
0	0	1	0	Вимкнений
0	1	0	1	Світиться
1	0	0	1	Світиться
1	1	0	1	Світиться

Тобто схема фактично реалізує **NOR** (інверсію OR).

Щоб отримати чистий OR (де вихід = 1, якщо будь-який вхід = 1), потрібно **додати інвертор** - ще один транзистор після виходу.

Практичний зв’язок із комп’ютером

У мікропроцесорі логіка **OR** використовується для:

- **Об’єднання сигналів** - наприклад, коли потрібно активувати дію, якщо спрацювала будь-яка з подій (клавіша, сенсор, переривання).
- **Побудови умов “або”** у коді: if (A || B), коли процесор вирішує виконувати дію, якщо будь-яка умова істинна.

- **Формування сигналів “готовності” або “зайнятості”** у схемах керування пам’яттю чи портами вводу-виводу.

Вентиль **XOR** або **«виключне АБО»** - це логічний елемент, який видає «1» лише тоді, коли один із входів = 1, а інший = 0. Інакше кажучи: Результат істинний, якщо входи різні, і хибний, якщо входи однакові.

Звичайне життєве порівняння: Дві лампи керуються двома вимикачами - світло горить лише тоді, коли натиснута лише одна клавіша, а не обидві чи жодна.

Як улаштована схема

Вентиль XOR неможливо побудувати на одному або двох транзисторах, як попередні. Він поєднує AND, OR і NOT у єдину логіку (рис. 5):

XOR(A, B) = (A AND NOT B) OR (NOT A AND B)

На макетній платі схема містить: 5–6 NPN-транзисторів:

- Два транзистори формують інвертори (NOT A, NOT B).
    - Два транзистори утворюють AND-каскади для умов:
        - A AND NOT B
        - B AND NOT A
    - Ще один або два транзистори виконують об’єднання (OR) цих сигналів у спільний вихід.
- Два входи - A та B (через базові резистори до відповідних каскадів).
- Резистори підтягування (pull-up) - тримають вихід у високому стані, якщо каскади не проводять струм.
- LED із резистором - показує результат на виході (світиться лише коли один із входів активний).

- Живлення - 9 В або адаптоване до 5 В.

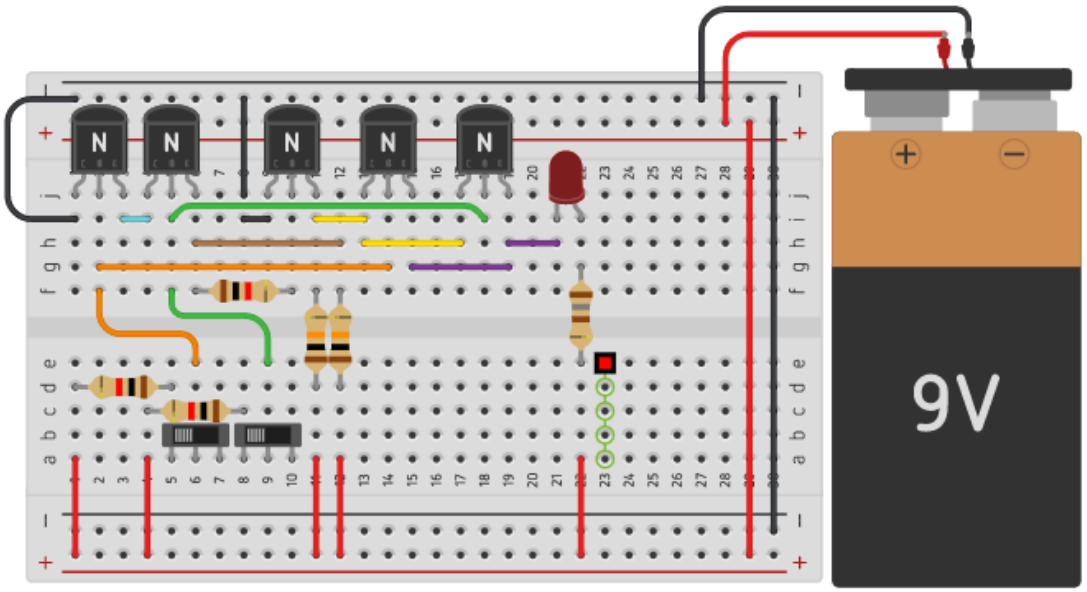


Рис. 5 - Реалізація логічного вентиля XOR у середовищі Tinkercad.

Як це працює

1. **A = 0, B = 0**
    - Обидва каскади не проводять струм.
    - Вихід через резистор підтягування залишається високим (1).
    - LED вимкнений.
2. **A = 0, B = 1**
    - Каскад «NOT A AND B» активується (бо NOT A = 1, B = 1).
    - Струм проходить - вихід стає низьким (0) → LED вмикається.
    - Вихід XOR = 1.
3. **A = 1, B = 0**
    - Каскад «A AND NOT B» активний.
    - Результат такий самий - LED світиться.
    - Вихід XOR = 1.
4. **A = 1, B = 1**
    - Обидва каскади проводять одночасно → вихід “шунтується” (струм розподіляється) → напруга падає → LED гасне.
    - Вихід XOR = 0.

Таблиця істинності

Вхід A	Вхід B	Вихід XOR	Стан LED
0	0	0	Вимкнений
0	1	1	Світиться
1	0	1	Світиться
1	1	0	Вимкнений

Як це пов’язано з комп’ютером

Вентиль **XOR** - один із найважливіших у цифровій логіці. Він використовується там, де потрібно **порівняти або додати два біти**:

- **Арифметичні операції** - при складанні двох бітів (у суматорі XOR формує “Sum”).
- **Порівняння даних** - виявлення різниці між двома сигналами.
- **Шифрування та контроль парності (parity check)** - XOR використовується для обчислення бітів перевірки, бо виявляє, чи змінилася кількість одиниць у наборі даних.

Завдання для виконання

Завдання №1: Реалізуйте вентиль NOT за допомогою транзистора, резистора, LED і кнопки.

1. Увімкніть симуляцію в Tinkercad і перевірте:
    - Input = 0 → LED світиться
    - Input = 1 → LED гасне
2. Зробіть скріншот робочої схеми.

Завдання №2: Реалізуйте вентиль AND та перевірте таблицю істинності:

A	B	Output	LED
0	0	0	Викл.

0	1	0	Викл.
1	0	0	Викл.
1	1	1	Увімк.

Завдання №3: Реалізуйте вентиль OR та перевірте таблицю істинності:

A	B	Output	LED
0	0	0	Викл.
0	1	1	Увімк.
1	0	1	Увімк.
1	1	1	Увімк.

Завдання №4: Реалізуйте вентиль XOR використайте комбінацію попередніх вентилів (NOT, AND, OR).

Перевірте таблицю істинності:

A	**B**	**Output**	**LED**
0	0	0	Викл.
0	1	1	Увімк.
1	0	1	Увімк.
1	1	0	Викл.

Завдання №5: Оптимізуйте XOR. Спробуйте зменшити кількість транзисторів у вашій схемі XOR до мінімуму (5 або навіть 4). Опишіть, які зміни ви внесли та покажи результати.

Лабораторна робота №10

Реалізація двійкового суматора на логічних елементах

Теоретичні відомості

У цифрових пристроях інформація представлена бітами (0 та 1). На рівні електронних схем ці стани відповідають різним рівням напруги: низький рівень - **логічний 0**, високий рівень - **логічна 1**.

Для обробки числових даних потрібно реалізувати арифметичні операції. Якщо додати два біти 0 та 1, то результат легко уявити: 0+0=0, 0+1=1, 1+0=1. Однак сума 1 + 1 дорівнює двом бітам, що у двійковій системі записується як 10, тобто сума дорівнює 0, а вищий розряд отримує перенос 1.

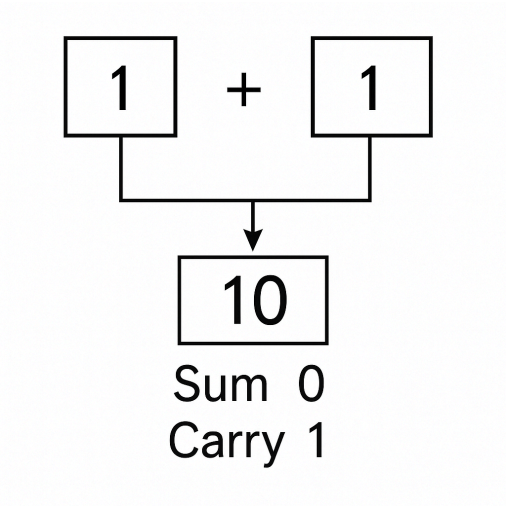


Рис. 1 - Приклад реалізації суми 1+1

Цей перенос повинен передаватися наступному розряду при додаванні багаторозрядних чисел, аналогічно тому як при додаванні десяткових чисел 9 + 1 = 10 переноситься в десятки.

У таблиці 1 наведені правила додавання двох одно-бітових чисел у двійковій системі. Крім результуючої суми S ще наведено утворений перенос C (carry), який стає вхідним сигналом для наступного розряду.

Вхід A	Вхід B	Сума S	Перенос C
0	0	0	0
0	1	1	0

1	0	1	0
1	1	0	1

Напівсуматор (half adder)

Напівсуматор - це найпростіший арифметичний блок, який додає два одно-бітові числа A та B і формує два виходи: **суму** (S) та **перенос** (C). Він не має входу для переносу, тому використовується лише для наймолодшого розряду багаторозрядного додавання (рис. 2). Суть його роботи відображено в таблиці істинності: для кожної комбінації входів наведено результати S і C[2]. З таблиці видно, що сума дорівнює 1, коли входи різні, а перенос - коли обидва входи 1. Формально ці функції описуються рівняннями:

- **Сума:** $S = A \oplus B$ - операція виключного «або» (XOR) дає 1 тільки тоді, коли A та B відрізняються[3].
- **Перенос:** $C = A \cdot B$ - елемент AND формує 1 лише при A=1 і B=1[4].

Таким чином, реалізація напівсуматора потребує лише двох вентилів: XOR і AND. На рис. 2 показано структурну схему: два входи A і B підключаються одночасно до XOR-вентиля (формує суму) та до AND-вентиля (формує перенос). Виходи S та C можуть керувати світлодіодами для візуального контролю.

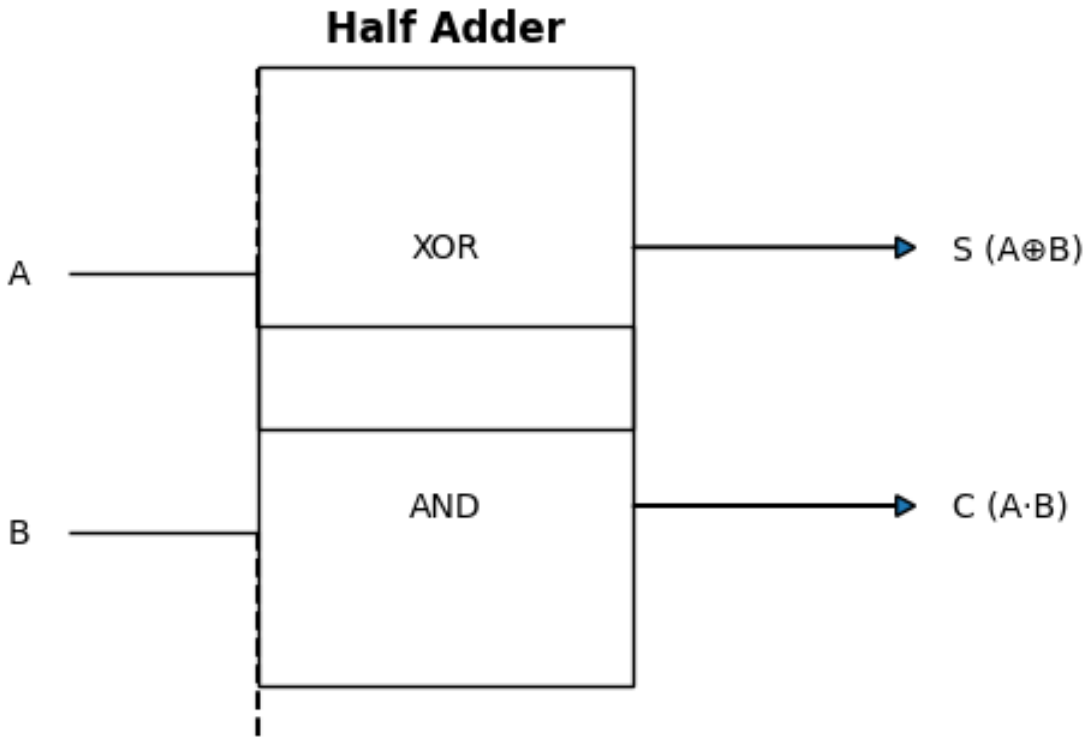


Рис. 2 - Принципова логічна схема напівсуматора.

Чому напівсуматор недостатній для багаторозрядного додавання? Під час додавання декількох бітів перенос із молодшого розряду повинен враховуватися у старшому. Напівсуматор не має входу для переносу, отже не здатен коректно додати, наприклад, три сигнали 1, 1 і перенесену 1. Для цього потрібен повний суматор.

Повний суматор (full adder)

Повний суматор приймає три входи: біти A та B і біт переносу C від молодшого розряду. Він формує два виходи: суму S та перенос C у наступний старший розряд. У загальному випадку повний суматор реалізує додавання трьох бітів, тому його таблиця істинності має 8 комбінацій (табл. 2). Функції S та Смають вигляд:

$$S = (A \oplus B) \oplus C_{in}, \quad C_{out} = A \cdot B + C_{in} \cdot (A \oplus B)$$ [6].

Тобто сума виходить операцією XOR трьох входів, а перенос - об’єднанням двох умов: перенос буде 1, коли одночасно A=1 і B=1 або коли C=1 і один з входів (A

або B) також 1. Повний суматор можна побудувати з двох напівсуматорів та OR-вентиля (рис. 2): перший напівсуматор додає A та B та видає проміжну суму S_1 і перенос C_1; другий напівсуматор додає S_1 та C; переноси C_1 і C_2 об'єднуються OR-вентилем, формуючи C.

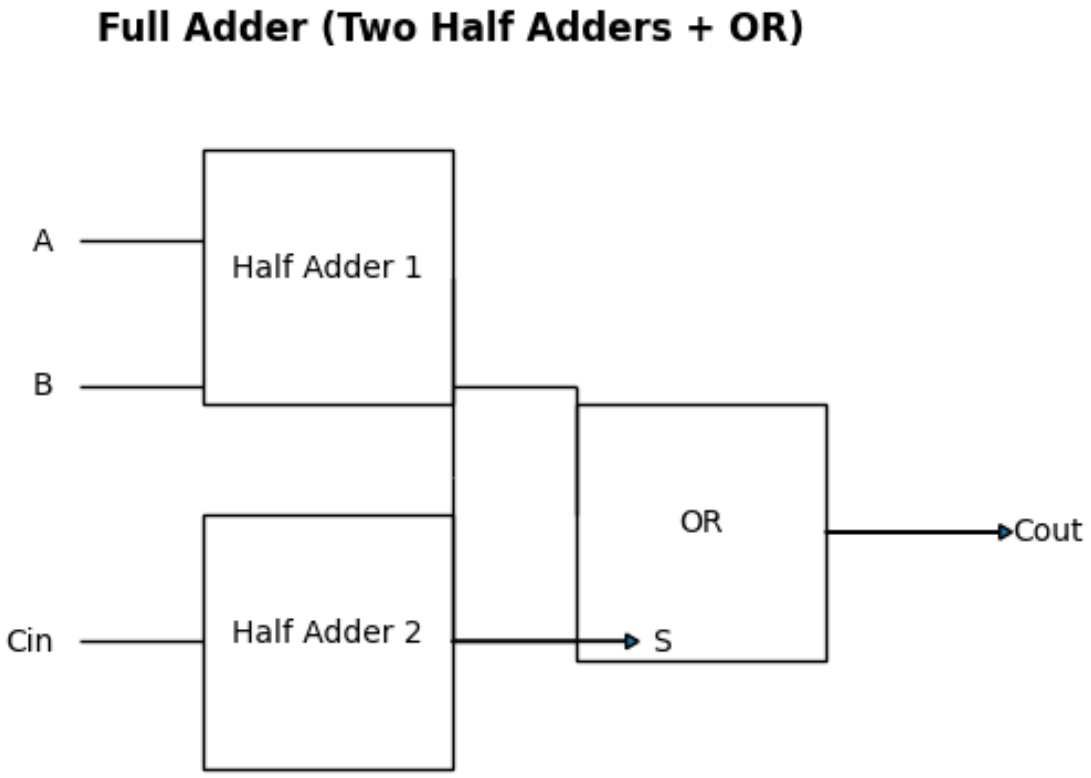


Рис. 3 - Принципова логічна схема повного суматора.

Таблиця 2 - **правильна таблиця істинності повного суматора**: (вхід C позначений Cin, вихідний перенос - Cout).

A	B	Cin	C out	Сума S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Багаторозрядні суматори (ripple carry adder)

Для складання двох багаторозрядних чисел необхідно каскадувати декілька повних суматорів: перенос виходу кожного молодшого розряду служить входом переносу для старшого (рис. 4). Таке з'єднання називають **пошаровим (ripple carry) суматором** - перенос «переливається» від найменшого до найстаршого розряду. Наприклад, двобітний суматор складається з двох повних суматорів: перший додає молодші біти A_0 та B_0 разом із C (звичайно 0), другий (FA**1**) - старші біти A_1 та B_1 разом з переносом C_1 від першого. На рис. 3 показано блок-схему двобітного суматора.

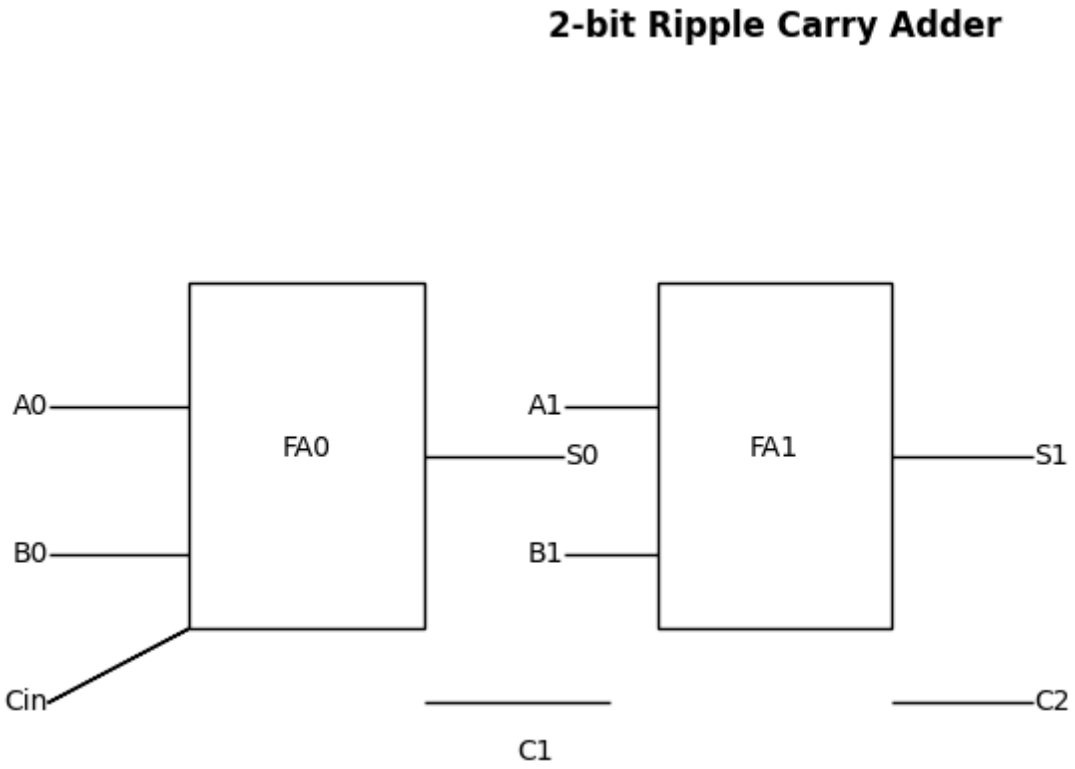


Рис. 4 - Принципова логічна схема багаторозрядного суматора.

Недолік пошарового суматора полягає у тому, що перенос повинен пройти через усі елементи, що збільшує затримку. На практиці для прискорення використовують схеми **carry-lookahead** або інші оптимізації, але основний принцип роботи суматора базується саме на з'єднанні повних суматорів.

Реалізація логічних елементів на транзисторах

У попередній лабораторній роботі ви ознайомилися з транзисторами як електронними керованими перемикачами. Транзистор **NPN** має три електроди: **база (B)**, **колектор (C)** та **емітер (E)**. База отримує керуючий сигнал, колектор - вхід основного струму, емітер - вихід на землю. Коли на базі з'являється невеликий струм, транзистор відкривається і між колектором та емітером може протікати значний струм; коли на базі сигналу немає - транзистор закритий. Використовуючи транзистори й резистори, можна створити будь-які логічні елементи:

- **NOT** - один транзистор та підтягувальний резистор реалізують інвертор: при подачі логічної 1 на базу транзистор з'єднує вихід із землею, формуючи 0; при 0 на вході транзистор закритий, і резистор підтягує вихід до логічної 1.
- **AND** - два NPN-транзистори, увімкнені послідовно; перенос виникає, коли обидва відкриті, тому вихід 0; щоб отримати стандартну функцію AND (1 коли обидва входи 1), після такого «транзисторного» AND додається інвертор.
- **OR** - два транзистори увімкнені паралельно між виходом і землею; принаймні один відкритий - вихід закорочується на землю (логічний 0), тому після інвертора отримуємо функцію OR.
- **XOR** - поєднання елементів AND, OR і NOT згідно з формулою $A \oplus B = (A \wedge \neg B) \vee (\neg A \wedge B)$, що потребує 5–6 транзисторів (можливо скорочення до 4).

У цій роботі для побудови суматорів ми використовуватимемо логічні вентилі, доступні в Tinkercad, але за бажанням можна реалізувати їх із транзисторів за аналогією до попередньої лабораторної роботи.

Завдання для виконання роботи

Завдання №1: Створення напівсуматора за допомогою транзисторів

1. Побудуйте напівсуматор: розмістіть XOR-вентиль і AND-вентиль. Вхідні сигнали A і B реалізуйте двома перемикачами (Slide Switch), підключаючи один контакт кнопки до + живлення (логічної 1), а інший - до вхідної клеми вентиля. Подаючи напругу через резистор (1 kΩ) на базу транзистора у варіанті зі схемою на транзисторах, ви імітуєте логічний сигнал.
2. Підключіть вихід XOR через резистор 220 Ω до світлодіода, щоб візуалізувати суму S. Вихід AND підключіть через резистор 220 Ω до другого світлодіода, який відображатиме перенос C. Не забудьте під'єднати катод світлодіода до землі.
3. Увімкніть симуляцію і перевірте таблицю істинності: для всіх чотирьох комбінацій входів (00, 01, 10, 11) зафіксуйте, чи світяться світлодіоди S і C. Зробіть скріншот робочої схеми із підписами.

Завдання 2: Створення повного суматора за допомогою транзисторів

1. Побудуйте повний суматор двома способами:
2. Комбінація двох напівсуматорів і OR-вентиля. Розмістіть два набори вентилів XOR та AND, як у завданні 1. З'єднайте їх відповідно до схеми на рис. 2: входи A і B подайте на перший напівсуматор, його вихід S_1 та вхід переносу Cin - на другий напівсуматор. Виходи C_1 та C_2 об'єднайте OR-вентилем для формування Cout.
3. З використанням готового Full Adder (якщо Tinkercad надає такий компонент). Розмістіть Full Adder і подайте на нього три входи: A, B і Cin. Виходи S і Cout підключіть через резистори до світлодіодів.
4. Для кожної комбінації трьох входів (000…111) заповніть таблицю: визначте, коли світяться світлодіоди S та Cout. Перевірте відповідність результатів таблиці істинності (табл. 2).
5. Скріншот: виконайте знімок екрана схеми, на якому видно підключення всіх елементів. Підпишіть вхідні та вихідні сигнали.

6. Коментарі: зверніть увагу, що суматор працює навіть при відсутності готових логічних вентилів - достатньо мати тільки транзистори та резистори. Поясніть у звіті, які елементи ви використали для реалізації XOR, AND та OR.

Завдання №3: Створення Дворозрядного суматора за допомогою транзисторів

1. Побудуйте двобітний суматор (ripple carry adder) із двох повних суматорів, з'єднавши їх так, як показано на рис. 3. Старший суматор отримує перенос C_1 від молодшого. Вхід переносу Cin для першого суматора встановіть в нуль (можна з'єднати з землею через кнопку). Виходи S_0 та S_1 підключіть до світлодіодів для індикації двобітної суми. Вихідний перенос C_2 також виведіть на окремий світлодіод.
2. Зафіксуйте роботу для всіх 16 комбінацій двох двобітних чисел A_1A_0 і B_1B_0 (від 00 до 11). Складіть таблицю з результатами (S_1S_0 і C_2). Переконайтесь, що перенесення від першого суматора впливає на другий.
3. Експеримент: змініть вхідний перенос Cin на 1 і спостерігайте, як змінюються результати. Це моделює додавання до суми додаткової 1 (наприклад, при обчисленні суми з урахуванням переносу з попередніх операцій).

Лабораторна робота №11

Формування байтів і перетворення у десяткову та шістнадцяткову системи

Теоретичні відомості

Біт, тетрада, байт та машинне слово

Біт (binary digit) – найменша одиниця інформації, яка може набувати значень 0 або 1. Фізично біт відповідає напрузі, струму чи заряду: наприклад, 0 В ≈ «логічний 0», а 5 В ≈ «логічна 1».

Тетрада або **нібл** – послідовність із чотирьох бітів. Чотири біти дозволяють представити до 16 різних станів, що достатньо для кодування однієї десяткової цифри (0–9) або однієї шістнадцяткової цифри (0–F). Тетради часто використовують для подання **BCD-декодера**, а дві тетради формують **байт**.

Позиційні системи числення

У **позиційній системі числення** значення цифри залежить від її позиції (розряду) та основи системи. Ми розглянемо три системи, які найчастіше використовують у комп'ютерній техніці.

- **Десяткова система (основа 10)** – цифри 0…9. Наприклад, $372_{10} = 3\cdot10^2 + 7\cdot10^1 + 2\cdot10^0$.
- **Двійкова система (основа 2)** – цифри 0 й 1. Число $1101_2 = 1\cdot2^3 + 1\cdot2^2 + 0\cdot2^1 + 1\cdot2^0 = 13_{10}$.

BCD-декодер (Binary-Coded Decimal)

BCD-декодер дозволяє кодувати **кожну десяткову цифру** окремою тетрадою. У **натуральному BCD 8421** для кожної цифри 0–9 використовується чотири біти (8421 – ваги бітів). Перевага BCD у тому, що кожна цифра подається незалежно, тому легко відображати числа на індикаторах.

Таблиця відповідності для натурального BCD (8421):

Десяткова цифра	Біт 8	Біт 4	Біт 2	Біт 1	BCD (8421)
0	0	0	0	0	0000_2
1	0	0	0	1	0001_2
2	0	0	1	0	0010_2
3	0	0	1	1	0011_2
4	0	1	0	0	0100_2
5	0	1	0	1	0101_2
6	0	1	1	0	0110_2
7	0	1	1	1	0111_2
8	1	0	0	0	1000_2
9	1	0	0	1	1001_2

Коди $1010_2 \ldots 1111_2$ у BCD 8421 **не використовуються**. Якщо на вхід BCD-дешифратора подати таку комбінацію, на індикаторі засвітиться «неправильний» символ.

1.4 Інтегральні мікросхеми

Інтегральна мікросхема (IC) – це кристал напівпровідника, на якому у спільному корпусі розміщено десятки, сотні або мільйони транзисторів, резисторів та інших елементів (рис. 1). На відміну від дискретних компонентів (окремих транзисторів, діодів), IC є компактними, надійними та дешевими.

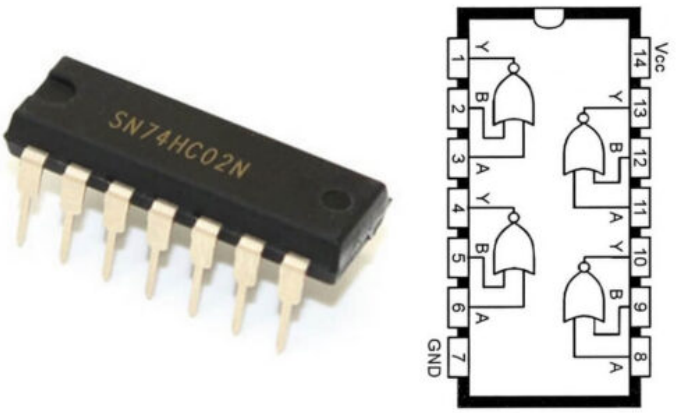


Рис. 1 - Інтегральна мікросхема

За **рівнем інтеграції** виділяють:

- **SSI (Small-Scale Integration)** – до 100 елементів; прості логічні вентилі та базові мікросхеми;

- **MSI (Medium-Scale Integration)** – від 100 до 10 000 елементів; дешифратори, мультиплексори, лічильники;
- **LSI (Large-Scale Integration)** – 10 000–100 000 елементів; прості мікропроцесори, пам’ять;
- **VLSI/ULSI** – сотні тисяч і більше елементів; сучасні процесори і мікроконтролери.

За функцією ІС поділяють на **комбінаційні** (вихід залежить лише від поточного набору входів) та **послідовні** (мають пам’ять, залежність від попередніх станів). Приклади комбінаційних ІС – логічні елементи AND, OR, XOR; послідовні – тригери, лічильники, регістри.

1.5 Логічні елементи 74HC08, 74HC32, 74HC86

Мікросхеми серії 74HC реалізують базові логічні функції. Кожен чіп містить чотири однакові елементи.

1. **74HC08** – містить чотири **AND-вентилі**. **AND** (кон’юнкція) видає 1 лише тоді, коли всі входи дорівнюють 1. Відомо, що AND-вентиль виводить 1 (HIGH) тільки якщо всі його входи HIGH. Таблиця істинності для двох входів:

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

2. **74HC32** – містить чотири **OR-вентилі**. **OR** (диз’юнкція) видає 1, якщо хоча б один із входів дорівнює 1. У 74HC32 вихід високий, якщо хоча б один вхід HIGH. Таблиця істинності:

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

3. **74HC86** – містить чотири **XOR-вентилі**. **XOR** (виключне або) видає 1 лише тоді, коли входи різні: один 1, інший 0. Таблиця істинності:

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Завдяки цим базовим елементам можна реалізувати практично будь-яку булеву функцію.

1.6. 7-сегментні індикатори

7-сегментний індикатор – це компактний модуль із семи світлодіодів, розташованих у формі цифри «8». Освітлюючи різні комбінації сегментів (a…g), можна відображати цифри 0–9 та деякі літери (рис. 2). Додатковий восьмий світлодіод часто використовується як **десяткова крапка**.

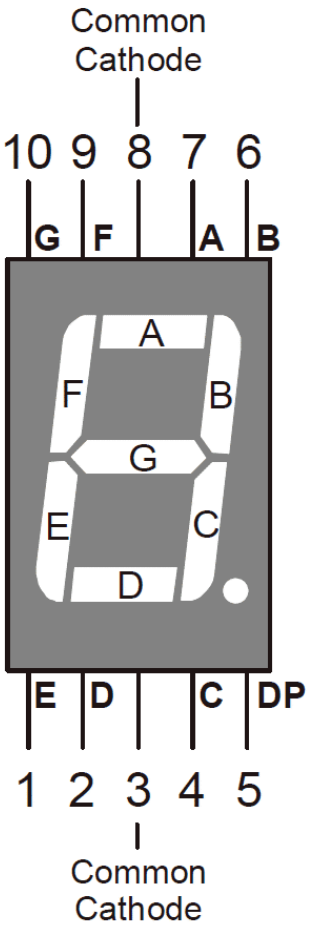


Рис. 2 - 7 сегментні індикатори

Існує два типи 7-сегментних індикаторів:

1. **Common Cathode (CC)** – усі катоди сегментів з’єднані та підключаються до «землі» (логічного 0). Кожен сегмент світиться, коли на його анод подається логічна 1 через обмежувальний резистор.
2. **Common Anode (CA)** – усі аноди з’єднані й підключаються до логічної 1. Сегмент світиться, коли його катод підтягується до землі (логічний 0) через резистор.

Змінювати тип індикатора в схемі без відповідної заміни драйвера **не можна**, оскільки CC і CA підключаються за протилежною полярністю. У нашій лабораторній роботі використовують індикатор із загальним катодом.

Чому потрібні резистори? Кожен сегмент – це світлодіод. Без обмежувального резистора струм через LED може перевищити допустимий рівень і зіпсувати деталь. Коли діод приводять у пряме зміщення, напруга на ньому близько 2 В (для червоних LED). Отже, при живленні 5 В через резистор на 220 Ω струм становить ≈ (5 В – 2 В)/0,015 А ≈ 200 Ω, що забезпечує надійне світіння.

1.7 BCD-декодер CD4511

CD4511 - це декодер BCD-7-сегментного коду. Це означає, що він приймає число у двійковій формі на вхід, а потім відображає це число на 7-сегментному індикаторі, використовуючи свої виходи (рис. 3).

7-сегментний індикатор – це компонент із сімома світлодіодами (LED), розташованими, як показано нижче. Увімкнення різних комбінацій світлодіодів відображає число від 0 до 9.

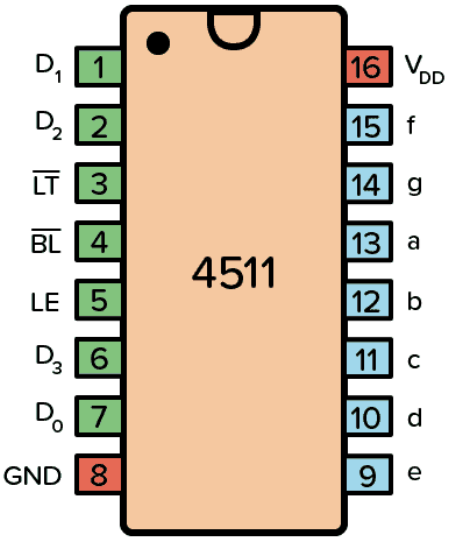


Рис. 3 - Схема виводів (pinout) мікросхеми CD4511.

Огляд пінів

Назва PIN-кода	PIN-код	Тип	Опис
VDD	16	Потужність	Напруга живлення (від +3 до +15 В)
GND	8	Потужність	Заземлення (0 В)
a-f	9-15	Вихід	Виходи для 7-сегментного індикатора
D0-D3	7, 1, 2, 6	Вхід	4-бітний вхід даних
LT	3	Вхід	Тестування лампи. Вмикає всі сегменти при низькому рівні напруги.
BL	4	Вхід	Тест на гашення. Вимикає всі сегменти при низькому рівні.
LE	5	Вхід	Увімкнення фіксації. Зберігає поточний стан при високому рівні.

Використовувати CD4511

Щоб мати змогу використовувати декодер BCD-7-сегментного коду в мікросхемі, спочатку потрібно підключити **контакт VDD** до позитивного виводу живлення, а **контакт GND –** до негативного виводу живлення.

Ви можете використовувати напругу живлення від 3 В до 15 В. Хоча деякі версії мікросхеми 4511 підтримують до 20 В. Перевірте технічні характеристики вашої версії мікросхеми, щоб дізнатися точні значення.

Виводи D0, D1, D2, D3 – це входи BCD, через які ви подаєте число, яке хочете відобразити на дисплеї у двійковому форматі.

Виводи від a до g – це вихідні виводи, які ви підключаєте до 7-сегментного індикатора.

Контакт LT **(Lamp Test - тест лампи)** призначений для перевірки роботи всіх сегментів дисплея. Встановіть LOW (низький рівень) для перевірки сегментів. Встановіть HIGH (високий рівень) для нормальної роботи **.**

Вивід BL (**тест гасіння)** вимикає всі сегменти при низькому рівні. Ви можете використовувати його для керування яскравістю дисплея за допомогою широтно-імпульсної модуляції (ШІМ). Встановіть значення HIGH для нормальної роботи.

Вивід **LE (Latch Enable - увімкнення фіксації)** , який також називають *сховищем* , використовується для зберігання поточного значення. Коли він має високий рівень, останні дані відображаються незалежно від змін на входах BCD. Встановіть цей вивід у низький рівень для нормальної роботи.

Завдання для виконання роботи

Завдання №1: Реалізує перетворення 4-бітного коду в десяткову цифру на 7-сегментний індикатор (рис. 5).

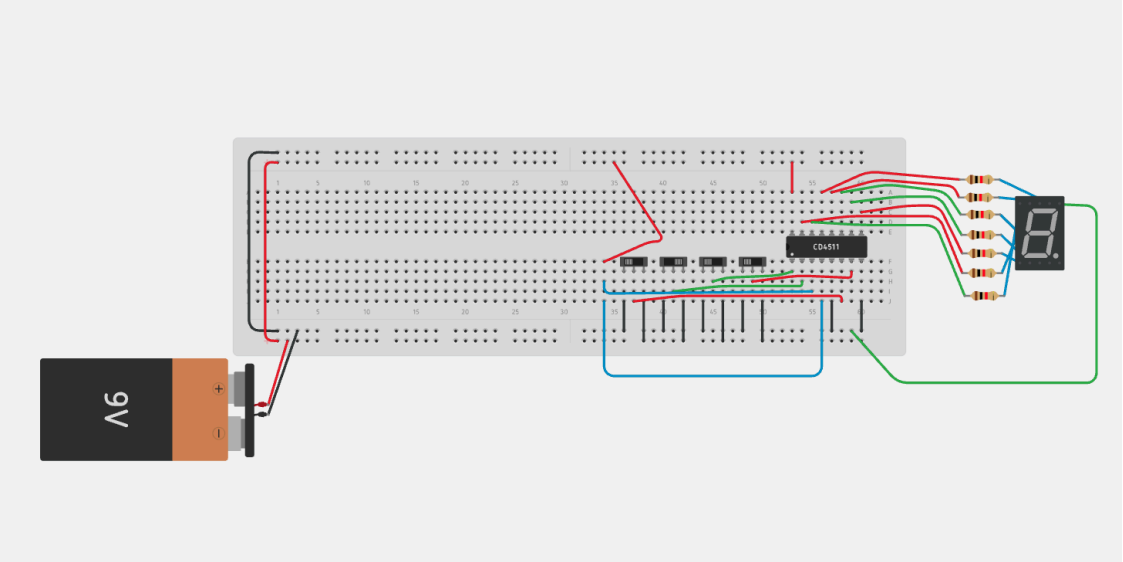


Рис. 5 - Схема перетворення BCD → 7-сегментний індикатор

1. Створіть проект. Відкрийте сайт tinkercad.com, увійдіть у свій акаунт та створіть нову схему Circuits в класі.
2. Додайте компоненти: макетну плату, батарею 9 В (або джерело 5 В), мікросхему CD4511, 7-сегментний індикатор (common cathode), 4 перемикачі, 7 резисторів по 1 кΩ (сегментні), 4 резистори 10 кΩ (підтягувальні).
3. Організуйте живлення: під'єднайте плюсову клему батареї до позитивної шини макетної плати, мінус – до негативної. Проведіть перемички, щоб обидві шини були під напругою.
4. Підключіть CD4511:
    - VDD – до плюса;
    - GND – до мінуса;
    - входи input 1-4 – до перемикачів до землі;
    - LE – до землі (0);
    - LT, BL – до плюса (1).
5. Підключіть індикатор:
    - загальний катод – до GND;
    - виводи сегментів a…g – до відповідних виходів CD4511 через резистори 1 кΩ.
6. Перевірте з'єднання: переконайтесь, що немає короткого замикання, всі елементи підключені до правильних рядків на макетній платі.
7. Запустіть симуляцію. Перемикайте тумблери й фіксуйте, яка цифра відображається.

Лабораторна робота №12

Побудова простої пам’яті на тригерах

Теоретичні відомості

У попередніх роботах ви вже будували логічні вентилі **AND, OR, NOT, XOR** на мікросхемах **74HC** та на транзисторах. Такі схеми належать до **комбінаційної логіки**: їхній вихід визначається **лише поточними** значеннями входів. Змінилися входи — (після малої затримки) змінився і вихід.

Але комп’ютер має не лише “рахувати”, а й **зберігати стан**: пам’ятати попередній результат, записаний біт або те, **яка команда виконується наступною**. Для цього потрібні схеми, у яких вихід залежить не тільки від входів, а й від **попереднього стану**. Такі схеми називаються **послідовними (sequential logic)**.

Послідовна логіка містить елементи, які можуть **утримувати 0 або 1**, доки стан не буде змінено. Базовий елемент такої “пам’яті” — **тригер (flip-flop)**.

Тригер (flip-flop) – це електронний пристрій з двома сталими станами, який може зберігати **один біт інформації** (0 або 1) доти, доки ми його не перепишемо.

- Має **щонайменше два виходи**:
  - Q – основний вихід (стан, який ми вважаємо “збереженим бітом”),
  - ($\overline{Q}$) – інверсія ($\overline{Q}$= NOT Q).
- Має **входи керування**, за допомогою яких ми:
  - записуємо 1 (установка / Set),
  - записуємо 0 (скидання / Reset),
  - або забороняє зміну стану (збереження / Hold).

Логічно тригер поводиться як **однобітна комірка пам’яті**: натиснули “записати 1” – він “пам’ятає” 1;

- натиснули “записати 0” – він “пам’ятає” 0;
- поки нічого не натискаємо – стан зберігається.

У реальних мікросхемах тригери складені з кількох **логічних вентилів**

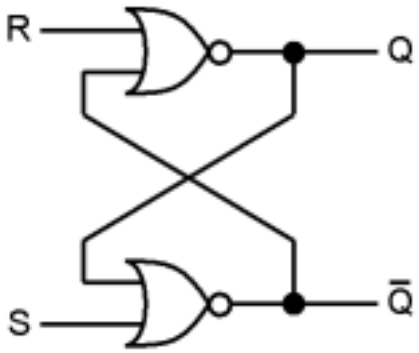
(наприклад, NAND або NOR), які утворюють **петлю зворотного зв'язку**.

Позитивний зворотний зв'язок – як схема "залипає" у стані

Щоб зрозуміти пам'ять, уявімо два інвертори, з'єднані "кільцем": вихід першого йде на вхід другого, а вихід другого – на вхід першого. Якщо один із виходів = 1, то другий стає 0, а потім знову "підтверджує" перший як 1. Виникає **самопідтверджуючий стан**. Система "залипає" або в комбінації: $Q = 1$, $\overline{Q} = 0$, або $Q = 0$, $\overline{Q} = 1$.

Це і є найпростіший принцип **електронної пам'яті**: схема сама підтримує свій стан, навіть якщо вхідний імпульс, який її змінив, вже зник. Тригери – це формалізована, керована версія такого "кільця", доповнена входами, які дозволяють **контрольовано** переключати й "заморожувати" стан.

RS-тригер (Latch) на основі вентилів NOR



Найпростіший різновид тригера – **RS-тригер** (або **RS-защіпка**):

- R - **Reset** (скидання, записати 0),
- S - **Set** (установка, записати 1).

Логічна схема на NOR-вентилях

Уявімо дві логічні схеми **NOR**, з'єднані навхрест:

- вихід першого NOR є входом другого;
- вихід другого NOR є входом першого.

Кожен вентиль NOR має два входи:

- один – зовнішній (S або R),
- другий – це вихід іншого вентиля (зворотний зв’язок).

NOR – це “**НЕ (А АБО В)**”: видає 1 лише тоді, коли **обидва** входи 0, і видає 0, якщо **хоч один** із входів =1.

Позначимо:

- верхній вентиль NOR → вихід Q,
- нижній вентиль NOR → вихід $\overline{Q}$.

Важливі властивості:

- Q та $\overline{Q}$завжди протилежні;
- схема має два стійких стани: $Q = 1, \overline{Q} = 0$ або $Q = 0, \overline{Q} = 1$.

Таблиця істинності RS-тригера (активні рівні “1”) Для RS-тригера на NOR зазвичай вважають, що:

- S=1 → встановити 1 на Q,
- R=1 → скинути Q у 0.

Але важливо враховувати **попередній стан Q(prev)**, тому в таблиці є стовпчик “Q(next)” – стан після дії:

S	R	Q(prev)	Q(next)	Пояснення
0	0	0	0	Збереження стану (0 “тримається”)
0	0	1	1	Збереження стану (1 “тримається”)
1	0	0	1	Установка (Set): записуємо 1
1	0	1	1	Set тримає 1
0	1	0	0	Reset тримає 0
0	1	1	0	Скидання (Reset): записуємо 0
1	1	*будь-який*	**заборонено**	Обидва вентилі одночасно прагнуть видати 0: стан стає нестійким

Режим збереження (Hold): коли S = 0 і R = 0, на входи NOR йдуть тільки

сигнали зворотного зв'язку, і схема просто продовжує "тримати" попередній стан.

Заборонена комбінація: S = 1 і R = 1 одночасно – логічно суперечлива ситуація, обидва виходи прагнуть стати 0, порушується умова $\overline{Q}$= NOT Q.

RS-тригер на NAND-вентилях

Аналогічний тригер можна побудувати на **NAND** (логічне "НЕ-І"). Схема знову містить два вентилі, з'єднані навхрест, але тепер активні рівні будуть **низькі (0)**:

- входи позначають як $\overline{S}$і $\overline{R}$,
- імпульс **0** означає активну дію (установку або скидання).

Коротко:

- $\overline{S}$= 0 → встановити Q = 1,
- $\overline{R}$= 0 → встановити Q = 0,
- $\overline{S}$= 1, $\overline{R}$= 1 → збереження,
- $\overline{S}$= 0, $\overline{R}$= 0 → заборонено.

D-тригер (Data / Delay flip-flop)

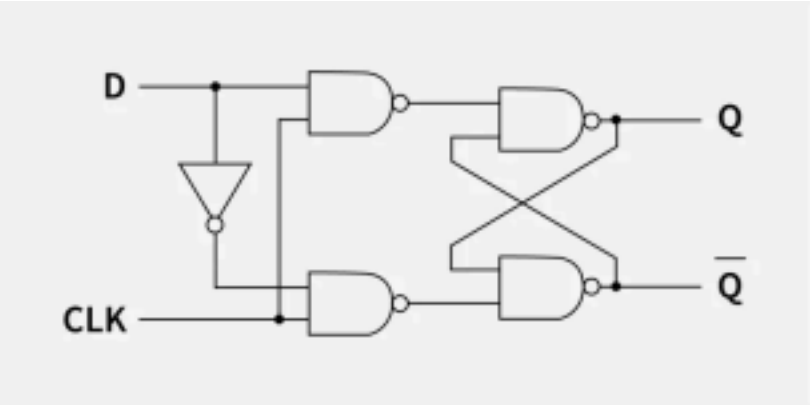
RS-тригер хороший для пояснення, але має недолік: **заборонена комбінація** S = 1, R = 1 і необхідність стежити за тим, щоб S і R не активувались разом.

Тому в практичних схемах широко використовують **D-тригер**: ● має один основний вхід **D** (Data – дані),

- вхід **C** або **CLK** (Clock – тактовий сигнал),
- виходи Q та $\overline{Q}$.

Ідея: Коли приходить тактовий імпульс, тригер **запам'ятовує значення D** і потім утримує його.

D-тригер на основі RS-тригера



Якщо взяти RS-тригер і: формувати сигнал S як D, сигнал R як NOT D, додатково дозволяти їм діяти тільки під час активного сигналу **Enable / Clock**, то ми отримаємо схему, в якій **неможливо** одночасно подати і Set, і Reset: для кожного D або S активний, або R, але не обидва.

Принцип роботи:

- D = 1, приходить тактовий імпульс → тригер записує 1 на Q; ● D = 0, приходить тактовий імпульс → тригер записує 0 на Q; ● між тактовими імпульсами тригер зберігає попередній стан.

1.7.2. Таблиця станів D-тригера

Для спрощеного рівнетактового D-тригера:

D		Q(prev)	Q(next)	Пояснення
0	0	0	0	Такт не активний, стан зберігається
0	0	1	1	
1	0	0	0	
1	0	1	1	
0	1	0	0	При активному такті записується D=0
0	1	1	0	
1	1	0	1	При активному такті записується D=1
1	1	1	1	

У реальних мікросхемах використовують **фронтові (edge-triggered)** тригери: вони реагують тільки на **фронт** такту (перехід 0→1 або 1→0). Це дозволяє будувати синхронні схеми, де всі тригери змінюються одночасно за командою годинника.

Інші види тригерів (короткий огляд)

У цифровій техніці існують також:

- **JK-тригер** – універсальний тригер, який загалом може працювати як RS, D або T залежно від конфігурації входів J, K. Усуває заборонену комбінацію RS-тригера.
- **T-тригер** (Toggle) – "перемикач": якщо вхід T=1 і приходить такт, вихід Q змінюється 0↔1. Використовується як основа для **лічильників**.
- **Master-Slave (майстер-слейв)** – архітектура з двох послідовних тригерів, щоб уникнути "проскакування" сигналів під час довгого тактового імпульсу.

Тригери і пам'ять комп'ютера

Тригери – це "цеглинки", з яких будують:

- **регістри** – групи тригерів, які зберігають байти, слова, адреси; ● **лічильники** – тригери, з'єднані так, щоб рахувати імпульси; ● **оперативну пам'ять** (в старіших технологіях – статичну SRAM); ● **внутрішні стани процесора** – наприклад, прапорці (carry, zero, overflow).

Один тригер = 1 біт:

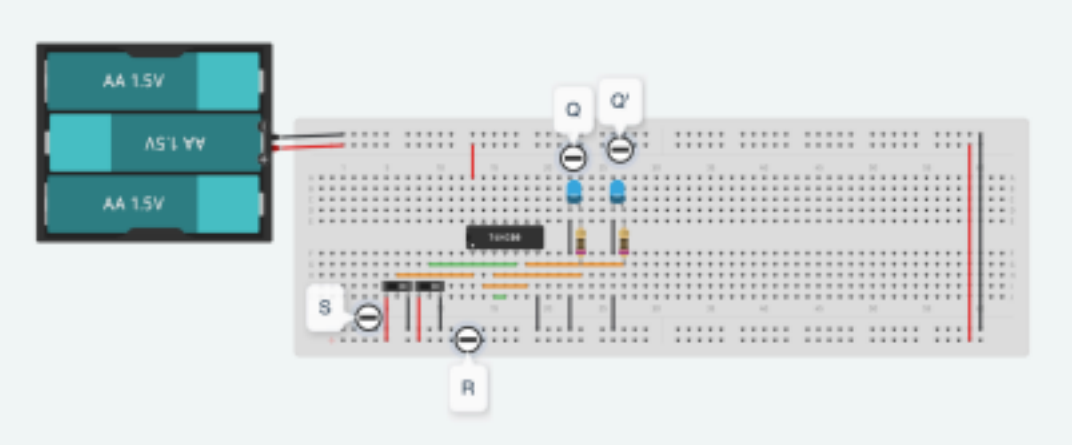
- 8 тригерів → 1 байт.
- 32 тригери → 32-бітний регістр.

Мільйони тригерів → цілі масиви пам'яті й регістри всередині процесора.

Завдання для виконання роботи

Завдання №1: побудувати RS-тригер на базі мікросхеми 74HC00,

використавши два NAND-елементи з чотирьох на мікросхемі, та перевірити роботу виходів Q і Q′.



Необхідні компоненти:

1. Джерело живлення (Power Supply) 5 В.
2. Макетну плату (Breadboard).
3. Мікросхему **74HC00** (4 елементи NAND).
4. 2 кнопки (Pushbutton) – для входів **S** і **R**.
5. 2 світлодіоди (LED) – для індикації виходів **Q** та **Q'**. 6. 2 резистори по 220–330 Ом – обмеження струму світлодіодів. 7. За потреби – 2 резистори 10 кОм (підтягування входів до «1», якщо не використовуєш внутрішні підтяги).
8. З’єднувальні дроти.

Лабораторна робота №13

Лічильники: від ручних рахівниць до електронних схем

Теоретичні відомості

Числа, біти та модуль лічильника

Лічильник завжди працює з **натуральними числами**: 0, 1, 2, 3, … У механічному лічильнику ми бачимо десяткові цифри, у електронному - **двійковий код**, тобто комбінацію нулів і одиниць.

- **Біт** - найменша одиниця інформації (0 або 1).

  Тетрада (нібл) - 4 біти; може кодувати числа від 0 до 15.
- **Байт** - 8 біт.
- У цій роботі ми використовуємо **4-бітний лічильник**. Це означає, що:
- максимально він може представити число $1111_2 = 15_{10}$;
- після 15 він «обнуляється» і починає рахунок спочатку: 0, 1, 2, …, 15, 0, 1

Таку поведінку описують терміном **модуль лічильника** (modulus):

Mod-16-лічильник - лічильник, який має 16 різних станів (0…15), після чого повертається до 0. Можна провести аналогію з **годинником**: годинна стрілка - це Mod-12-лічильник (після 12 знову 1).

Комбінаційна і послідовнісна логіка

Раніше ви вже будували схеми на логічних елементах AND, OR, NOT, XOR.

Такі схеми називають **комбінаційними**:

- вихід у кожний момент часу визначається тільки **поточними** значеннями на входах;
- немає елемента «пам’яті» - якщо вхід змінити, вихід одразу зміниться теж;
- приклади: суматор, декодер, мультиплексор.

Щоб реалізувати лічильник, **пам’ять необхідна**:

- треба знати, скільки імпульсів уже було (попередній стан);
- новий стан залежить і від того, що було до цього.

Схеми, в яких вихід залежить від **попереднього стану**, називають **послідовнісною логікою**. До неї належать:

- тригери;
- регістри;
- лічильники;
- Пам’ять.

Отже, лічильник - це типовий представник послідовнісної логіки.

Тригер як елементарна комірка пам’яті

Тригер - це найпростіший електронний пристрій, який може **зберігати один біт** інформації. Він має два стабільні стани:

- $Q = 0, \overline{Q} = 1$;
- $Q = 1, \overline{Q} = 0$.

Після того як тригер встановлено в певний стан, він **утримує** його, поки не надійде нова команда. У лабораторній роботі №12 ви будували RS-тригер на вентилях NAND і знайомилися з D-тригером.

RS (Reset-Set)-тригер має два основні входи:

- **S (Set)** - «встановити 1»;
- **R (Reset)** - «скинути в 0».

Якщо:

- S=1, R=0 - тригер встановлюється в 1;
- S=0, R=1 - тригер переходить у 0;
- S=0, R=0 - стан зберігається;
- S=1, R=1 - заборонена комбінація (виходи втрачають однозначність).

RS-тригер показує принцип збереження стану, але має незручність - заборонену комбінацію.

D-тригер

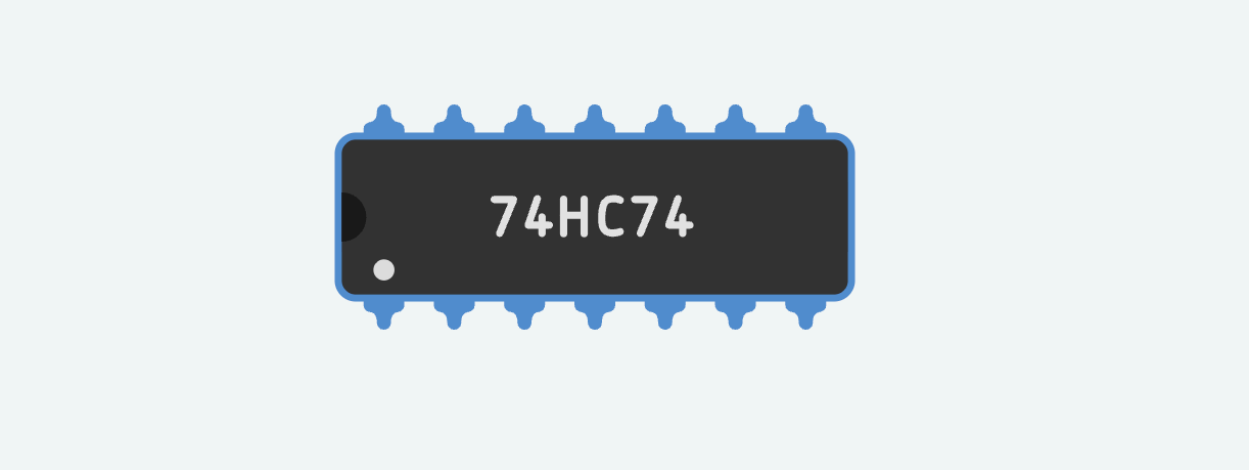
У практиці значно частіше використовують **D-тригер** (Data / Delay):

- має один основний вхід **D (дані)**;
- тактовий вхід **CLK (Clock)**;
- виходи **Q** та $\overline{\mathbf{Q}}$.

У момент активного тактового сигналу (перехід CLK з 0 у 1) тригер запам’ятовує значення D і потім утримує його до наступного імпульсу. Це дуже зручно для синхронних схем: усі тригери змінюються одночасно - за командою.

Мікросхема 74HC74 - двійний D-тригер

Для реалізації лічильника в нашій лабораторній ми використовуємо **інтегральну мікросхему 74HC74**, яка містить **два незалежні D-тригери**.



З точки зору логіки (на схемах у підручниках) кожен D-тригер має такі входи та виходи:

- **VCC** - вивід живлення (+5 В);
- **GND** - загальний провід (0 В);
- для кожного тригера:
  - **D** - вхід даних (Data);
  - **CLK** - тактовий вхід (Clock);
  - **PRE** - асинхронне встановлення (Preset, примусово Q = 1);
  - **CLR** - асинхронний скидання (Clear, примусово Q = 0);
  - **Q, $\overline{Q}$** - прямий та інверсний виходи.

Усередині мікросхеми ці елементи побудовані на десятках транзисторів, але для нас 74HC74 - це зручний «чорний ящик» з 14 виводами, який уміє зберігати й перемикати біти.

Як ці виводи називаються у Tinkercad

У **Tinkercad** той самий 74HC74 показаний не буквами D/CLK/PRE/CLR, а більш «людськими» підписами:

- **Power** - це VCC (живлення +5 В);
- **Ground** - це GND (спільний мінус);

для **першого тригера**:

- **Input 1** - вхід даних D_1;
- **Clock 1** - тактовий вхід CLK_1;
- **Set 1** - асинхронний вхід встановлення PRE_1;
- **Reset 1** - асинхронний вхід скидання CLR_1;
- **Output 1** - прямий вихід Q_1;
- **Inverted Output 1** - інверсний вихід $\overline{Q}_1$;

для **другого тригера**:

- **Input 2** - D_2;
- **Clock 2** - CLK_2;
- **Set 2** - PRE_2;
- **Reset 2** - CLR_2;
- **Output 2** - Q_2;
- **Inverted Output 2** - $\overline{Q}_2$.

У нашому лічильнику використано **дві мікросхеми 74HC74**, отже всього маємо чотири тригери та чотири біти:

- U1: Output 1 → Q_0, Output 2 → Q_1;
- U2: Output 1 → Q_2, Output 2 → Q_3.

Що робить кожен тип виводів (на інтуїтивному рівні)

1. **Power / Ground**
    - Power подаємо на **+5 В** - без цього мікросхема «мертва».
    - Ground з’єднуємо із загальною шиною **0 В**.

      Усі внутрішні транзистори працюють лише тоді, коли живлення підключене.

2. **Output / Inverted Output (Q / $\overline{Q}$)**
    - **Output n** - це той біт, який **ми використовуємо в лічильнику та виводимо на світлодіод**. Якщо Output 1 = 1 - відповідний LED світиться; якщо 0 - гасне.
    - **Inverted Output n** - завжди протилежний: якщо Q = 1, то $\overline{Q}$ = 0 і навпаки.

      У нашій схемі інверсний вихід ми **не показуємо на LED**, а використовуємо всередині, як «обернене значення» для реалізації Т-тригера (див. нижче).
3. **Input (D-вхід)**
    - На **Input n** подаються дані, які тригер має запам’ятати в момент тактового фронту.
    - За правилом D-тригера:

      при фронті Clock тригер копіює Input → Output.
    - У нашому лічильнику ми **не подаємо сюди окремий сигнал**, а просто з’єднуємо:
        - **Input n з Inverted Output n**.

          Таким чином тригер при кожному такті бачить на вході «протилежне самому собі» і змушений **перемикати стан 0↔1**.
4. **Clock (тактовий вхід)**
    - **Clock n** - це «команда на зміну».
    - Коли на цьому вході відбувається фронт (0→1), тригер дивиться на Input і оновлює Output.
    - У нашій схемі:
        - **Clock 1** (перший тригер U1) отримує імпульси від **кнопки** - кожне натискання дає один такт;
        - **Clock 2** U1 під’єднано до Output 1 (Q_0);
        - **Clock 1** U2 - до Output 2 (Q_1);

- **Clock 2** U2 - до Output 1 тієї ж мікросхеми (Q_2).

  Отже кожний старший тригер «дивиться» на молодший і перемикається рідше, утворюючи двійковий лічильник.

5. **Set / Reset (асинхронні входи)**
   - **Set n** - примусово встановлює Output n = 1 (обминаючи Clock);
   - **Reset n** - примусово встановлює Output n = 0 (обминаючи Clock).

     Це дуже «сильні» входи: якщо вони активні, тригер не чекає такту, а одразу стрибає в потрібний стан.
   - У нашому лічильнику ми **не хочемо** користуватися ними, щоб схема не плутала студентів, тому:
     - **Set 1, Set 2, Reset 1, Reset 2** ми просто підключаємо до Power (+5 В) - у такому стані вони не впливають на роботу (неактивні).

Перетворення D-тригера на T-тригер (однорозрядний лічильник)

Лічильник має **міняти** свій стан при кожному імпульсі. D-тригер сам по собі лише **копіює D** на Q при приході такту. Але якщо правильно під'єднати виходи, він перетворюється на **T-тригер** (Toggle).

Схема ідеї:

- вхід **D** з'єднати з інверсним виходом $\overline{\mathbf{Q}}$ (тобто D = NOT(Q));
- тактовий вхід **CLK** підключити до джерела імпульсів (кнопка / генератор);
- входи **PRE** і **CLR** тримати в неактивному стані (логічна 1).

Розглянемо таблицю переходів:

Попередній стан Q(prev)	**Значення $\overline{\mathbf{Q}}$ = NOT Q(prev)**	**D (подано на вхід)**	**Новий стан Q(next)**
0	1	1	1
1	0	0	0

Отже, при кожному фронті CLK:

- якщо Q було 0 → стане 1;
- якщо Q було 1 → стане 0.

Такий тригер називають **Т-тригером (Toggle)**. Це **лічильник по модулю 2** - однорозрядний двійковий лічильник.

Побудова багаторозрядних лічильників

Щоб рахувати не лише 0/1, а й більше значень, кілька Т-тригерів з'єднують **каскадно**.

Пульсаційний (асинхронний) лічильник

Найпростіша схема - **пульсаційний (ripple) двійковий лічильник**:

1. Перший тригер (Q_0) отримує тактовий сигнал безпосередньо (від кнопки чи генератора).
2. Другий тригер (Q_1) отримує тактовий сигнал від виходу Q_0.
3. Третій (Q_2) - від Q_1.
4. Четвертий (Q_3) - від Q_2.

Тоді:

- Q_0 змінюється при **кожному** імпульсі;
- Q_1 - при кожному **другому** (коли Q_0 переходить із 1 у 0);
- Q_2 - при кожному **четвертому**;
- Q_3 - при кожному **восьмому**.

У результаті на чотирьох виходах $Q_3 \ldots Q_0$ формується **двійковий код номера імпульсу** (за модулем 16).

Таблиця переходів 4-бітного лічильника

№ імпульсу	Q_3	Q_2	Q_1	Q_0	Десяткове значення
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3

4	0	1	0	0	4
5	0	1	0	1	5
6	0	1	1	0	6
7	0	1	1	1	7
8	1	0	0	0	8
9	1	0	0	1	9
10	1	0	1	0	10
11	1	0	1	1	11
12	1	1	0	0	12
13	1	1	0	1	13
14	1	1	1	0	14
15	1	1	1	1	15
16	0	0	0	0	0 (новий цикл)

Джерело тактового сигналу: кнопка й генератор

Лічильнику потрібен **ряд імпульсів** - як серце, що задає ритм.

1. **Кнопка (ручний режим).** Найпростіший варіант - кнопка з резистором підтягування/спуску. Коли студент натискає кнопку, на вході CLK_0 коротко з’являється 1, і лічильник робить один крок. Це дозволяє спостерігати зміну станів повільно, «у ручному режимі».
2. **Генератор (автоматичний режим).**

   У реальних пристроях кнопка замінюється на генератор прямокутних імпульсів (RC-генератор, таймер 555, кварцовий осцилятор). Лічильник тоді працює постійно з певною частотою.
3. **Проблема «дребезгу контактів».**

   Реальна кнопка часто дає не один, а кілька дуже коротких імпульсів при одному натисканні (дребезг). У Tinkercad це здебільшого не моделюється, але в реальній схемі довелося б додавати фільтри або тригер Шмітта.

Зв'язок теорії з практичною схемою в Tinkercad. У практичній частині лабораторної роботи ви зберете в Tinkercad **4-бітний двійковий лічильник** за схемою «Copy of 4-bit Counter with D Flip-Flops».

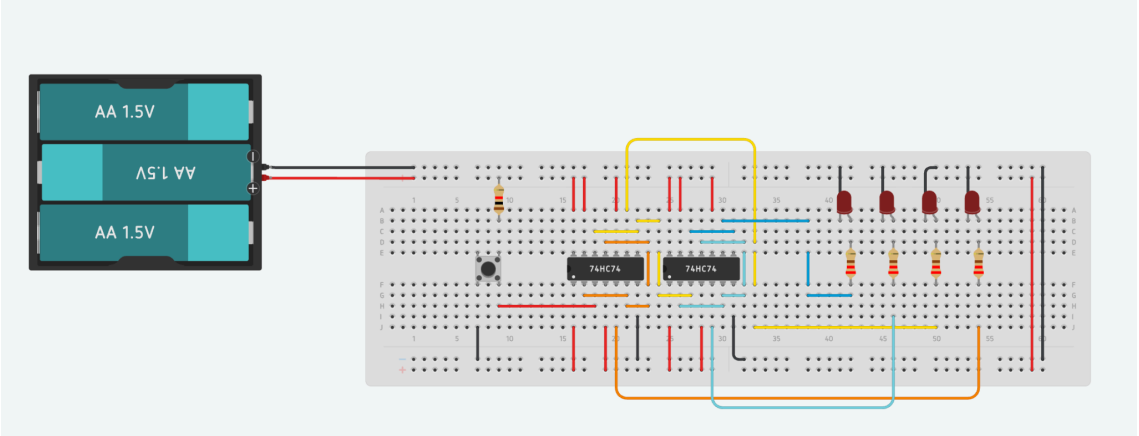
Коротко:

1. **Джерело живлення** (батарейний блок 3×AA ≈ 4,5 В) підключається до шин +Vcc і GND макетної плати.
2. На платі розташовано дві мікросхеми **74HC74** (U1 та U2), кожна - два D-тригери.
3. **Входи PRE і CLR** у всіх тригерів під'єднані до логічної 1, щоб тригери працювали в нормальному режимі без асинхронних установок/скидань.
4. Для кожного тригера **D з'єднаний з $\bar{Q}$**, що переводить його в режим Т-тригера (лічильник по модулю 2).
5. **CLK першого тригера (Q_0)** отримує сигнал від кнопки S1 через резистор (формування імпульсу).

   CLK наступних тригерів підключені до виходів попередніх (Q_0 → CLK другого; Q_1 → CLK третього; Q_2 → CLK четвертого).
6. Виходи **Q_0-Q_3** через резистори підключені до світлодіодів, які відображають двійковий код.
7. Таким чином, побудована у Tinkercad схема є **прямим фізичним втіленням** теоретичної моделі 4-бітного пульсаційного лічильника.

Завдання для виконання роботи

Завдання №1: Побудова 4-бітного лічильника з D-тригерами.



Необхідні компоненти

- Макетна плата (breadboard).
- Батарейний блок 3×AA (≈4,5 В) або інше джерело 5 В.
- 2 інтегральні мікросхеми **74HC74** (кожна містить 2 D-тригери).
- 1 кнопка (pushbutton) - джерело тактових імпульсів.
- 4 світлодіоди (LED) для індикації бітів Q_0-Q_3.
- 4 резистори 220-330 Ом (послідовно з кожним LED).
- 1 резистор 1 кОм (формування сигналу кнопки, як у схемі Tinkercad).
- Провідники для монтажу.

Підготовка макетної плати

1. Створіть нову схему у Tinkercad → Circuits → Create new Circuit → Додайте роботу в Class Tinkercad.
2. Розмістіть breadboard і батарейний блок 3×AA (або інше джерело 5 В).
3. Підключіть:
   - «+» батарейного блоку → до червоної шини живлення breadboard (це буде +Vcc ≈ 4,5-5 В);
   - «−» батарейного блоку → до синьої шини (це GND).
4. Якщо на платі дві пари шин (верх/низ) - об’єднайте їх перемичками, щоб усі червоні шини були під +Vcc, а всі сині - під GND.

Встановлення мікросхем 74HC74

1. Розмістіть дві IC 74HC74 так, щоб кожна «перекривала» центральний розрив breadboard (класична DIP-орієнтація).
2. Для кожної 74HC74:
    - вивід Power (VCC) підключіть до +Vcc (червона шина);
    - вивід Ground (GND) - до синьої шини (0 В).

Налаштування тригерів у режим Т-тригера

Перетворення кожного D-тригер мікросхеми на Т-тригер, щоб він працював як однобітний лічильник mod-2.

Для всіх чотирьох тригерів (2 в U1 і 2 в U2):

1. З'єднайте вхід даних із інверсним виходом:
    - у Tinkercad це:
        - Input 1 з'єднати з Inverted Output 1;
        - Input 2 з'єднати з Inverted Output 2 (на обох мікросхемах).
    - Це реалізує правило: $D = \overline{Q}$, тому при кожному такті тригер перемикає стан (0→1, 1→0).
2. Встановіть Set і Reset у неактивний стан:
    - виводи Set 1, Reset 1, Set 2, Reset 2 для кожної IC підключіть до +Vcc (Power).
    - Так асинхронні встановлення/скидання відключені, тригер реагує тільки на тактовий сигнал.

Після цього кожен тригер 74HC74 у схемі працює як Т-тригер.

Організація тактового сигналу від кнопки

1. Розмістіть кнопку (pushbutton) на платі так, щоб вона теж перекривала центральний розрив.
2. Один контакт кнопки під'єднайте до +Vcc.
3. Другий контакт кнопки підключіть до вузла, який буде лінією такту для першого тригера (Clock 1 U1).
4. Між цією лінією та GND підключіть резистор 1 кОм (pull-down):
    - один кінець резистора → до вузла Clock 1;

- другий → до GND.

  Це забезпечує, що при відпусканні кнопки на вході Clock 1 гарантовано логічний «0».

5. (Опційно) Паралельно до цієї ж лінії можна підключити LED з резистором 220 Ом, щоб візуально бачити імпульси такту (як у схемі Tinkercad з LED D1).

Кожне натискання кнопки формує один імпульс, який «просуває» лічильник на один крок.

Каскадне з’єднання тригерів у 4-бітний лічильник

Тепер ви об’єднаєте чотири T-тригери в один 4-бітний двійковий лічильник.

1. Перший тригер U1 (Q_0) - молодший біт:
   - Clock 1 (U1) уже підключений до кнопки - це основний тактовий вхід лічильника.
   - Output 1 (U1) буде виходом Q_0.
2. Другий тригер U1 (Q_1):
   - Clock 2 (U1) підключіть до Output 1 (U1).
   - Тепер Q_1 перемикається кожного разу, коли Q_0 проходить повний цикл 0→1→0, тобто вдвічі рідше.
3. Перший тригер U2 (Q_2):
   - Clock 1 (U2) підключіть до Output 2 (U1).
   - Вихід Output 1 (U2) буде бітом Q_2.
4. Другий тригер U2 (Q_3) - старший біт:
   - Clock 2 (U2) підключіть до Output 1 (U2).
   - Вихід Output 2 (U2) буде бітом Q_3.

У результаті отримаємо пульсаційний (ripple) лічильник mod-16:

- Q_0 змінюється при кожному імпульсі;
- Q_1 - при кожному другому;
- Q_2 - при кожному четвертому;
- Q_3 - при кожному восьмому.

Світлодіодна індикація бітів Q_0-Q_3

1. Розмістіть 4 LED на правій частині плати - це індикатори розрядів Q_0-Q_3.
2. Для кожного світлодіода:
    - анод підключіть до відповідного виходу Output n через резистор 220-330 Ом;
    - катод підключіть до GND.
3. Рекомендується розташувати їх у порядку:
    - зліва направо: Q_3, Q_2, Q_1, Q_0 (MSB → LSB)

      або навпаки - але обов’язково зазначити це у звіті.

У типовій схемі Tinkercad LED D4, D3, D2, D1 підключені саме так:

D4 ← Q_3, D3 ← Q_2, D2 ← Q_1, D1 ← Q_0.

Перевірка та дослідження роботи лічильника

1. **П**еревірте з’єднання:
    - переконайтеся, що немає коротких замикань між +Vcc та GND;
    - усі входи Set/Reset підключені до +Vcc;
    - всі Input з’єднані з відповідними Inverted Output.
2. Запустіть симуляцію в Tinkercad (кнопка Start Simulation).
3. Повільно натискайте кнопку 16 разів і після кожного натискання фіксуйте стан LED (Q_3…Q_0).
4. Заповніть таблицю:

№ імпульсу	**Q_3**	**Q_2**	**Q_1**	**Q_0**	**Десяткове число**

5. Порівняйте отриману послідовність із теоретичною таблицею двійкового лічильника від 0 до 15.
6. Зробіть 5-8 скріншоти схеми з різними станами лічильника та додайте до звіту.

Лабораторна робота №14

Моделювання мікропроцесора: пам’ять, суматор і лічильник команд

Теоретичні відомості

Мікропроцесор: Архітектурний фундамент

Щоб зрозуміти, що ми будемо моделювати, необхідно розібратися, що таке мікропроцесор (Central Processing Unit — CPU).

Уявіть великий завод.

- Мікропроцесор — це головний інженер, який стоїть біля конвеєра. Він не тримає деталі (дані) в кишенях постійно, він лише обробляє їх згідно з інструкцією.
- Його завдання: взяти деталь -> зробити з нею дію (зігнути, пофарбувати) -> покласти назад або передати далі.

На технічному рівні мікропроцесор — це надскладна логічна схема, виготовлена на кристалі напівпровідника (кремнію), яка складається з транзисторів (мільярдів мікроскопічних перемикачів).

Основні вузли мікропроцесора:

1. Арифметико-логічний пристрій (АЛП / ALU): Це «калькулятор» всередині процесора. Саме тут фізично відбувається додавання ($A+B$), віднімання або логічні операції (AND, OR, NOT).
2. Регістри (Registers): Це надшвидка пам'ять безпосередньо всередині процесора. Це ніби "руки" інженера. Перш ніж додати два числа, процесор мусить взяти їх із зовнішньої пам'яті (RAM) і покласти у свої регістри.
3. Пристрій керування (Control Unit): Це блок, який розшифровує команди програми та смикає за "ниточки" (керуючі сигнали), вказуючи АЛП та регістрам, що робити в даний момент.

Функціональні елементи в нашій моделі

У цій лабораторній роботі ми не створюємо процесор з нуля (з транзисторів), ми моделюємо його роботу. Розберемо, хто і за що відповідає в нашій лабораторній установці:

Виконання функції Пам'яті (Регістрів)?

У реальному процесорі пам'ять — це тригери (flip-flops).

У нашій лабораторній роботі функцію Регістра Виводу виконують Світлодіоди (LEDs).

- Якщо світлодіод світиться — у цьому біті регістра записана логічна «1».
- Якщо погашений — логічний «0».
- Сукупність 6 світлодіодів утворює 6-бітний регістр даних.

Виконання функції Суматора (АЛП)?

У реальному «залізі» це складна схема з логічних вентилів XOR та AND.

У нашій роботі функцію АЛП виконує Програмний код всередині мікроконтролера. Коли ми пишемо команду перемикання світлодіодів, внутрішній процесор Arduino проводить обчислення і змінює напругу на портах. Тобто ми емулюємо апаратну логіку програмним шляхом.

Виконання функції Лічильника команд (Program Counter)?

Лічильник команд — це вказівник, який каже процесору, який рядок коду виконувати наступним.

У нашій роботі це реалізується через структуру функції loop(). Програма виконується рядок за рядком зверху вниз. Перехід від команди digitalWrite(led1, ...) до digitalWrite(led2, ...) — це і є робота лічильника команд, який просуває нас по алгоритму.

Перехід до практики: Зв'язок Arduino та Мікропроцесорів

Тут криється фундаментальна різниця понять:

1. Мікропроцесор (CPU): Тільки обчислювальне ядро (як двигун машини). Йому потрібні зовнішні чіпи пам'яті, порти, генератори частоти, щоб працювати. (Приклад: Intel Core i5).
2. Мікроконтролер (MCU): Це комп'ютер на одному кристалі. В одному корпусі запаяні: Мікропроцесор + Оперативна пам'ять + Флеш-пам'ять + Порти вводу-виводу. (Приклад: ATmega328P).
3. Arduino Uno: Це плата розробника (Development Board). Це зручна друкована плата, на якій розпаяний мікроконтролер, стабілізатор живлення

та USB-інтерфейс для прошивки.

Використання Arduino в цій лабораторній. Серцем плати Arduino Uno є чіп ATmega328P. Всередині нього працює класичний 8-бітний мікропроцесор архітектури AVR. Використовуючи команди Arduino (наприклад, digitalWrite), ми безпосередньо звертаємося до регістрів цього мікропроцесора. Ми змушуємо транзистори всередині чіпа відкриватися і подавати струм на ніжки плати. Це ідеальний стенд, щоб візуалізувати роботу шини даних, яку зазвичай неможливо побачити оком.

Основи роботи з Arduino

Програма (скетч) для Arduino пишеться мовою C++ і завжди складається з двох обов'язкових частин:

1. **void setup():** Ця функція запускається один раз при увімкненні живлення.
    - *Аналогія:* Підготовка робочого місця перед зміною. Тут ми кажемо мікропроцесору: "Налаштуй ніжки 8-13 як виходи (OUTPUT), щоб ми могли керувати ними".
2. **void loop():** Ця функція запускається після setup і виконується нескінченно по колу.
    - *Аналогія:* Робочий процес конвеєра. Як тільки остання команда виконана, процесор миттєво повертається до першої.

Ключові команди, які ми використаємо:

- pinMode(pin, OUTPUT) — перемикає порт у режим джерела струму.
- digitalWrite(pin, HIGH) — подає на порт +5 Вольт (Логічна "1"). Світлодіод світиться.
- digitalWrite(pin, LOW) — подає на порт 0 Вольт (GND, Логічний "0"). Світлодіод гасне.

- delay(ms) — зупиняє виконання програми на вказану кількість мілісекунд. Це емуляція тактової частоти (якби процесор працював дуже повільно, щоб ми могли помітити зміни).

Завдання для виконання роботи

Завдання №1: Створити модель 6-бітного регістра зсуву, використовуючи платформу Arduino Uno.

Складання віртуальної схеми (Hardware)

1. Увійдіть у Tinkercad Circuits.
2. Виберіть компоненти:
    - Arduino Uno R3 — 1 шт.
    - Breadboard Small (Макетна плата) — 1 шт.
    - LED (Світлодіод) — 6 шт. (Бажано різних кольорів для наочності: наприклад, пари Зелений-Жовтий-Червоний).
    - Resistor (Резистор) — 6 шт.
3. Налаштування резисторів: Клацніть на кожен резистор і встановіть опір **110 ом**
    - Важливо: Без резистора струм буде занадто великим, і порт мікропроцесора або світлодіод згорять. Резистор обмежує струм.
4. З'єднання:
    - Встановити світлодіоди на макетну плату.
    - Катод (коротка ніжка) кожного діода підключіть до лінії "мінус" (GND). З'єднайте лінію "мінус" плати з піном GND на Arduino.
    - Анод (довга ніжка) через резистор підключіть до цифрових пінів Arduino у такій послідовності:
        - LED 1 -> Pin 13
        - LED 2 -> Pin 12
        - LED 3 -> Pin 11
        - LED 4 -> Pin 10
        - LED 5 -> Pin 9

- LED 6 -> Pin 8

Програмна реалізація (Software)

Відкрийте панель Code, виберіть режим Text і введіть наступний код. Цей код імітує роботу лічильника та зсув даних.

```
C++
// 1. Оголошення змінних (Mapping).
// Ми даємо імена портам мікропроцесора, щоб зручніше до них
звертатися.
int led1 = 13;
int led2 = 12;
int led3 = 11;
int led4 = 10;
int led5 = 9;
int led6 = 8;
// 2. Блок налаштування (Setup). Виконується 1 раз.
void setup()
{
// Конфігурація регістрів порту. Ми кажемо процесору відкрити ці порти
на вихід.
pinMode(led1, OUTPUT);
pinMode(led2, OUTPUT);
pinMode(led3, OUTPUT);
pinMode(led4, OUTPUT);
pinMode(led5, OUTPUT);
pinMode(led6, OUTPUT);
}
// 3. Основний цикл (Loop). Емуляція роботи процесора.
void loop() {
// Частина А: Швидка послідовна обробка
// Імітація проходження сигналу по шині даних
digitalWrite(led1, HIGH); // Записати "1" у 1-й розряд
delay(25); // Чекати тактовий імпульс
digitalWrite(led6, LOW); // Очистити останній розряд
delay(25);
digitalWrite(led2, HIGH); // Зсув даних: "1" переходить у 2-й розряд
delay(25);
digitalWrite(led1, LOW); // Очистити попередній
delay(25);
digitalWrite(led3, HIGH);
delay(25);
digitalWrite(led2, LOW);
delay(25);digitalWrite(led5, HIGH);
delay(25);
digitalWrite(led4, LOW);
delay(25);
digitalWrite(led6, HIGH);
delay(25);
digitalWrite(led5, LOW);
delay(25);
digitalWrite(led6, HIGH); // Утримання
delay(25);
// Частина Б: Повільна послідовність (Step-by-step debug mode)
// Тут затримка 100 мс дозволяє краще роздивитися перемикання
digitalWrite(led5, HIGH);
delay(100);
digitalWrite(led6, LOW);
delay(100);
digitalWrite(led4, HIGH);
delay(100);
```

```
digitalWrite(led5, LOW);
delay(100);
digitalWrite(led3, HIGH);
delay(100);
digitalWrite(led4, LOW);
delay(100);
digitalWrite(led2, HIGH);
delay(100);
digitalWrite(led3, LOW);
delay(100);
digitalWrite(led2, LOW);
delay(100);
}
```

Лабораторна робота №15

Тема: Будова сучасного персонального комп’ютера та ноутбука

Теоретичні відомості:

Базова архітектура комп’ютера.

Архітектура комп’ютера - це сукупність принципів і правил організації апаратних та програмних засобів, які визначають:

- з яких компонентів складається комп’ютер;
- як ці компоненти взаємодіють між собою;
- яким чином виконуються команди користувача і програми.

Класична архітектура комп’ютера (модель фон Неймана) складається з кількох ключових компонентів, які спільно забезпечують роботу системи (рис. 1).

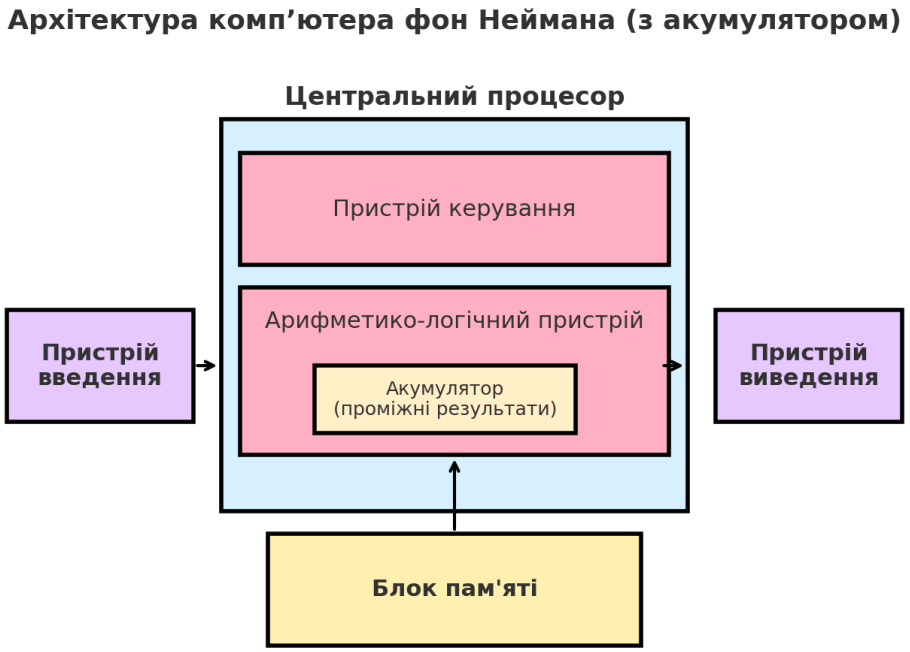


Рис. 1 - Архітектура комп’ютера фон Неймана

Основними блоками є:

Блок пам’яті

- Тут зберігаються *програми і дані*.
- Уявіть «робочий стіл»: сюди ми кладемо і підручник (програма), і зошит (дані).
- Процесор завжди звертається сюди по інформацію.

Пристрій керування

Це «диригент» комп'ютера.

- Він зчитує з пам'яті команду і віддає наказ арифметико-логічному пристрою, що робити.
- Приклад: натиснули «Enter» → CU подає сигнал «перейти на новий рядок».

Арифметико-логічний пристрій

Це «калькулятор комп'ютера».

Виконує:

- арифметичні операції (додавання, віднімання, множення, ділення);
- логічні операції (порівняння «так/ні»).
- Усередині є **акумулятор** - невелика комірка для проміжних результатів.

Пристрої введення та виведення

Введення: клавіатура, миша, мікрофон.

- *Виведення*: монітор, принтер, динаміки.
- Це наші «очі й руки», через які ми взаємодіємо з машиною.

Шини

- Це «дороги», якими бігають дані.
- Якщо дорога забита - виникає затримка → **«вузьке місце фон Неймана»**.

Приклади реалізації моделі фон Неймана стали:

EDVAC (Electronic Discrete Variable Automatic Computer, 1949, США) (рис. 2.**)**

- Одна з перших машин зі **збереженою програмою**: і програми, і дані лежали в одній пам'яті.
- Виконувала військово-наукові та інженерні розрахунки: балістика, криптоаналіз, математична фізика.
- Показала перевагу універсальності: щоб розв'язувати іншу задачу, достатньо змінити **програму**, а не апаратну схему.

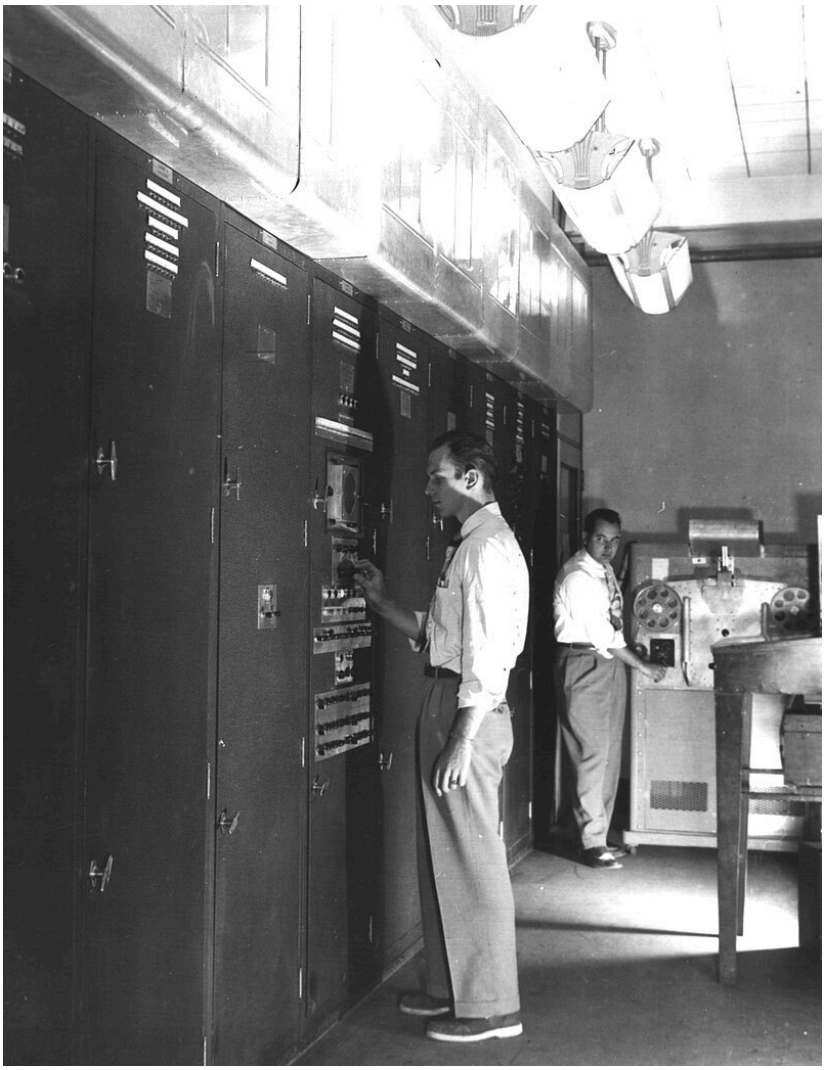


Рис. 2 - Вигляд EDVAC

МЕОМ (Мала електронна обчислювальна машина, 1951, СРСР, Київ) (рис. 3**)**

- Перша діюча ЕОМ континентальної Європи (керівник - Сергій Лебедєв).
- Також реалізувала **фон-нейманівський підхід** із єдиною пам’яттю для програм і даних.
- Застосування: ядерна фізика, авіаційно-ракетні розрахунки, моделювання складних процесів.

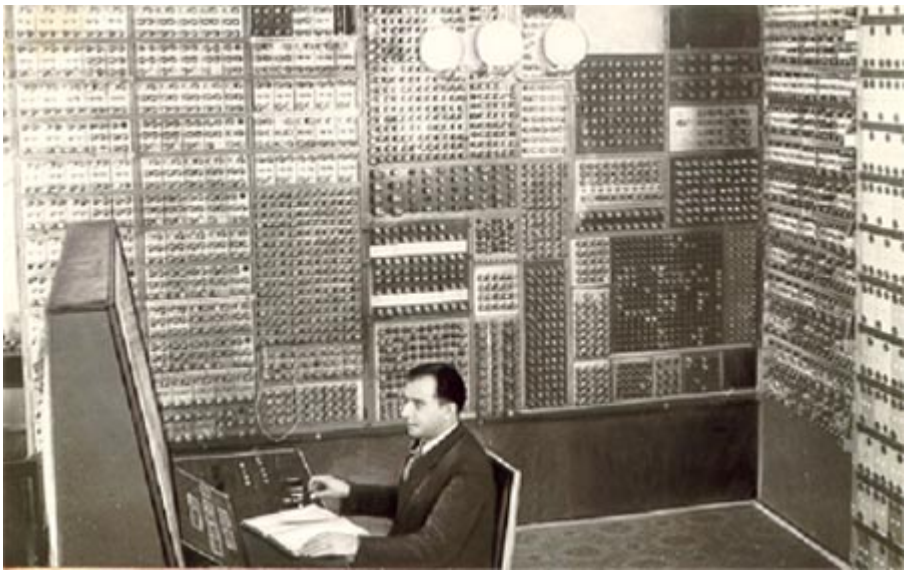


Рис. 3 - Вигляд МЕОМ

Практика експлуатації EDVAC та МЕОМ підтвердила життєздатність моделі фон Неймана, але й виявила її обмеження: усі звернення до пам’яті йдуть спільним каналом, через що процесор часто чекає на дані чи команди. Саме це підводить нас до ключового поняття курсу - «вузького місця» архітектури фон Неймана.

Проблемним місцем архітектури фон Неймана є наявність одного каналу до пам’яті. Через це Арифметико-логічний пристрій іноді простоює. Керуючий пристрій спочатку зчитує команду програми з пам’яті. Потім через той самий канал потрібно дістати дані для виконання цієї команди.

Щоб обійти це обмеження, у сучасних ПК з’явилися:

Кеш-пам’ять (L1, L2, L3)

- Вбудована прямо в процесор.
- Зберігає найчастіше використовувані дані та команди.

Багатоядерні процесори

- Кілька ядер (**Пристрій керування** + **Арифметико-логічний пристрій**) у одному кристалі.
- Кожне ядро може незалежно виконувати свій набір інструкцій.

Використання окремих процесорів (GPU, AI-чіпи)

- Графічний процесор (GPU) бере на себе паралельні завдання.

- AI-акселератори (TPU, NPU) працюють з машинним навчанням.

1.2. Сучасний персональний комп’ютера: CPU, RAM, GPU, накопичувачі, I/O, живлення.

Якщо модель фон Неймана описує загальні принципи роботи комп’ютера, то сучасний персональний комп’ютер є їхньою конкретною реалізацією. У ньому класичні блоки - процесор, пам’ять, введення/виведення та накопичувачі - набули нового вигляду, стали складнішими й продуктивними. Розглянемо детальніше кожен із компонентів.

CPU (Central Processing Unit, центральний процесор) (рис. 4)

- Призначення:
    - «Мозок» комп’ютера, виконує арифметичні та логічні операції, керує іншими пристроями.
- Будова:
    - ALU (арифметико-логічний пристрій) – обчислення.
    - Блок керування (CU) – інтерпретує та виконує інструкції.
    - Кеш-пам’ять (L1, L2, L3) – швидка пам'ять для зберігання часто використовуваних даних.
- Особливості сучасних CPU (2025):
    - Багатоядерність (4–64 ядер).
    - Підтримка багатопоточності (SMT, HyperThreading).
    - Вбудовані NPU (нейронні процесори для AI).
- Взаємодія: CPU напряму працює з RAM і керує обміном даних між усіма компонентами.

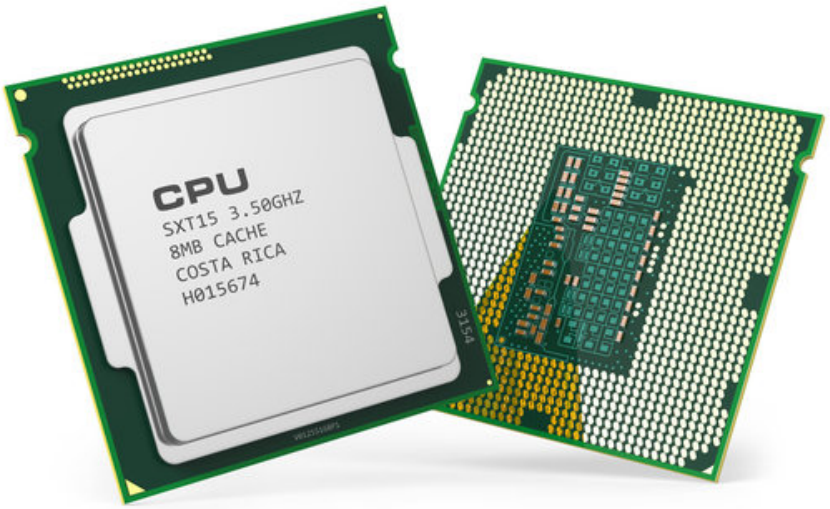


Рис. 4 - Центральний процесор (CPU)

RAM (Random Access Memory, оперативна пам’ять) (рис. 5)

- **Призначення:**
  - Тимчасове зберігання даних і програм, з якими працює CPU.
  - Після вимкнення живлення дані зникають.
- **Характеристики:**
  - Обсяг (8–128 ГБ у ПК, 16–64 ГБ у ноутбуках).
  - Типи: DDR4 (застарілий), DDR5 (основний у 2025), LPDDR5X (у ноутбуках).
  - Швидкість: впливає на пропускну здатність між CPU і RAM.
- **Взаємодія:** CPU завантажує дані з накопичувача у RAM → обробляє → зберігає назад у RAM чи накопичувач.

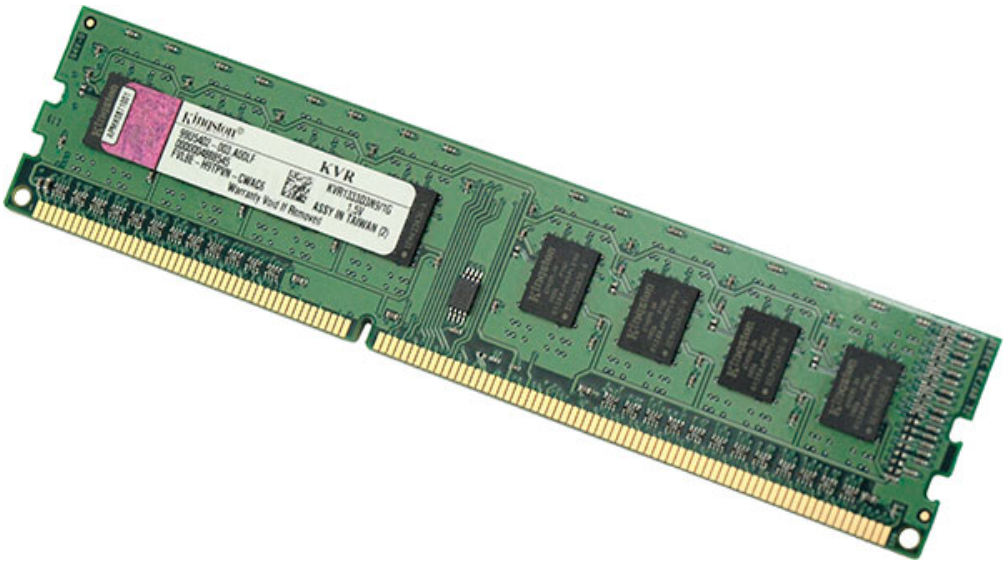


Рис. 5 - Модуль оперативної пам'яті DDR

GPU (Graphics Processing Unit, графічний процесор) (рис. 6)

- **Призначення:**
  - Виконує масово-паралельні обчислення.
  - Відповідає за обробку графіки, відео, 3D-моделей, AI.
- **Види:**
  - **Інтегровані (iGPU):** вбудовані у CPU, підходять для офісних задач.
  - **Дискретні (dGPU):** окремі плати (NVIDIA RTX, AMD Radeon) з власною відеопам'яттю (VRAM).
- **Сучасні GPU (2025):**
  - NVIDIA RTX 50xx (архітектура Blackwell).
  - AMD Radeon RX 8000.
  - Apple M3 GPU (ARM, інтегрований).
- **Взаємодія:** CPU передає задачі GPU → GPU обчислює та віддає результат (зображення, AI-результати).

Рис. 6 - Графічний процесор (GPU)

Накопичувачі (Storage) (рис. 7)

- **Призначення:**
  - Постійне зберігання операційної системи, програм і файлів.
- **Типи:**
  - **HDD (жорсткі диски):** великі обсяги, але повільні. Використовуються для архівів.
  - **SSD NVMe (M.2, PCIe 4.0/5.0):** сучасний стандарт, надвисока швидкість читання/запису.
- **Взаємодія:** CPU через чіпсет звертається до SSD/HDD → дані завантажуються у RAM → опрацьовуються.

Рис. 7 - Накопичувачі даних (HDD та SSD)

I/O (Input/Output – введення/виведення) (рис. 8)

- **Призначення:** забезпечує взаємодію користувача та комп’ютера.
- **Пристрої введення:** клавіатура, миша, мікрофон, камера, сенсори.
- **Пристрої виведення:** монітор, принтер, динаміки, VR-шоломи.
- **Сучасні інтерфейси (2025):**
    - USB 4.0, Thunderbolt 5 (швидкий обмін даними).
    - HDMI 2.1, DisplayPort 2.1 (відео 8K/120Hz).
    - Wi-Fi 7, Bluetooth 5.3.
- **Взаємодія:** дані від користувача через I/O → CPU → обробка → GPU/монітор.

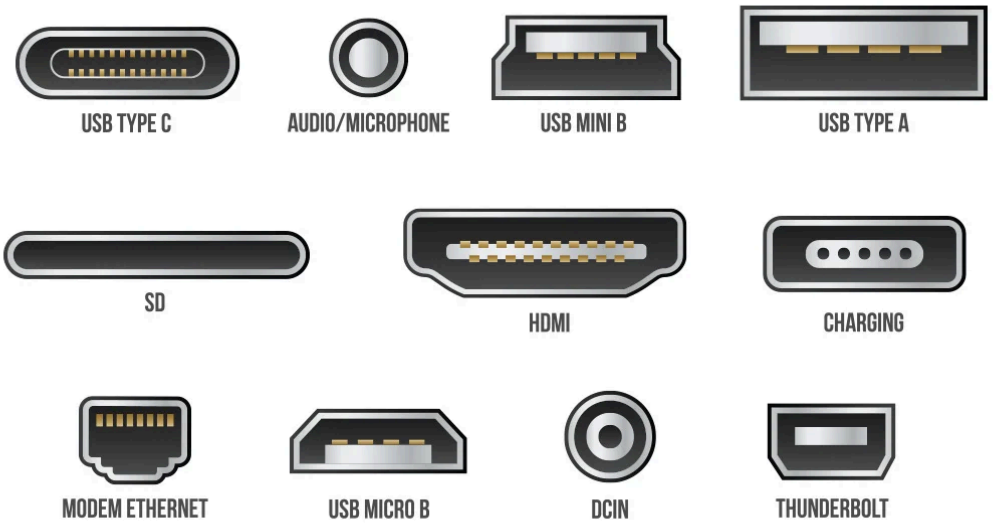


Рис. 8 - Інтерфейси введення та виведення (I/O)

Живлення (Power Supply Unit, PSU) (рис. 9)

- **Призначення:**
    - Забезпечує стабільне живлення всіх компонентів.
    - Перетворює змінний струм (220 В) у постійний (12/5/3.3 В).
- **Сучасні стандарти:**
    - 80 PLUS Gold/Platinum (висока ефективність).

- Потужність: 500–1200 Вт залежно від конфігурації.

- **Взаємодія:** якщо блок живлення слабкий → система працює нестабільно (краші, перезавантаження).

Рис. 9 - Блоки живлення персонального комп'ютера (PSU)

Материнська плата (Motherboard) (рис. 10)

Призначення: Основа комп'ютера, що з'єднує всі компоненти в єдину систему. Забезпечує обмін даними між процесором, пам'яттю, накопичувачами та периферією.

Будова та ключові елементи:

- **Сокет CPU** – роз'єм для встановлення процесора.
- **Слоти RAM** – модулі оперативної пам'яті (DDR4/DDR5).
- **Чипсет (PCH)** – «посередник» між CPU і периферією.
- **Слоти розширення (PCIe 4.0/5.0)** – для GPU, мережевих та звукових карт.
- **Накопичувачі** – роз'єми SATA, M.2 NVMe.
- **BIOS/UEFI** – прошивка для старту системи та налаштування апаратури.
- **Контролери I/O** – USB, аудіо, мережа, Thunderbolt.

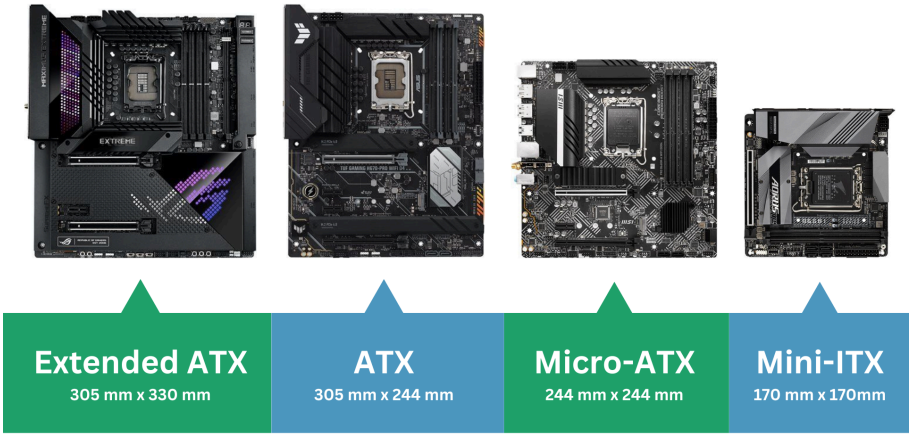


Рис. 10 - Основні види материнських плат

Взаємодія: Материнська плата з'єднує CPU, RAM, GPU, накопичувачі та I/O через системну шину (Front-Side Bus або сучасні внутрішні інтерконекти) та забезпечує живлення і синхронізацію всіх компонентів.

Взаємозалежність компонентів між собою:

- Потужна GPU без відповідного CPU «нудьгує».
- Швидкий CPU без достатньої RAM працює у «вузькому місці».
- SSD розкриває себе тільки з достатнім об'ємом RAM.
- Сильний блок живлення і якісне охолодження - основа стабільності.

У таблиці 1 наведено короткий перелік характеристик ПК, на які слід звертати увагу під час вибору компонентів, а також пояснено їх вплив на реальне використання.

Таблиця 1. Основні характеристики ПК та їх вплив на роботу системи

Характеристика	**На що впливає**	**Пояснення**
CPU (частота, ядра, потоки)	Швидкість обчислень, ігри, робота з програмами	Більше ядер → краще багатозадачність (рендер, відео, AI). Вища частота → швидше виконання інструкцій.
GPU (тип, обсяг VRAM)	Ігри, відеомонтаж, 3D, AI	Визначає якість і плавність графіки. VRAM (8–24 ГБ) важливий для ігор 4K і машинного навчання.

RAM (обсяг, швидкість)	Багатозадачність, швидкість роботи	16 ГБ - мінімум для 2025. DDR5 швидше за DDR4. Якщо мало RAM → система починає “гальмувати”, використовуючи SSD.
SSD (тип: NVMe, Gen 4/5)	Швидкість завантаження ОС, ігор, програм	NVMe у 10–20 разів швидше HDD. Gen 5 у 2025 - топ (до 12 ГБ/с).
Материнська плата (чіпсет, сокет)	Сумісність компонентів, можливість апгрейду	Має визначати, який CPU/RAM підтримується, і чи можна буде оновити систему через кілька років.
PSU (потужність, сертифікація)	Стабільність і довговічність роботи	Потужність повинна перевищувати споживання ПК на 20–30%. Сертифікація 80+ (Gold/Platinum) = ефективність і надійність.
Cooling (тип: повітряне, водяне)	Температура, стабільність, рівень шуму	Якщо охолодження слабке → CPU/GPU гріються і знижують частоти (throttling). Важливо для ігор і рендеру.
Монітор (частота, роздільна здатність)	Якість картинки, комфорт у роботі та іграх	144 Гц і вище для геймерів. 4K і IPS/miniLED - для дизайнерів.
Корпус (форм-фактор, вентиляція)	Охолодження і зручність апгрейду	Погана вентиляція = перегрів. Просторий корпус = легше оновлювати ПК.

Сучасний комп’ютер та його різновиди

Персональний комп’ютер (Desktop PC) - це тип комп’ютерної системи, призначений для стаціонарного використання, що складається з окремих апаратних компонентів і забезпечує високу продуктивність та широкі можливості модернізації. (Рис. 11)

Рис. 11 - Персональний комп'ютер

Характеристика:

- Призначений для стаціонарного використання.
- Забезпечує широкі можливості апаратної модернізації (розширення оперативної пам'яті, встановлення SSD, заміна відеокарти).
- Характеризується високим рівнем продуктивності порівняно з мобільними комп'ютерними системами.

Приклади

- Ігрові конфігурації на базі процесорів Intel Core i9 з дискретною графікою NVIDIA RTX 4090.
- Офісні та навчальні системи на базі Intel Core i5 з інтегрованою графікою.

Сфера застосування:

- Рендеринг та обробка тривимірної графіки (Blender, 3ds Max).
- Розробка програмного забезпечення та робота з великими проєктами (Visual Studio, IntelliJ).
- Обробка графічних і відеоматеріалів (Adobe Photoshop, Premiere Pro).
- Аналіз і обробка великих масивів даних (Python, R, MATLAB).

Ноутбук (Laptop) - це портативна комп'ютерна система, призначена для мобільного використання, яка поєднує в одному корпусі обчислювальні компоненти, дисплей, клавіатуру та акумулятор. (рис. 12)

Рис. 12 - Ноутбук

Характеристика:

- Забезпечує мобільність та автономну роботу без постійного підключення до електромережі.
- Має компактні апаратні компоненти; оперативна пам’ять і графічний процесор часто інтегровані в материнську плату.
- Характеризується обмеженими можливостями апаратної модернізації порівняно зі стаціонарними ПК.

Приклади:

- Професійні ноутбуки: MacBook Pro (Apple M3).
- Ігрові ноутбуки: ASUS ROG Strix.
- Бізнес- та корпоративні моделі: Lenovo ThinkPad.

Сфера застосування:

- Навчальна діяльність: робота з текстовими документами, презентаціями, відеоконференціями.
- Розробка вебсайтів та мобільних застосунків.
- Дистанційне навчання та онлайн-курси (LMS-платформи, Moodle, Coursera).
- Мобільна робота, бізнес-задачі та ігри «на ходу».

Міні-ПК (Mini PC) - це компактна стаціонарна комп'ютерна система малого форм-фактора, призначена для виконання базових і середніх обчислювальних завдань за мінімального енергоспоживання та займаного простору. (рис. 13)

Рис. 13 - Міні-ПК

Характеристика:

- Відрізняється компактними розмірами та малим форм-фактором (Intel NUC, Mac mini).
- Використовує енергоефективні процесори з інтегрованою графікою.
- Обмежений у можливостях апаратної модернізації та не призначений для ресурсомістких ігор і складного 3D-рендерингу.
- Оптимальний для офісних, мультимедійних і повсякденних задач.

Приклади: Intel NUC 13, Mac mini M2, ASUS PN series.

Сфера застосування:

- Домашній мультимедійний центр (домашній кінотеатр).
- Офісні робочі місця.
- Освітні заклади та навчальні аудиторії.

IoT-комп'ютери (Internet of Things) - це малі обчислювальні пристрої, призначені для збору, обробки та передавання даних з датчиків і виконавчих модулів, які працюють у складі розподілених систем «Інтернету речей». (рис. 14)

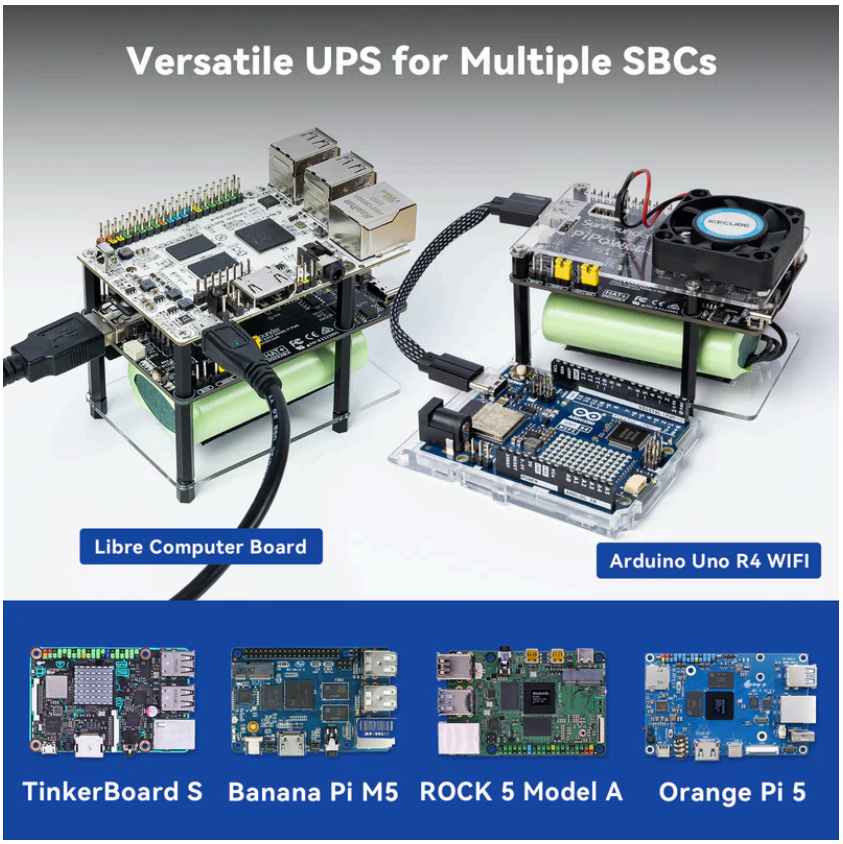


Рис. 13 - IoT-комп'ютери

Характеристика:

- Мають невеликі розміри, низьку вартість та високу енергоефективність.
- Переважно базуються на ARM-архітектурі або мікроконтролерах (Raspberry Pi, Arduino, ESP32).
- Підтримують роботу з датчиками, модулями введення/виведення та бездротовими інтерфейсами (Wi-Fi, Bluetooth).
- Орієнтовані на виконання вузькоспеціалізованих завдань у реальному часі.

Приклади: Raspberry Pi 5, Arduino Uno R4, ESP32.

Сфера застосування:

- Системи автоматизації та «розумний дім».
- Робота з датчиками та сенсорними системами.
- Робототехніка та вбудовані системи керування.
- Навчальні та експериментальні проєкти з програмування й електроніки.

Геймерські консолі - це спеціалізовані обчислювальні пристрої, призначені насамперед для запуску відеоігор та мультимедійного контенту, з апаратною і програмною частинами, оптимізованими під ігрові навантаження. (рис. 14)

Рис. 14 - Геймерські консолі

Характеристика:

- Орієнтовані на ігровий та мультимедійний контент.
- Мають архітектуру, подібну до персонального комп'ютера, але з глибокою оптимізацією під ігри.
- Працюють у закритій екосистемі з власними операційними системами та сервісами.
- Не передбачають апаратної модернізації або мають її суттєві обмеження.

Приклади: PlayStation 5, Xbox Series X, Nintendo Switch, Steam Deck, ASUS ROG Ally.

Сфера застосування:

- Відеоігри та розважальний контент.
- Онлайн-сервіси та стримінгові платформи.
- Віртуальна та доповнена реальність (VR/AR).

Завдання для виконання роботи

Завдання №1: Аналіз власного комп'ютера. Використовуючи програми **CPU-Z** або **Speccy**, визначити характеристики свого комп'ютера.

- **CPU** – модель, кількість ядер і потоків;
- **RAM** – обсяг, тип, частота;
- **GPU** – відеокарта або інтегрована графіка;
- **Накопичувачі** – HDD/SSD, обсяг, тип;
- **ОС** – версія.

Завантажте програми для діагностики ПК:

- CPU-Z та Speccy → https://lnk.ua/5V1R0m5Nd або Google Drive

Завдання №2: Пошук аналога

1. Перейти на сайт **https://rozetka.com.ua/** або **https://hotline.ua**
2. У категорії **«Популярні комп'ютери або ноутбуки»** знайти модель у **схожій ціновій категорії**.
3. Записати її характеристики (CPU, RAM, GPU, накопичувач, ОС).

Завдання №3: Порівняння

1. Скласти таблицю порівняння власного комп'ютера та популярного комп'ютера (таблиця 2).
2. Визначити переваги та недоліки кожної системи на основі наведених характеристик.

Таблиця 2 - Порівняльна характеристика

Компонент	Мій комп'ютер	Популярний комп'ютер	Порівняння
CPU	Intel Core i5-7400 (4 ядра)	Intel Core i5-13400F (10 ядер)	Новий у 2–3 рази швидший
RAM	8 GB DDR4-2400	16 GB DDR5-5200	У сучасному більше і швидша

Завдання №4: Пропозиція покращень (апгрейду)

1. Визначити слабкі місця свого комп'ютера (CPU, RAM, GPU, накопичувач).
2. Запропонувати 1–2 можливих покращення.

Лабораторна робота №16

Процесори та пам’ять сучасних комп’ютерних систем

Теоретичні відомості

Центральний процесор (CPU) - це основний обчислювальний елемент комп’ютера, який виконує обробку даних, команди користувача та керує роботою інших компонентів системи. Протягом багатьох років провідними виробниками процесорів залишаються компанії **Intel** та **AMD**, конкуренція між якими є важливим чинником розвитку комп’ютерних технологій.

Зовні процесор має вигляд компактного модуля з великою кількістю контактів для підключення до материнської плати та металевою теплорозподільною кришкою. Усередині він містить складну мікроструктуру з мільйонів (і мільярдів) транзисторів, що забезпечують високу продуктивність сучасних комп’ютерів. (рис. 1)

Рис. 1 - Сучасні центральні процесори Intel та AMD

Техпроцес

Основним матеріалом для виготовлення процесорів є кремній. З очищеного кремнію формують монокристал, який розрізають на тонкі пластини (вафлі) товщиною близько 1 мм.

На цих пластинах за допомогою технології **фотолітографії** створюються напівпровідникові структури майбутніх процесорів. Суть процесу полягає у формуванні мікроскопічних схем шляхом локального впливу заряджених частинок

через спеціальні трафарети, що дозволяє створювати в кремнії структури з мільйонів транзисторів.

Сучасні технології дають змогу виготовляти транзистори розміром у десятки нанометрів (7 нм і менше). Чим тонший техпроцес, тим більше транзисторів можна розмістити на одному кристалі, що підвищує продуктивність і енергоефективність процесора.

Після формування напівпровідникову структуру вирізають з кремнієвої пластини, монтують на підкладку з контактами для підключення до материнської плати та захищають металевою теплорозподільною кришкою.

Поняття архітектури, ядра, ревізії процесора

Процесори постійно еволюціонують, удосконалюється як технологія їх виготовлення, так і внутрішня структура. Кожне нове покоління відрізняється будовою та характеристиками компонентів.

Процесори, що використовують однакові базові принципи побудови, належать до однієї архітектури (мікроархітектури). У межах однієї архітектури процесори можуть мати різні параметри - частоти, техпроцес, обсяг і структуру кеш-пам’яті. Такі варіанти реалізації називають різними ядрами.

Під час доопрацювання одного ядра виробники можуть вносити незначні зміни для усунення помилок або оптимізації роботи. Такі зміни називають ревізіями.

Архітектурам, ядрам і ревізіям надаються відповідні назви та позначення. Наприклад, процесори Intel Core 2 Duo належали до мікроархітектури Intel Core та виготовлялися з різними ядрами (Allendale, Conroe, Merom тощо), кожне з яких мало кілька ревізій.

Основні характеристики процесора

Кількість обчислювальних ядер.

Багатоядерні процесори містять два і більше обчислювальних ядер на одному кристалі або в одному корпусі. Багатоядерність є основним напрямом підвищення продуктивності сучасних CPU. Якщо раніше стандартом були 2–4

ядра, то нині споживчі процесори зазвичай мають 8–16 ядер, а високопродуктивні моделі - до 24–32.

У серверному сегменті кількість ядер значно більша: **AMD EPYC** - до 128, **Intel Xeon** - до 64. Для спеціалізованих задач створюються процесори з сотнями й тисячами ядер (наприклад, Wafer-Scale Engine).

Кількість потоків.

Кількість потоків показує, скільки завдань процесор може виконувати одночасно. Чим більше потоків - тим вища ефективність у багатозадачності та програмах, оптимізованих під паралельні обчислення. Кількість потоків не завжди дорівнює кількості ядер.

- У Intel цю можливість забезпечує технологія Hyper-Threading,
- У AMD - SMT (Simultaneous Multi-Threading).

Завдяки цим технологіям одне фізичне ядро може обробляти два потоки. Наприклад, сучасний процесор AMD Ryzen 9 7950X має 16 ядер і 32 потоки, а Intel Core i9-14900K - 24 ядра (8 продуктивних + 16 енергоефективних) та 32 потоки.

Розмір кеша 2 і 3 рівнів.

Кеш - це внутрішня пам'ять процесора, яка використовується ним як буфер для тимчасового зберігання інформації, що обробляється в конкретний момент часу. Чим кеш більший - тим краще. Чим більший обсяг кешу - тим менше затримок при доступі до даних і тим вища загальна продуктивність системи.

Розглядаючи процесори **2020** років кеш **L2** у більшості процесорів (AMD Ryzen 3000/5000, Intel Core 10–11 покоління) становив **256–512 КБ на ядро**, **L3** у споживчих процесорах коливався в межах **8–32 МБ** (Ryzen 7, Core i7/i9).

У сучасних процесорах:

- **кеш L2** у нових архітектурах (AMD Zen 4 / Zen 5, Intel Raptor Lake / Arrow Lake) збільшено до **1–2 МБ на ядро**, що дало помітний приріст у швидкодії.
- **кеш L3** (спільний для кластера або всіх ядер) може сягати **32–96 МБ** у десктопних моделях, наприклад **AMD Ryzen 7000/9000** чи **Intel Core i9-14900K**.

Частота процесора.

Тактова частота визначає швидкість виконання операцій: чим вона вища, тим більша продуктивність за інших рівних умов.

Швидкість шини процесора (FSB, HyperTransport або QPI).

Через цю шину центральний процесор взаємодіє з материнською платою. Її швидкість (частота) вимірюється в мегагерцах і чим вона вища - тим краще.

Наявність і продуктивність відеоядра.

Багато сучасних процесорів оснащені інтегрованим графічним ядром (iGPU), що дозволяє працювати без окремої відеокарти. Такі процесори забезпечують комфортну роботу з офісними програмами, веб-серфінг, перегляд відео високої роздільності (у т.ч. 4K), базове редагування мультимедіа та запуск невимогливих ігор.

У моделях з графікою **Intel Iris Xe** або **AMD Radeon Graphics** продуктивності достатньо навіть для сучасних ігор на середніх налаштуваннях. За відсутності потреб у професійному геймінгу чи важких графічних задач інтегроване відеоядро дозволяє зменшити вартість системи, відмовившись від дискретної відеокарти.

Тип і максимальна швидкість підтримуваної оперативної пам'яті.

Процесор визначає підтримуваний тип і максимальну частоту оперативної пам’яті. Якщо встановити модулі з вищою частотою, ніж підтримує контролер пам’яті CPU, вони працюватимуть на зниженій швидкості.

Тому оптимально підбирати оперативну пам’ять відповідно до можливостей процесора та материнської плати, що забезпечує ефективну роботу системи й запобігає зайвим витратам.

Сокет, слот

Важливим моментом, який треба враховувати при виборі процесора, є те, для установки в сокет якого типу він призначений. Сокет (socket, роз'єм центрального процесора) - це щілинний або гніздовий роз'єм на материнській платі, у який встановлюється процесор. Кожен процесор можна встановити лише на материнську плату з підходящим роз'ємом, що має відповідні розміри,

необхідну кількість і структуру контактних елементів. Кожен новий сокет розробляється виробниками процесорів, коли можливості старих роз'ємів вже не можуть забезпечити нормальну роботу нових виробів (рис. 2).

Для процесорів Intel починаючи з 2020 року основний сокет - **LGA1200** (10-те покоління Comet Lake, 11-те Rocket Lake). Перед цим використовувався **LGA1151** (6–9 покоління Core), що підтримували пам’ять: **DDR4** до ~3200 МГц.

Сумісність: процесори від 6-го до 9-го покоління працювали на **LGA1151**, але 10-те і 11-те вже вимагали LGA1200.

Наступний крок до 2024 року широко використовувався сокет **LGA1700** (12–14 покоління Core), а у 2024 році компанія представила **LGA1851**, призначений для нових процесорів **Meteor Lake-S та Arrow Lake-S**, з підтримкою **DDR5** (DDR4 більше не використовується).

Рис. 2 - Порівняння сокетів процесорів Intel: LGA1200 та LGA1700

Встановлювати нові процесори у старі сокети неможливо - доводиться міняти і процесор, і материнську плату.

Для процесорів AMD у 2020 році основним залишався сокет AM4, представлений ще у 2017 році. Він підтримував широкий спектр процесорів - від Ryzen першого покоління до Ryzen 5000 (Zen 3), які з’явилися наприкінці 2020

року. Пам’ять: DDR4, оптимально - частоти від 3000 до 3600 МГц. AM4 став однією з найдовговічніших платформ на ринку: завдяки оновленням BIOS користувачі могли модернізувати систему без заміни материнської плати протягом більш ніж 5 років. У 2022 році AMD представила новий сокет AM5 (LGA1718), який використовується у процессорах Ryzen 7000 / 8000 / 9000 серій (рис. 3). AM5 повністю перейшов на DDR5 (сумісності з DDR4 немає). AMD офіційно заявила, що AM5 підтримуватиметься щонайменше до 2027 року, тож ця платформа має великий запас для майбутніх апгрейдів.

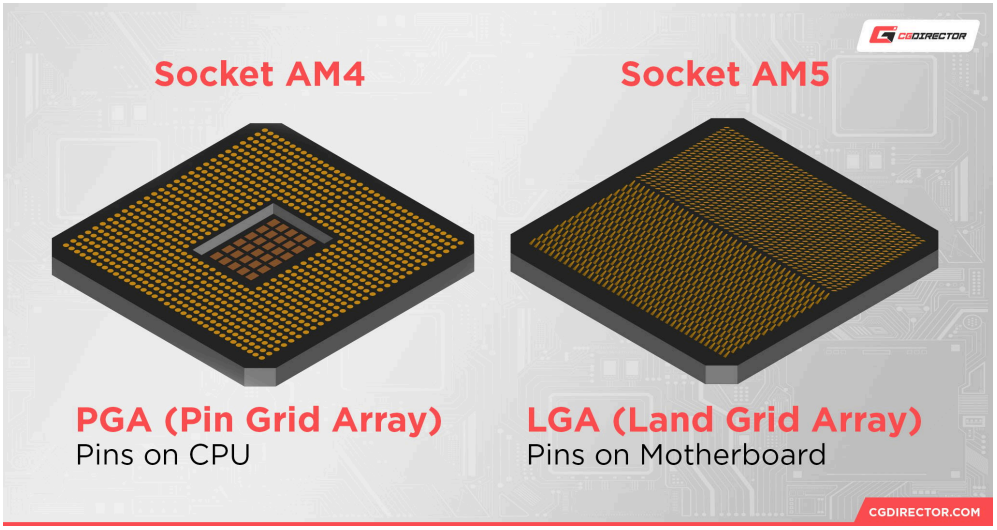


Рис. 3 - Порівняння сокетів процесорів AMD: AM4 та AM5

Якщо ви задумали модернізувати старий комп'ютер шляхом придбання продуктивнішого процесора, переконайтеся, що по сокету він підійде до вашої старої материнської плати. Інакше однозначно доведеться замінювати і її.

Встановлювати центральний процесор в сокет системної плати треба обережно, щоб не пошкодити контакти.

Система охолодження процесора

Процесор потребує ефективного охолодження, оскільки перегрів може призвести до зниження продуктивності або виходу з ладу. Його поверхня закрита металевою кришкою **IHS (Integrated Heat Spreader)**, що виконує захисну та тепловідвідну функції. На процесор встановлюється система охолодження

(повітряний кулер або рідинний контур), радіатор якої має щільно контактувати з поверхнею через шар термопасти.

При виборі охолодження необхідно враховувати **TDP процесора**. Штатний кулер, що постачається з CPU, достатній для роботи на базових частотах, але може бути неефективним за високих навантажень або під час розгону.

Сучасні процесори стабільно працюють при температурах до **80–95 °C**, а критичні значення становлять **100–105 °C**. У разі перегріву процесор автоматично знижує частоту або вимикається. Основними причинами перегріву є слабке охолодження, пил, висохла термопаста або несправність вентилятора чи помпи. Для контролю температур використовують спеціальні програми моніторингу.

Який процесор вибрати для комп'ютера

Вибір процесора залежить від завдань, для яких використовується комп’ютер. Надлишкова продуктивність призводить до зайвих витрат, тоді як занадто слабкий процесор обмежує можливості системи.

Для ігор. Для простих та онлайн-ігор достатньо процесора з **6 ядрами і 12 потоками** (наприклад, Ryzen 5 або Core i5). Для сучасних ресурсомістких ігор (Triple-A, VR) оптимальним є вибір процесорів з **8–12 ядрами**, які у поєднанні з потужною відеокартою забезпечують високу продуктивність.

Для дому та офісу. Для роботи з офісними програмами, навчання та перегляду мультимедіа достатньо **4–6 ядерних** процесорів початкового або середнього рівня. Такі моделі підтримують сучасні стандарти пам’яті та залишаються актуальними протягом кількох років.

Для роботи з вимогливими програмами. Для професійних задач (програмування, обробка фото й відео, 3D-графіка) важливі **кількість ядер і потоків, висока частота та обсяг кеш-пам’яті**. Оптимальними є процесори середнього та високого класу з **8–16 ядрами**, які ефективно працюють у багатозадачному режимі.

Вибір процесора для комп'ютера: що важливо знати

Що потрібно знати при виборі процесора? В першу чергу, основні характеристики, які відповідають за швидкодію комп'ютера.

Кількість ядер

У сучасних процесорах кількість ядер варіюється від 4 у бюджетних моделях до 16 у десктопних флагманах і навіть 64–128 у серверних рішеннях.

Вибір залежить від завдань:

- для офісу та навчання достатньо 4–6 ядер,
- для ігор - 8–12 ядер,
- для професійної роботи (монтаж, рендеринг, AI) - 16 і більше.

Також важливо враховувати покоління процесора: новіші архітектури (наприклад, Intel Raptor Lake / Arrow Lake або AMD Ryzen Zen 4 / Zen 5) при тій же кількості ядер забезпечують значно кращу продуктивність, ніж попередні.

Кількість потоків

Завдяки технологіям **Hyper-Threading** (Intel) та **SMT** (AMD) одне ядро може виконувати два потоки, що підвищує ефективність у багатозадачності. У сучасних системах нормою є 8–24 потоки, у топових моделях - 32 і більше.

Частота процесора

Тактова частота впливає на швидкість виконання операцій, але не є єдиним показником продуктивності. На результат також впливають архітектура, кількість ядер і кеш-пам’ять. У сучасних CPU базові частоти становлять 3–4 ГГц, у турборежимі - до 5–6 ГГц.

Тип і частота підтримуваної оперативної пам'яті

Продуктивність системи залежить від пропускної здатності пам’яті. У 2025 році актуальним стандартом є **DDR5** з оптимальними частотами **5600–6400 MT/c**. Для бюджетних систем достатньо DDR5-5200/5600, для ігор і професійних задач - DDR5-6000 і вище.

Кеш-пам'ять

Кеш-пам’ять (L1, L2, L3) зменшує затримки доступу до даних і суттєво впливає на продуктивність. У сучасних процесорах обсяг кешу L3 у десктопних моделях сягає **32–96 МБ**, а у високопродуктивних і серверних рішеннях - значно більше.

На що не потрібно звертати увагу

Виробник процесора

Основними виробниками процесорів є **Intel** та **AMD**, і жоден з них не має абсолютної переваги. Intel традиційно демонструє високу продуктивність на одне ядро, що важливо для програм і ігор з обмеженою багатопоточністю, а також часто пропонує зручні рішення з інтегрованою графікою. AMD, своєю чергою, вигідніший за співвідношенням ціна/можливості та добре підходить для багатопоточних задач і сучасних ігор, що активно використовують кеш і багато ядер.

Водночас це порівняння є умовним: за правильно підібраної конфігурації як процесори Intel, так і AMD забезпечують комфортну роботу з вимогливими програмами та сучасними іграми. Вирішальним фактором є не бренд, а баланс усієї системи.

«Ігровий» або «неігровий» процесор

Поняття «ігровий процесор» є умовним і не означає окремого класу CPU. Бюджетні моделі з **6–8 ядрами і 12–16 потоками** забезпечують комфортний ігровий процес у Full HD, тоді як процесори верхнього сегмента з **12–16 ядрами** дозволяють грати на максимальних налаштуваннях у високих роздільностях за наявності потужної відеокарти.

Не існує одного параметра, що визначає процесор як «ігровий». Його придатність до геймінгу залежить від сукупності характеристик: кількості ядер і потоків, частоти, кеш-пам’яті, архітектури, а також від відеокарти й типу оперативної пам’яті.

Оперативний запам’ятовуючий пристрій пам'ять

Це пристрій, також зветься RAM або ОЗП. Цей компонент є енергозалежним. При роботі ПК в ньому зберігається код, який виконує система. Він представлений різними програмами, прийнятими і переданими, а також проміжними даними, які обробляє CPU (рис. 4).

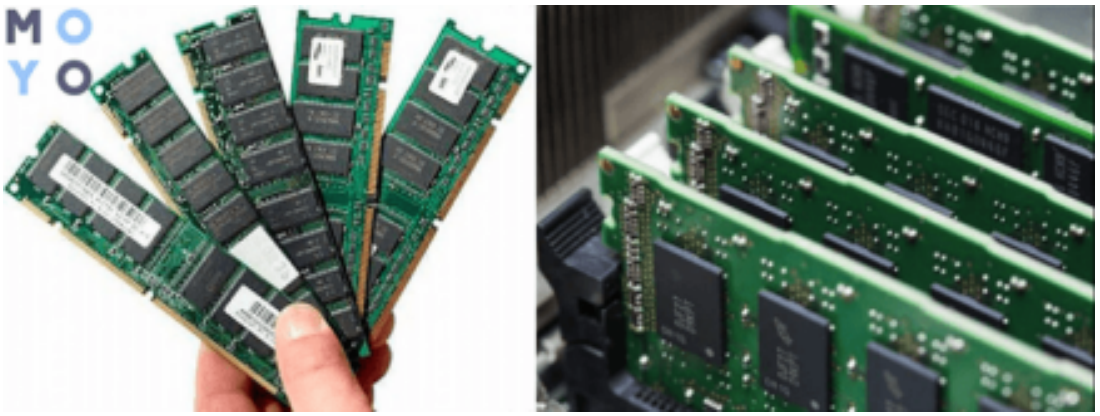


Рис. 4 – Модулі оперативної пам'яті (RAM)

Типи ОЗП

Ділиться на 5 типи в залежності від хронології. Кожен новий варіант стає потужнішим і швидшим.

DDR - першопроходець. На даний момент він не актуальний, тому що його потужності недостатньо для того, щоб впоратися з обробкою сучасного об'єму даних: перші модулі працювали на частоті 400 MHz.

DDR2 - удосконалений тип пам'яті, який перевершив перше покоління майже вдвічі за швидкістю. Частота роботи - 400–1066 MHz, при цьому споживання енергії знижено до 1,8 В (проти 2,5 В у DDR). У свій час був популярним у настільних ПК.

DDR3 - значно вдосконалений варіант, який дозволив отримати приріст продуктивності до 10–15% у порівнянні з DDR2. Пам'ять працює на частотах 800–2133 MHz, має нижчу напругу (1,5 В) і вищу стабільність. Цей тип широко використовувався в комп'ютерах і ноутбуках аж до середини 2020-х років.

DDR4 - з'явився у 2014 році та став основним стандартом оперативної пам'яті до початку 2020-х. Частота роботи - 2133–4266 MHz, напруга - 1,2 В. Цей тип забезпечує високу швидкість доступу до даних і значно менше енергоспоживання. Він використовується у більшості сучасних комп'ютерів і серверів.

DDR5 - найновіше покоління оперативної пам'яті, представлене у 2020 році. Частоти - від 4800 до 8400 MHz, напруга - 1,1 В. DDR5 має удвічі більшу

пропускну здатність у порівнянні з DDR4 і вбудовані контролери живлення, що підвищують стабільність роботи. На сьогодні цей тип є актуальним стандартом 2025 року і застосовується у високопродуктивних ПК, ноутбуках та серверах.

Щоб обрати модулі оперативної пам’яті, які забезпечать комфортну швидкодію ПК, потрібно враховувати декілька основних параметрів, а саме: об’єм ОЗП, частота, таймінги оперативної пам'яті, напруга живлення, форм-фактор

Об'єм ОЗП

Один з основних показників, який вказується в гігабайтах (ГБ). Тут діє просте правило: чим більше - тим краще, адже від цього залежить, скільки даних здатна зберігати оперативна пам’ять під час роботи системи.

Якщо обсягу пам’яті недостатньо, комп’ютер починає активно використовувати жорсткий диск або SSD як тимчасову пам’ять (файл підкачки), що істотно знижує швидкодію. Для офісних машин оптимально мати 8 ГБ. Цього вистачає для роботи з текстовими документами, браузером та базовими офісними програмами. Сучасні застосунки стають дедалі вимогливішими, тому краще мати запас у 12–16 ГБ, навіть для повсякденної роботи. Для комп’ютерів, на яких планується запуск “важкого” програмного забезпечення - наприклад, систем моделювання, обробки фото, відео чи 3D-графіки - доцільно використовувати від 16 ГБ оперативної пам’яті. Приклад сучасного модуля - Kingston Fury Beast DDR5-5600 16GB або Corsair Vengeance LPX DDR4-3200 16GB (рис. 5).

Рисунок 5 – Модуль оперативної пам’яті DDR5

Для професійних робочих станцій і геймерських ПК (обробка 4K-відео, створення візуальних ефектів, студійний звукозапис, ігри нового покоління)

оптимальним вважається обсяг 32–64 ГБ.

Сучасні материнські плати підтримують двоканальний або навіть чотириканальний режим роботи оперативної пам’яті. Це означає, що для підвищення продуктивності краще встановлювати два або чотири однакові модулі пам’яті, а не один великий.

Наприклад, замість однієї планки на 16 ГБ краще встановити дві по 8 ГБ - у такому випадку дані передаються паралельно по двох каналах, що збільшує швидкодію системи до 15–25%. Крім того, таймінги (затримки) і частоти роботи модулів мають збігатися. Тому потрібно встановлювати комплект пам’яті (kit) від одного виробника, наприклад 2×8 ГБ DDR5-6000 з однаковими параметрами.

Частота

Вказується в MHz і відображає пропускну здатність модульних каналів.

Природно, чим вона вище, тим спритніше інфо обробляється і передається на материнську плату, а далі - в ЦП або накопичувач. Пам'ять DDR4 здатна забезпечувати частоту понад 2400 MHz, наприклад, HX440C19PB3AK2/16 працює на 4 тисячах МГц. Це дає відчутний приріст швидкодії, особливо якщо мова йде про збірки на базі сучасних процесорів типу AMD RYZEN. Вони кілька більш чутливі саме до частоти, якщо порівнювати з процесорами Intel.

Таймінги оперативної пам'яті

Це показник затримки даних при їх перенесенні між модулями RAM і CPU. Відповідно, чим він нижчий, тим швидше переноситься інформація і функціонує ПК. Таймінги зазвичай позначаються набором чисел через дефіс, наприклад: CL16-18-18-36 або CL40-40-40-77.

У нових поколіннях пам’яті (зокрема DDR5) таймінги номінально вищі, ніж у DDR4, але завдяки зростанню частоти й оптимізації контролера пам’яті реальна затримка в наносекундах часто навіть менша.

Напруга живлення

Оперативна пам’ять є енергозалежною, тому в її характеристиках вказується **номінальна напруга**, необхідна для стабільної роботи при базових частотах і таймінгах. Для різних типів ОЗП вона відрізняється:

- **DDR3** - 1,5 В (1,35 В для DDR3L),
- **DDR4** - 1,2 В,
- **DDR5** - 1,1 В (у деяких моделях - до 1,05 В).

Під час розгону підвищення частоти й зміна таймінгів зазвичай вимагають збільшення напруги, що призводить до зростання температури компонентів і може негативно впливати на стабільність системи. Тому всі розгінні дії слід виконувати поступово, з перевіркою після кожного кроку, оскільки надмірні значення можуть призвести до нестабільної роботи або пошкодження обладнання.

Оперативна пам'ять з радіатором

Модулі ОЗП з інтегрованими радіаторами забезпечують додаткове охолодження та мають привабливий зовнішній вигляд. Для звичайних офісних або домашніх ПК вони не є обов’язковими, але корисні для геймерів і користувачів, які планують розгін або роботу під високими навантаженнями.

Форм-фактор оперативної пам’яті

Форм-фактор оперативної пам’яті визначає фізичний розмір модуля, кількість контактів і сумісність з материнською платою. Саме він визначає, чи пі

Основні стандарти та форм-фактори

DDR3 (≈ 2007–2016 рр.):

- **DIMM DDR3** - для настільних ПК, **240 контактів**.
- **SO-DIMM DDR3** - для ноутбуків, **204 контакти**.

DDR4 (з 2014 р.):

- нижча напруга (1,2 В), вища пропускна здатність;
- **DIMM DDR4** - **288 контактів**, настільні ПК;
- **SO-DIMM DDR4** - **260 контактів**, ноутбуки.

DDR5 (з 2021 р.):

- удвічі більша пропускна здатність, напруга 1,1 В;
- вбудований контролер живлення (PMIC);
- **DIMM DDR5** - **288 контактів**, але інший виріз, ніж у DDR4;
- **SO-DIMM DDR5** - **262 контакти**, ноутбуки нового покоління.

дходить пам’ять для настільного ПК, ноутбука або сервера.

Пам'ять для ноутбуків. Модулі **SO-DIMM** відрізняються від десктопних:

- меншими розмірами та кількістю контактів;
- відсутністю радіаторів через обмежений простір корпусу;
- незначними відмінностями у характеристиках, які залишаються в межах стандарту.

Розуміючи всі ключові показники ОЗП, можна без зусиль підібрати необхідну для складання ПК модель. Це стане в нагоді і для більш повної оцінки продуктивності PC.

Збільшення обсягу оперативної пам’яті позитивно впливає на швидкодію системи, оскільки дозволяє уникнути вивантаження тимчасових файлів у повільний жорсткий диск або SSD. Як наслідок - підвищується швидкість відкриття програм, робота браузера і загальна стабільність системи. Обсяг пам’яті потрібно підбирати відповідно до архітектури операційної системи: 64-бітні ОС (Windows, Linux, macOS) здатні ефективно працювати з великими обсягами - від 8 до 64 ГБ і більше, залежно від потреб користувача.

Як підібрати ОЗП необхідного обсягу і швидкості

Оперативна пам’ять повинна мати достатній обсяг, якщо комп’ютер використовується для сучасних ігор або професійної роботи з аудіо- та відеофайлами. У більшості випадків **8–16 ГБ ОЗП** є оптимальним вибором і дозволяє уникнути модернізації ПК протягом кількох років. Більші обсяги зазвичай потрібні лише для серверних або спеціалізованих систем.

Навіть для найвимогливіших сучасних ігор **16 ГБ оперативної пам’яті є достатнім обсягом**. У такому разі доцільніше звертати увагу не на збільшення обсягу, а на швидкість пам’яті. Водночас слід пам’ятати, що **частота ОЗП обмежується можливостями материнської плати**, тому придбання модулів із частотою, вищою за підтримувану, є економічно недоцільним.

Коли потрібен апгрейд:

Оновлення RAM шляхом встановлення додаткового модуля або заміни старого є доцільним не лише у разі виходу пам’яті з ладу. Збільшення обсягу ОЗП безпосередньо підвищує продуктивність системи, особливо під час багатозадачної роботи. Перед вибором нової оперативної пам’яті необхідно з’ясувати:

- який тип ОЗП підтримує система;
- кількість слотів для оперативної пам’яті на материнській платі.

Як дізнатися, який тип ОЗП встановлений

У Windows 10 базову інформацію про оперативну пам’ять можна переглянути через Диспетчер завдань → вкладка «Продуктивність», де відображається тип і поточна частота пам’яті. Для отримання детальніших даних рекомендовано використовувати програму CPU-Z.

Інформація в CPU-Z:

- SPD - кількість слотів, тип, обсяг, виробник модуля
  (якщо дані відсутні - слот порожній);
- Memory - загальний обсяг ОЗП, тип, частота та таймінги;
- Mainboard - відомості про материнську плату і підтримувані типи пам’яті.

Як правильно вибирати нові модулі ОЗП

Приклад: у комп’ютері встановлено 1 модуль 4 ГБ DDR4, є 2 слоти, а максимальний підтримуваний обсяг - 16 ГБ. Можливі варіанти апгрейду:

- встановити два модулі по 4 ГБ (подвоєння обсягу);
- замінити старий модуль на два модулі по 8 ГБ (максимальний обсяг).

Важливо: різні частоти, обсяги й таймінги модулів можуть негативно впливати на стабільність системи. Тому рекомендовано:

- купувати **комплекти модулів**;
- або додавати **ідентичний модуль** до вже встановленого.

Також слід враховувати, що в двоканальному режимі продуктивність оперативної пам’яті зростає. Максимальний обсяг, зазначений виробником,

стосується усіх слотів разом, а не одного модуля.

Як встановити оперативну пам'ять в ноутбук

Як правило, виробники комплектують кількома гніздами під ОЗП і передбачають швидкий доступ до них, оснащуючи корпус окремою кришкою. Якщо ж її немає, тоді всю нижню частину доведеться відкручувати і прибирати.

Послідовність дій:

1. Знеструмити лептоп.
2. Зняти акумулятор. Якщо батарея незнімна, її знадобиться відключити після того, як користувач добереться до апаратної частини пристрою.
3. Зняти гвинти з кришки викруткою. Під нею користувач виявить оперативну пам'ять, планки якої стоять в слотах.
4. Якщо необхідно, витягти старі модулі, акуратно відігнувши фіксатори.
5. Вставити нову планку так, щоб пази збіглися. Потрібно фіксувати модулі щільно: до клацання.
6. Підключити акумулятор.
7. Включити ноутбук.

Завдання для виконання роботи

Завдання 1. Порівняння процесорів і вибір базової платформи

1. Зробити порівняння процесорів, скласти таблицю основних відмінностей по характеристиках.
2. Скласти **таблицю основних характеристик** процесорів, використовуючи спеціалізовані ресурси:

- https://cpu.userbenchmark.com
- https://www.chaynikam.info/ukr/cpu_comparison.html
- https://technical.city/en/cpu

3. **Обґрунтувати вибір одного процесора** як основи для подальшої конфігурації ПК (ігрової, офісної або робочої).

Варіанти робіт

1. AMD Ryzen 7 5700X, Intel Core i5-12400F, AMD Ryzen 5 7600
2. AMD Ryzen 5 5600X, Intel Core i5-13600K, AMD Ryzen 7 7800X3D
3. AMD Ryzen 5 5500, Intel Core i5-13400, AMD Ryzen 9 7900
4. Intel Core i5-10400F, AMD Ryzen 5 3600, AMD Ryzen 5 5600G
5. AMD Ryzen 9 5900X, Intel Core i7-12700, AMD Ryzen 7 7700
6. AMD Ryzen 7 5800X, Intel Core i5-11400F, Intel Core i5-13500
7. AMD Ryzen 5 7600X, Intel Core i5-12600K, AMD Ryzen 7 7700X
8. AMD Ryzen 9 7950X, Intel Core i9-13900K, AMD Ryzen 9 7900X
9. AMD Ryzen 7 7700, Intel Core i5-12600KF, Intel Core i7-13700F
10. AMD Ryzen 7 5700G, Intel Core i5-9400F, AMD Ryzen 5 4600G
11. AMD Ryzen 3 3200G, Intel Core i5-7500, AMD Ryzen 3 5300G
12. AMD Ryzen 3 4100, Intel Core i5-6500, AMD Ryzen 5 5500U

Приклад порівняльної таблиці

Процесор	**Ядра / Потоки**	**Частота (базова/турбо)**	**Кеш L3**	**TDP**	**Пам'ять (DDR4/DDR5)**	**Вбудована графіка**	**Продуктивність (UserBenchmark, %)**
AMD Ryzen 5 5600							
Intel Core i5-12400							
Intel Core i9-15900K							

У висновку порівняти дані процесори за схемою: *завдання → що краще → чому → підсумок*.

Завдання 2. Аналіз оперативної пам'яті власного комп'ютера або ноутбука

1. Встановити та запустити програму **CPU-Z**.
2. Проаналізувати оперативну пам'ять власного комп'ютера або ноутбука за допомогою програми CPU-Z.
3. Отримані результати **занести до таблиці 1.**

4. На основі зібраних даних **запропонувати оптимальний варіант оновлення оперативної пам’яті**.

Таблиця 1 - Аналіз оперативної пам’яті комп’ютера

Вкладка	**Що потрібно дізнатись**	**Дані персонального ноутбука**
SPD	1. Кількість гнізд.	
	2. Тип.	
	3. Обсяг.	
	4. Виробника.	
Memory	Загальному обсязі оперативки, атакож інформація про таймінги.	
Mainboard	Визначити, які типи пам'яті і з яким обсягом підтримує лептоп.	

Лабораторна робота №17

BIOS та UEFI. Завантаження системи

Теоретичні відомості

Інтерфейс більшості версій BIOS, за винятком найсучасніших, представляє примітивну графічну оболонку, де є кілька пунктів меню, з яких можна перейти в інший екран з вже налаштованим параметрами. Наприклад, пункт меню «Boot» відкриває користувачу параметри розподілу пріоритету завантаження комп'ютера, тобто там можна вибрати девайс, з якого буде виконуватися завантаження ПК.

Всього на ринку є три основні виробники BIOS, і у кожного з них інтерфейс може суттєво відрізнятися. Наприклад, у BIOS компанії AMI (American Megatrends Inc.) використовується верхнє меню з основними розділами налаштувань (рис. 1)

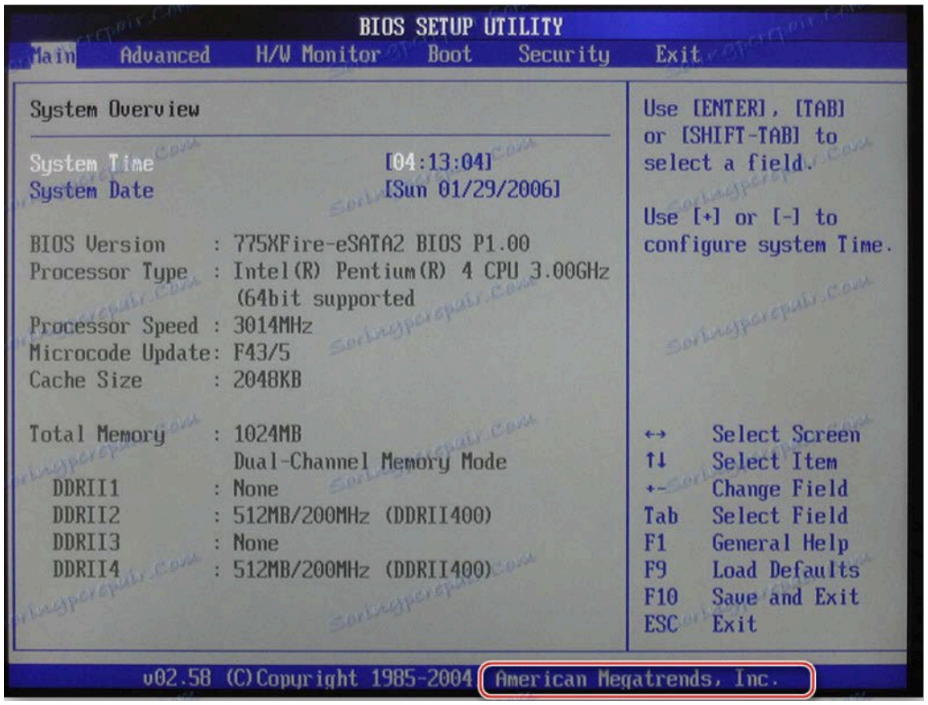


Рис. 1 - Приклад інтерфейсу BIOS виробника AMI

У деяких версій Phoenix і Award всі пункти розділів розташовані на головній сторінці у вигляді стовпчиків. (рис. 2)

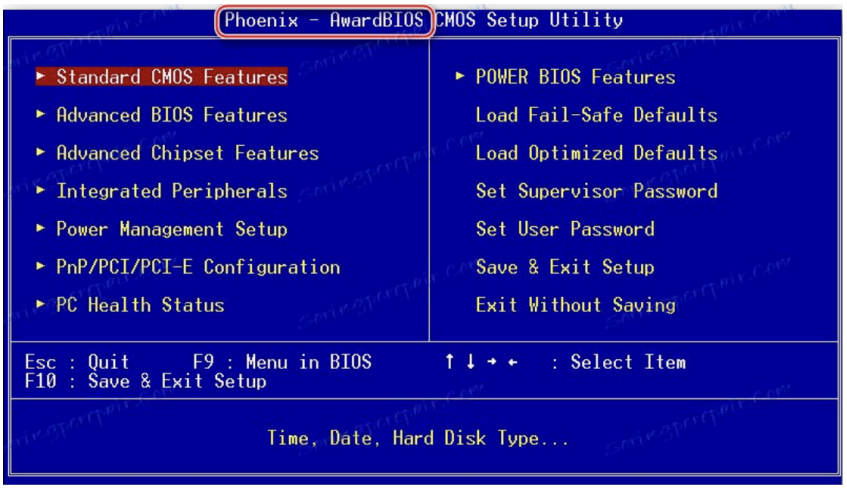


Рис. 2 - Приклад інтерфейсу BIOS Phoenix/Award

Плюс, в залежності від виробника, можуть відрізнятися і назви деяких пунктів і параметрів, хоча сенс вони будуть нести один і той же.

Всі переміщення між пунктами відбуваються за допомогою клавіш зі стрілками, а вибір - за допомогою Enter. Деякі виробники роблять навіть спеціальну виноску в інтерфейсі BIOS, де написано яка клавіша за що відповідає.

В UEFI (найсучасніша різновид BIOS) є більш просунутий призначений для користувача інтерфейс, можливість управління за допомогою комп'ютерної миші, а також переклад деяких пунктів на українську мову (останнє зустрічається досить рідко). (рис. 3)

Рис. 3 - Приклад UEFI-інтерфейсу BIOS (AMI)

Базові налаштування

До базових налаштувань відносяться параметри часу, дати, пріоритету завантаження комп'ютера, різні настройки пам'яті, жорстких дисків і дисководів.

За умови, що ви тільки зібрали комп'ютер, необхідно провести налаштування даних параметрів.

Вони будуть знаходитися в розділі «Main», «Standard CMOS Features» і «Boot». Варто пам'ятати, що в залежності від виробника назви можуть відрізнятися. Для початку налаштуйте дату і час по даній інструкції:

1. У розділі «Main» знайдіть «System time», виберіть його і натисніть Enter для внесення коригувань. Виставте час. В BIOS від іншого розробника параметр «System time» може називатися просто «Time» і знаходитися в розділі «Standard CMOS Features».
2. Аналогічне потрібно проробити з датою. У «Main» знайдіть «System Date» і виставте прийнятне значення. Якщо у вас інший розробник, то дивіться настройки дати в розділі «Standard CMOS Features», потрібний вам параметр повинен називатися просто «Date»

Тепер необхідно зробити настройку пріоритетності жорстких дисків і дисководів. Іноді, якщо її не робити, то система просто не буде завантажуватися.

Всі потрібні параметри знаходяться в розділі «Main» або «Standard CMOS Features» (в залежності від версії BIOS). Покрокова інструкція на прикладі Award / Phoenix BIOS виглядає наступним чином:

1. Зверніть увагу на пункти «IDE Primary Master / Slave» і «IDE Secondary Master, Slave». Виберіть один з цих пунктів за допомогою клавіш зі стрілками і натисніть Enter для переходу до розширених налаштувань
2. Навпаки параметра «IDE HDD Auto-Detection» бажано поставити «Enable», так як він відповідає за автоматичну розстановку розширених налаштувань диска. Ви самі можете їх задати, але для цього доведеться знати кількість циліндрів, оборотів і т. Д. У випадку, якщо щось з цього вказати

неправильно, то диск не працюватиме взагалі, тому найкраще ці настройки довірити системі.

3. Аналогічно слід зробити і з іншим пунктом з 1-го кроку.

Схожі настройки потрібно зробити і користувачам BIOS від AMI, тільки тут міняються SATA-параметри. Використовуйте цей посібник для роботи:

1. У «Main» зверніть увагу на пункти, які носять назву «SATA (номер)». Всього їх буде стільки, скільки жорстких дисків підтримується вашим комп'ютером. Вся інструкція розглянута на прикладі «SATA 1» - виберіть цей пункт і натисніть Enter. Якщо у вас кілька пунктів «SATA», то всі кроки, що нижче потрібно проробити з кожним з пунктів.
2. Перший параметр, який потрібно налаштувати - це «Type». Якщо ви не знаєте тип підключення вашого жорсткого диска, то поставте навпроти нього значення «Auto» і система визначить його самостійно.
3. Перейдіть до «LBA Large Mode». Цей параметр відповідає за можливість роботи дисків з розміром більше 500 МБ, тому навпроти нього обов'язково поставте «Auto».
4. Інші налаштування, аж до пункту «32 bit Data Transfer», поставте на значення «Auto».
5. Навпроти «32 bit Data Transfer» потрібно встановити значення «Enabled».

Користувачі AMI BIOS на цьому можуть закінчити до заводських значень, а ось у розробників Award і Phoenix є ще кілька додаткових пунктів, які потребують участі користувача. Всі вони знаходяться в розділі «Standard CMOS Features». Ось їх список:

1. «Drive A» і «Drive B» - ці пункти відповідають за роботу дисководів. Якщо таких в конструкції немає, то навпаки обох пунктів потрібно поставити значення «None». Якщо дисководи є, то доведеться вибрати тип дисковода, тому заздалегідь рекомендується вивчити всі характеристики свого комп'ютера більш детально;
2. «Halt out» - відповідає за припинення завантаження ОС при виявленні будь-яких помилок. Рекомендується виставити значення «No errors», при

якому завантаження комп'ютера, не буде перериватися, якщо будуть виявлені несерйозні помилки. Всі інформація про останні виводиться на екран.

На цьому стандартні настройки можна завершити. Зазвичай половина з цих пунктів вже буде мати якісь потрібно значення.

Параметри «Advanced»

На цей раз все настройки будуть проводитися в розділі «Advanced». Він є в BIOS від будь-яких виробників, правда, може носити трохи інше найменування. Усередині нього може бути різна кількість пунктів в залежності від виробника.

Розглянемо інтерфейс на прикладі AMI BIOS:

- «JumperFree Configuration». Тут знаходиться велика частина налаштувань, яку необхідно зробити користувачеві. Даний пункт відповідає відразу за настройку напруги в системі, розгін вінчестера і установку робочої частоти для пам'яті. Детальніше про налаштування - трохи нижче;
- «CPU Configuration». Як випливає з назви - тут проводяться різні маніпуляції з процесором, однак якщо ви робите стандартні настройки після складання комп'ютера, то в цьому пункті нічого міняти не потрібно. Зазвичай до нього звертаються, якщо потрібно прискорити роботу ЦП;
- «Chipset». Відповідає за чіпсет і функціонування чіпсета і BIOS. Звичайному користувачеві сюди заглядати не потрібно;
- «Onboard device configuration». Тут налаштовуються конфігурації для спільного функціонування різних елементів на материнській платі. Як правило, всі налаштування зроблені вірно вже автоматом;
- «PCIPnP» - настройка розподілу різних обробників. Вам нічого робити в цьому пункті не треба;
- «USB Configuration». Тут можна налаштувати підтримку USB-портів і USB-пристроїв для введення (клавіатуру, мишку та ін.). Зазвичай всі параметри вже активні за умовчанням, але рекомендується зайти і перевірити - якщо якийсь з них не активний, то підключити його.

Тепер приступимо безпосередньо до налаштувань параметрів з пункту «JumperFree Configuration»:

1. Спочатку замість потрібних параметрів там може бути один або кілька підрозділів. Якщо це так, то перейдіть в той, який називається «Configure System Frequency / Voltage».
2. Перевірте, щоб навпроти всіх параметрів, які там будуть, стояло значення «Auto» або «Standard». Винятки становлять лише ті параметри, де виставлено якесь цифрове значення, наприклад, «33,33 MHz». У них міняти нічого не потрібно
3. Якщо навпаки якогось з них варто «Manual» або будь-яке інше, то виберіть цей пункт за допомогою клавіш зі стрілками і натисніть Enter, щоб внести зміни.

У Award і Phoenix налаштовувати дані параметри не потрібно, так як вони за умовчанням налаштовані вірно і знаходяться зовсім в іншому розділі. Зате в розділі «Advanced» ви знайдете розширені настройки для установки пріоритетів завантаження. Якщо в комп'ютері вже є жорсткий диск з встановленою на ньому операційною системою, то в «First Boot Device» виберіть значення «HDD-1» (іноді потрібно вибрати «HDD-0»).

Якщо ж операційна система на жорсткий диск ще не встановлена, то замість нього рекомендується поставити значення «USB-FDD».

Також у Award і Phoenix в розділі «Advanced» є пункт щодо налаштувань входу в BIOS з паролем - «Password Check». Якщо ви задали пароль, то рекомендується звернути увагу на цей пункт і виставити прийнятне для вас значення, всього їх два:

- «System». Для отримання доступу до BIOS і його налаштувань потрібно ввести вірний пароль. Система буде запитувати пароль від BIOS при кожному завантаженні комп'ютера;
- «Setup». Якщо ви вибрали цей пункт, то зможете увійти в BIOS без введення паролів, але для отримання доступу до його настройок доведеться ввести

пароль, заданий раніше. Вам необхідно ввести пароль тільки тоді, коли ви намагаєтеся увійти в BIOS.

Налаштування безпеки і стабільності

Ця можливість актуальна тільки для власників машин з BIOS від Award або Phoenix. Ви можете включити режим максимальної продуктивності або стабільності. У першому випадку система стане працювати трохи швидше, але при цьому є ризик несумісності з деякими операційними системами. У другому випадку все працює більш стабільно, але повільніше (не завжди).

Щоб включити режим високої продуктивності, в головному меню виберіть «Top performance» і поставте в ньому значення «Enable». Варто пам'ятати, що є ризик порушити стабільність роботи операційної системи, тому попрацюйте в такому режимі кілька днів, і якщо в системі з'являться якісь збої, яких раніше не спостерігалося, то вимкніть його, встановивши значення «Disable».

Якщо ж швидкодії ви віддаєте перевагу стабільність, то рекомендується завантажити протокол безпечних налаштувань, всього їх є два види:

- «Load Fail-Safe Defaults». В цьому випадку BIOS завантажує найбезпечніші протоколи. Однак продуктивність сильно страждає;
- «Load Optimized Defaults». Проводиться завантаження протоколів, виходячи з особливостей вашої системи, завдяки цьому продуктивність страждає не так сильно, як в першому випадку. Рекомендується до завантаження.

Для завантаження будь-якого з цих протоколів потрібно вибрати один з пунктів, розглянутих вище, в правій частині екрана, після чого підтвердити завантаження за допомогою клавіш Enter або Y.

Установка пароля

Після завершення основних налаштувань ви можете задати пароль. У цьому випадку ніхто крім вас не зможе отримати доступ до BIOS і / або можливість будь-яких змінювати його параметри (залежить від налаштувань, які були описані вище).

У Award і Phoenix для того, щоб задати пароль, потрібно в головному екрані вибрати пункт «Set Supervisor Password». Відкриється вікно, куди вводиться пароль довжиною до 8 символів, після введення відкривається аналогічне віконце, де потрібно прописати цей же пароль для підтвердження. При наборі використовуйте тільки латинські символи та арабські цифри.

Для зняття пароля вам потрібно знову вибрати пункт «Set Supervisor Password», але коли з'явиться вікно введення нового пароля, просто залишайте його порожнім і натисніть Enter.

У AMI BIOS пароль задається трохи по іншому. Для початку вам потрібно перейти в розділ «Boot», що у верхньому меню, а там вже знайти «Supervisor Password». Пароль задається і знімається аналогічним чином з Award / Phoenix.

По завершенні всіх маніпуляцій в BIOS вам потрібно вийти з нього зі збереженням раніше зроблених налаштувань. Для цього знайдіть пункт «Save & Exit». У деяких випадках можна скористатися комбінацією клавіш F10.

Налаштовувати BIOS не так складно, як це може здатися на перший погляд. До того ж, більшість з описаних налаштувань часто вже виставлені за замовчуванням так, як це потрібно для нормальної роботи комп'ютера.

Завдання для виконання роботи

1. Ознайомитися з інтерфейсом та налаштуваннями різних версій Setup BIOS.
2. Увійти у Setup BIOS власного компютера та пройтися усіма пунктами. Зробити скріншоти. Налаштування не зберігати..
3. Дослідити, яким чином можна обнулити налаштування Setup BIOS саме вашої плати.

Лабораторне заняття №18

Пристрої зберігання даних: HDD, SSD, NVMe, хмарні сервіси

Теоретичні відомості

Жорсткий диск (вінчестер, Hard Disk Drive - HDD) є основним пристроєм для довготривалого зберігання великих обсягів даних і програм. Конструктивно він являє собою герметичний корпус, усередині якого на спільній осі розміщено кілька жорстких алюмінієвих або скляних пластин круглої форми. Поверхня кожної пластини покрита тонким феромагнітним шаром, на якому здійснюється запис інформації.

Запис і зчитування даних відбувається за допомогою магнітних головок, розташованих над поверхнею дисків на надзвичайно малій відстані (десяті частки мікрометра). Пакет дисків обертається з великою швидкістю (зазвичай 5400–7200 об/хв), тому механічний контакт між головками та поверхнею дисків є недопустимим.

Запис інформації здійснюється шляхом зміни магнітного поля, яке намагнічує відповідні ділянки феромагнітного шару. Під час зчитування намагнічені області індукують електричні сигнали в головках, які підсилюються та передаються на обробку.

Керування роботою жорсткого диска здійснює контролер. У сучасних комп’ютерах функції контролера реалізовані у вигляді мікросхем, розміщених у чіпсеті або безпосередньо на платі накопичувача.

Поверхня кожного диска поділяється на концентричні доріжки, які під час форматування розбиваються на сектори. Доріжки з однаковими номерами на різних пластинах утворюють циліндр. Запис інформації можливий лише після форматування диска.

Для зручності зберігання та організації даних жорсткий диск може бути поділений на кілька логічних дисків, що спрощує структурування інформації та роботу з нею.

Характеристика жорсткого диска

Інтерфейс – набір ліній зв'язку, сигналів, що посилають по цих лініях, технічних засобів (контролерів), що підтримують ці лінії, і правил обміну (протоколів). Сучасні внутрішні жорсткі диски можуть мати інтерфейси ATA (IDE, він же Parallel ATA), (EIDE), Serial ATA, SCSI, SAS, FireWire, USB, SDIO і Fibre Channel.

Ємність – кількість даних, які можуть зберігатися накопичувачем. Ємність сучасних жорстких дисків з форм-фактором 3,5" сягає 4 ТБ.

Швидкість обертання диска – кількість обертів шпинделя за хвилину. Від цього параметра у значній мірі залежать час доступу й швидкість передавання даних. Випускаються вінчестери з стандартними швидкостями обертання: 4200, 5400 (ноутбуки), 7200 (персональні комп'ютери), 10000 і 12000 об./хв. (сервери і високопродуктивні робочі станції). Збільшенню швидкості обертання шпинделя у жорстких дисках для ноутбуків перешкоджає гіроскопічний ефект, впливом якого можна знехтувати у стаціонарно встановлених комп’ютерах.

Місткість буфера – розмір проміжної пам'яті (кеш-пам'яті), що призначена для згладжування різниці швидкостей читання/запису і передавання даних через інтерфейс. У жорстких дисках вона зазвичай може становити 8, 16, 32 або 64 МБ.

Формфактор – майже всі сучасні накопичувачі для персональних комп'ютерів і серверів мають розмір 3,5" або 2,5". Останні частіше застосовують у ноутбуках. Інші поширені формати – 1,8", 1,3" і 0,85".

Час доступу до інформації – від 3 до 15 мс, як правило, мінімальним часом відрізняються серверні диски, максимальним – диски для портативних пристроїв.

Виробники жорстких дисків. Спочатку на ринку було велике різноманіття жорстких дисків, які виробляли багато компаній. У зв'язку з жорсткістю конкуренції та зниженням норм прибутку більшість виробників була або куплена

конкурентами, або перейшла на інші види продукції. Тому зараз більша частина всіх вінчестерів виробляється всього декількома компаніями: Seagate, Western Digital та Toshiba. Fujitsu продовжує випускати жорсткі диски для ноутбуків і SCSI-диски, але покинула масовий ринок.

Форматування низького рівня (фізичне форматування)

Форматування низького рівня жорсткого магнітного диска (ЖМД) є фізичною процедурою, яка виконується однаково незалежно від типу операційної системи та параметрів форматування високого рівня. Під час цього процесу на диску створюється фізична структура секторів і доріжок.

Сектори нумеруються, починаючи з одиниці, та мають фіксований розмір (зазвичай 571 байт), з яких 512 байт доступні для зберігання даних користувача. Кожен сектор складається із заголовка, області даних і завершення. Заголовок містить адресний маркер та адресу сектора (у форматі CHS) разом із контрольною сумою для перевірки цілісності. В області даних, окрім інформації користувача, зберігаються службові дані для корекції помилок (ECC) та контрольні суми.

Під час форматування низького рівня здійснюється:

- запис заголовків і завершень секторів;
- формування проміжків між секторами та доріжками;
- заповнення секторів тестовими або довільними даними;
- перевірка читаності секторів і цілісності інформації.

У разі виявлення непоправних помилок сектор позначається як дефектний і надалі не використовується для зберігання даних.

Повне форматування низького рівня, як правило, виконується лише в заводських умовах або за допомогою спеціалізованих сервісних програм, які застосовуються у сервісних центрах та ремонтних майстернях.

Твердотільні накопичувачі (SSD – solid-state drive)

SSD - це запам’ятовуючий пристрій для зберігання даних, у якому відсутні механічні елементи. Інформація зберігається у мікросхемах пам’яті, що забезпечує значно вищу швидкодію порівняно з жорсткими дисками (HDD).

Розрізняють два основні типи SSD:

- **SSD на базі RAM-пам’яті** - найшвидші, але дуже дорогі та потребують постійного живлення.
- **SSD на базі флеш-пам’яті** - найпоширеніші у персональних комп’ютерах; дешевші, енергонезалежні та достатньо швидкі для більшості завдань.

Функціонально SSD виконує ті самі завдання, що й HDD, але замість магнітних пластин і рухомих механізмів використовує флеш-мікросхеми.

Переваги SSD:

- дуже висока швидкість читання і запису (у десятки разів швидше за HDD);
- безшумна робота;
- низьке енергоспоживання.

Недоліки SSD:

- висока вартість (у 4–6 разів дорожчі за HDD аналогічного обсягу);
- обмежена кількість циклів перезапису, що може скорочувати термін служби при інтенсивному використанні;
- вимоги до сумісності з операційною системою (сучасні ОС, такі як Windows 7–10, оптимізовані для SSD).

Для зменшення зносу SSD доцільно використовувати **комбіновану схему SSD + HDD**: SSD - для операційної системи та програм, HDD - для зберігання даних і часто перезаписуваних файлів.

Види SSD для комп'ютерів – форм-фактори дисків

Форм-фактор (типорозмір) - це сукупність стандартів, яких необхідно дотримуватися під час проєктування та виробництва комп’ютерних комплектуючих. До таких стандартів належать геометричні розміри компонентів (довжина, ширина, висота), форма корпусу, розташування кріпильних отворів і контактних елементів. Уніфікація форм-факторів є необхідною умовою сумісності комплектуючих між собою, адже без цього підбір апаратних компонентів був би складним і трудомістким процесом.

На сьогодні існує **чотири найбільш поширені форм-фактори SSD**, які широко використовуються у персональних комп’ютерах і ноутбуках. Окрім них,

також існує кілька менш поширених або спеціалізованих стандартів (U.2, DDR-T, EDSFF, NF1), які застосовуються переважно у серверному або комерційному сегменті, або перебувають на етапі розробки та впровадження.

SSD форм-фактора 2.5" (SATA SSD)

SSD накопичувачі форм-фактора **2.5 дюйма** є класичним і найпоширенішим варіантом, особливо для ноутбуків, але також активно використовуються і в стаціонарних персональних комп'ютерах. За своїми габаритами такі SSD повністю сумісні з 2.5-дюймовими жорсткими дисками (HDD), що значно спрощує процес модернізації системи.

Підключення 2.5" SSD здійснюється через стандартний інтерфейс **SATA III**, а у випадку ноутбуків можливе встановлення через спеціальний **адаптер-перехідник замість CD/DVD-приводу (рис. 1)**. Зовні твердотільний накопичувач цього типу відрізняється від HDD лише гладким корпусом та значно меншою масою, оскільки в ньому відсутні механічні рухомі елементи.

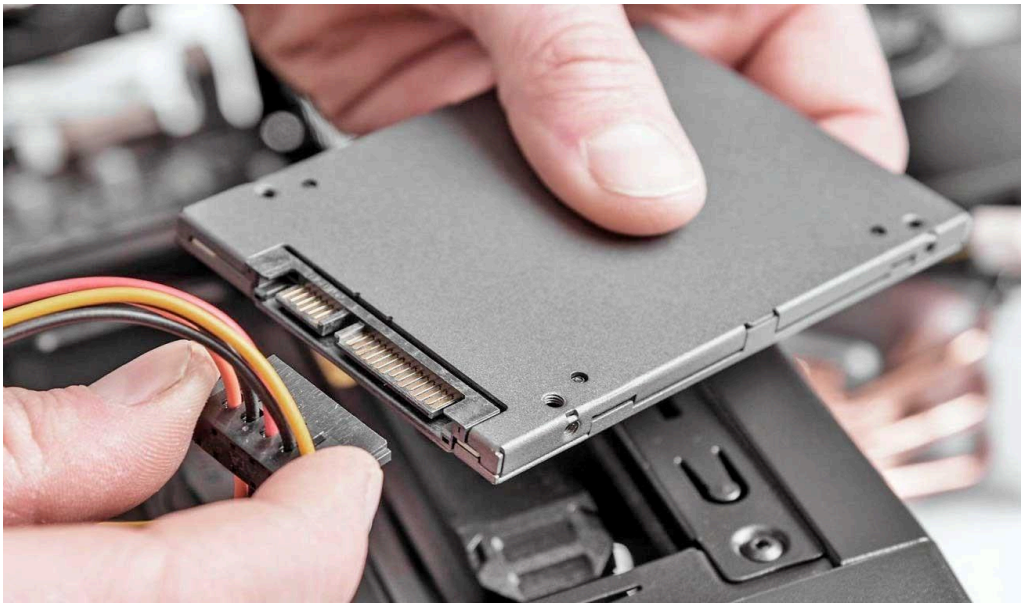


Рис. 1 - Твердотільний накопичувач SSD форм-фактора 2.5"

SSD форм-фактора M.2

SSD форм-фактора **M.2** - це компактна друкована плата з мікросхемами пам'яті, яка за своєю формою дещо нагадує модуль оперативної пам'яті (рис. 2). Встановлення SSD M.2 здійснюється безпосередньо у відповідний **роз'єм M.2 на**

материнській платі (за умови, що він передбачений конструкцією плати). Монтаж не потребує кабелів живлення чи передачі даних, що спрощує складання комп’ютера та покращує внутрішню організацію корпусу.

У випадку, якщо материнська плата не має роз’єму M.2, SSD цього типу може бути підключений **за допомогою спеціальних перехідників**:

- у відсік для стандартного 2.5-дюймового накопичувача;
- або через адаптер у слот **PCI-Express**.

Різновиди SSD M.2

У межах форм-фактора M.2 існує додатковий поділ за типом інтерфейсу підключення:

- **M.2 SATA** - використовує інтерфейс SATA і має обмеження за швидкістю, подібні до 2.5" SATA SSD;
- **M.2 NVMe (PCIe)** - працює через шину PCI-Express і забезпечує значно вищу швидкість читання та запису даних.

Рис. 2 - Твердотільний накопичувач SSD форм-фактора M.2

Накопичувачі форм-фактора M.2 також відрізняються за фізичними розмірами, які позначаються числовими індексами (наприклад, 2242, 2260, 2280, 22110) (рис. 3). У цьому маркуванні перші дві цифри означають ширину модуля (стандартно 22 мм), а наступні цифри - його довжину в міліметрах.

За габаритними розмірами ширина M.2-накопичувачів є уніфікованою і становить 22 мм, тоді як довжина може відрізнятися. Найбільш поширеними є такі варіанти:

- 2280 - 22 × 80 мм (найпопулярніший формат для настільних ПК і ноутбуків);
- 2260 - 22 × 60 мм;
- 2242 - 22 × 42 мм.

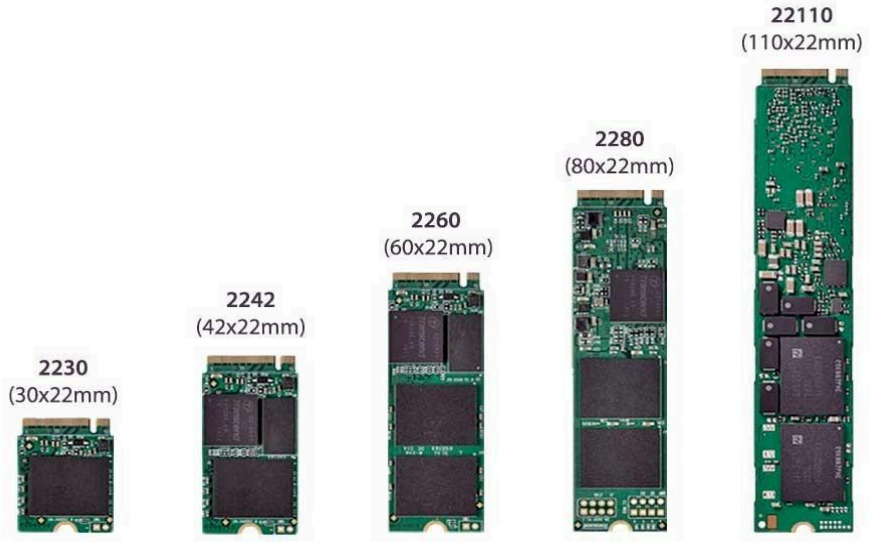


Рис. 3 – Основні типорозміри накопичувачів SSD форм-фактора M.2

За формою ключа (розташуванням вирізу на контактній смузі) накопичувачі M.2 відрізняються залежно від інтерфейсу підключення. Ключ визначає фізичну сумісність накопичувача з роз'ємом на материнській платі та підтримувану швидкість роботи (рис. 4).

Існують такі основні різновиди ключів:

- B Key - підтримує інтерфейси SATA та PCI Express x2;
- M Key - підтримує PCI Express x4 (а також SATA у деяких реалізаціях) і забезпечує найвищу швидкодію;
- B+M Key - універсальний варіант, сумісний з обома типами роз'ємів, але обмежений за швидкістю до PCI Express x2.

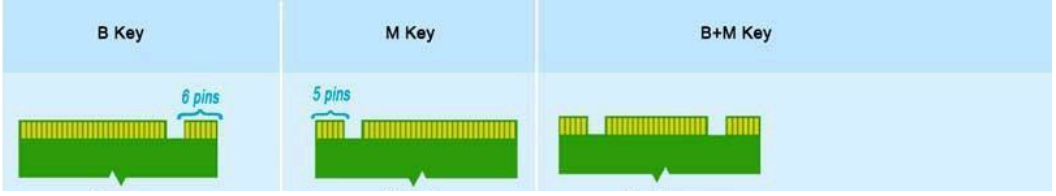


Рис. 4 – Типи ключів накопичувачів M.2

mSATA SSD (mini-SATA). Відмирає варіант, витіснений більш успішним M.2. Відрізняється відсутністю корпусу і скромним розміром в 1,8 дюйма, що робило свого часу особливо привабливим вибір SSD mSATA диска для ноутбука, і особливо – ультрабуків (рис. 5). До материнської плати підключається через спеціальний роз'єм mSATA, присутній на деяких десктопних і ноутбучних платах. За швидкістю роботи mSATA відповідає SATA III.

Рис. 5 – Твердотільний накопичувач форм-фактора mSATA

PCI-Express SSD - це високопродуктивні твердотільні накопичувачі, які встановлюються безпосередньо в слот PCI-Express материнської плати. Такі накопичувачі застосовуються лише у стаціонарних комп’ютерах, оскільки потребують повнорозмірного роз’єму розширення (рис. 6).

Через високу швидкість роботи PCI-E SSD часто оснащуються радіаторами або активним охолодженням, що необхідно для стабільної роботи при великих навантаженнях.

Форм-фактори PCI-E SSD

За габаритними розмірами PCI-Express SSD зустрічаються у кількох стандартних виконаннях:

- FHFL (Full-Height Full-Length) - повнорозмірні карти розширення (приблизно 120 мм по висоті та 312 мм по довжині);
- FHHL (Full-Height Half-Length) - половинної довжини;
- HHHL (Half-Height Half-Length) - компактні варіанти (приблизно 79,2 × 175,26 мм).

Інтерфейс підключення

Залежно від моделі та класу накопичувача PCI-E SSD можуть працювати через:

- PCI-Express x2,
- PCI-Express x4,
- PCI-Express x8.

Чим більше ліній PCI-Express використовується, тим вища пропускна здатність і швидкість обміну даними.

Рис. 6 – Твердотільний накопичувач форм-фактора PCI-Express (PCI-E SSD)

Як продовжити життя SSD

Щоб SSD служив довго, слід максимально знизити частоту запитів системи. Це можна зробити за допомогою певних настройок:

У настільній системі, для якої актуально використання режиму сну, вимкніть гібернацію. Оскільки з SSD завантаження комп'ютера відбувається з високою швидкістю, користувач не помітить різниці в темпі запуску ПК в активний режим або звичайним способом. Але якщо ноутбуком (наприклад, HP 250 G5) користуються в автономному режимі, коли живлення йде від батареї, режим сну все ж популярні.

Windows створює персональні каталоги, які зберігає на системному диску. Там же знаходяться і особисті папки користувача. І ті, і інші треба перенести на

вінчестер. Так на SSD-диску звільниться пам'ять, що прискорить його роботу. Також сповільниться процес зносу SSD.

Якщо припинити роботу файлу підкачки або налаштувати його функціонування на іншому диску, то на системному стане більше місця. Об'єм віртуальної пам'яті залежить від розміру оперативної пам'яті на вашому комп'ютері і операцій, які на цьому ПК виконуються. Якщо користувач зазвичай користується тільки браузером і месенджером, дивиться фільми, то при наявності 4 ГБ оперативної пам'яті віртуальна пам'ять понад 1 Гб на SSDдиску буде зайвою.

Купити більш місткий SSD, ніж хотіли спочатку (наприклад, SAMSUNG 2.5" 850). Вільне місце на накопичувачі не завадить, а ситуація з економією кожного Мб бачиться неприємною.

Файлова система накопичувача

Для того, щоб процесор міг швидко звернутися до будь-якої інформації, твердий диск має так звану файлову систему, де записані координати всіх файлів, що містяться на диску. Одними з поширених файлових систем є FAT-32 і NTFS.

Одним з базових понять файлової системи є кластер (блок) – мінімальна порція даних на диску. Вся інформація на диск записується блоками. Наприклад, якщо файл «важить» всього 1 байт, а розмір кластера на диску – 8 Кілобайт, то у підсумку на жорсткому диску розмір файлу буде також 8 Кілобайт (один кластер). Якщо файл займає 8,1 Кілобайт, на диску він буде «важити» всі 16 Кілобайт (два кластери).

При запису, файл розбивається на фіксовані блоки, які мають свою адресу, свій номер і відомості про те, до якого файлу вони належать.

З метою ефективного розміщення на диску, блоки не завжди знаходяться разом. Файлова система FAT-32 містить відомості про кожен файл, з кількох блоків він складається і за якими адресами вони зберігаються. Коли процесор звертається до файлової системи за певним файлом, він швидко збирається з різних блоків і обробляється вже як єдиний об’єкт.

Головним недоліком файлової системи FAT-32 є неналежна надійність збереження інформації, бо достатньо загубити чи пошкодити бодай один блок, весь файл може бути зіпсутим.

Файлова система NTFS замінила файлову систему FAT 32. Вона використовується в операційних системах Windows XP, Windows Server 2003, Windows Vista, Windows 7 и Windows 10.

NTFS використовує спеціалізовані структури даних для збереження інформації про файли, що підвищує надійність та ефективність використання дискового простору. Тут кожен файл є єдиним об’єктом.

Форматування високого рівня

При форматуванні високого рівня операційна система створює логічну структуру диска, тобто структури для роботи файлами. Простір розділу розподіляється на кластери, в кожний розділ (логічний диск) записується завантажувальний сектор тому (Volume Boot Sector – VBS), дві копії таблиці розташування файлів (FAT) і кореневий каталог (Root Directory).

За допомогою цих структур даних операційна система розподіляє дисковий простір, стежить за розташуванням та цілісністю файлів, а також «обходить» дефектні ділянки диску. Таким чином, логічне форматування не знищує повністю інформацію на диску, а тільки очищує зміст розділу та таблиці розташування файлів. Форматування високого рівня виконується командою FORMAT (MS DOS), або аналогічними командами інших операційних систем.

Особливості сучасних вінчестерів

Сучасні накопичувачі по продуктивності і можливостям значно зросли, якщо порівнювати з попередникам. Так, якщо говорити про внутрішні пристрої, то сучасним став стандартом інтерфейс SATA 3 і демонструє пропускну здатність в 6 Гбіт/с, що в два рази вище, ніж у моделей минулого покоління, в яких використовувався другий SATA.

Значно збільшено максимальний об'єм простору для зберігання даних, кеш-пам'ять, що особливо важливо при використанні магнітних носіїв в

професійних цілях, коли мова йде про значному обсязі інформації та обробка даних ведеться без зупинки.

Деякі моделі вже з заводу оптимізовані під RAID-масиви (сукупність вінчестерів). При створенні дискового масиву підвищується рівень надійності зберігання інформації, зростають показники швидкості зчитування даних і запису файлів. Якщо з якихось причин один з накопичувачів прийде в непридатність, то інформація буде знаходитися на другому жорсткому. Слід пам'ятати, що при створенні або видаленні рейду, вся інформація на HDD, що входять в масив, видаляється. З цієї причини краще заздалегідь створити резерв.

Важливо: всі диски в масиві повинні бути ідентичні щоб уникнути конфліктів комплектуючих. Також варто врахувати, що знадобиться внести зміни в налаштування БІОС, та й материнська плата повинна підтримувати можливість створення RAID.

Портативні магнітні носії також не відстають. Сумісність з більш сучасними варіантами USB - 3.0 і 3.1 - позитивно впливає на швидкодію девайса. Користуватися такими зручно: не доведеться підбирати відповідну сторону, щоб увіткнути його в роз'єм. Об'єм буфера, власне сховища теж збільшився.

Для користувачів, які ведуть активний спосіб життя, знайдуться моделі з підвищеною стійкістю до ударів. Вони оснащені цифровою панеллю і можливістю введення персонального пароля, без якого отримати доступ до даних, що зберігаються на вінчестері, неможливо. Крім того, портативні HDD можна підключати до маршрутизатора. Це дає можливість створити локальну мережу і відкрити доступ до файлів для всіх пристроїв, які знаходяться всередині цієї мережі. За рахунок такої функції можна транслювати аудіо- та відеоконтент, зберігати на гвинті гри і розважатися компанією, проходячи кооперативні онлайн-хіти. Правда, це можливо, тільки якщо роутер оснащений відповідним портом.

Програми HDDScan - https://shorturl.at/GMnUw або Google drive

NVMe (Non-Volatile Memory Express)

NVMe - це сучасний протокол доступу до твердотільних накопичувачів, розроблений спеціально для роботи з флеш-пам’яттю через високошвидкісний інтерфейс PCI-Express. На відміну від застарілих протоколів AHCI (який використовувався для HDD та SATA SSD), NVMe максимально оптимізований під паралельну обробку даних і мінімальні затримки.

Основні особливості NVMe:

- використовує PCI-Express (x4, x8) замість SATA;
- підтримує до 65 536 черг команд, у кожній з яких - до 65 536 команд;
- забезпечує надзвичайно малу затримку доступу (у кілька разів меншу, ніж SATA SSD);
- дозволяє повністю розкрити потенціал сучасних SSD.

Продуктивність NVMe:

- SATA SSD: до 550 МБ/с;
- NVMe SSD (PCIe 3.0 x4): 3000–3500 МБ/с;
- NVMe SSD (PCIe 4.0): 5000–7500 МБ/с;
- NVMe SSD (PCIe 5.0): 10 000 МБ/с і більше.

Переваги NVMe:

- висока швидкість читання і запису;
- краща робота з багатопотоковими задачами;
- швидке завантаження ОС і програм;
- ідеально підходить для ігор, 3D-графіки, відеомонтажу, серверних рішень.

Недоліки NVMe:

- вища вартість порівняно з SATA SSD;
- підвищене тепловиділення (часто потребує радіатора);
- залежність від підтримки материнської плати та процесора.

Хмарні сервіси зберігання даних

Хмарні сервіси - це модель зберігання та обробки даних, за якої інформація фізично розміщується на віддалених серверах дата-центрів, а користувач отримує доступ до неї через Інтернет.

До найпоширеніших хмарних сервісів належать:

- Google Drive
- Microsoft OneDrive
- Dropbox
- iCloud
- Amazon S3, Azure Storage (корпоративний сегмент)

Принцип роботи:

1. Дані зберігаються не на локальному ПК, а на серверах провайдера.
2. Доступ здійснюється через браузер або спеціальний клієнт.
3. Інформація автоматично синхронізується між пристроями.

Переваги хмарних сервісів:

- доступ до файлів з будь-якого пристрою;
- резервне копіювання і захист від втрати даних;
- спільна робота з файлами в реальному часі;
- відсутність необхідності купувати власні носії великого обсягу.

Недоліки хмарних сервісів:

- залежність від інтернет-з’єднання;
- обмежений безкоштовний обсяг;
- питання конфіденційності та безпеки;
- ризики, пов’язані з політикою провайдера.

Завдання для виконання роботи

1. **Аналіз накопичувачів домашнього комп’ютера**

- Визначити типи встановлених накопичувачів (HDD, SSD, NVMe).
- Зафіксувати їх основні характеристики:
    - модель;
    - обсяг пам’яті;
    - інтерфейс підключення;
    - форм-фактор.

2. **Дослідження параметрів S.M.A.R.T. жорсткого диска**

- За допомогою програми **HDDScan** визначити S.M.A.R.T.-параметри жорсткого диска.
- Виписати основні параметри S.M.A.R.T. (температура, кількість перерозподілених секторів, помилки читання тощо).
- Надати обґрунтування отриманим параметрам та зробити висновок щодо технічного стану накопичувача.

3. **Підбір накопичувача для персональної збірки**

- На основі конфігурації комп’ютера, сформованої у попередніх лабораторних роботах, підібрати оптимальний накопичувач.
- Обґрунтувати вибір за такими критеріями:
    - тип накопичувача (HDD / SSD / NVMe);
    - обсяг пам’яті;
    - інтерфейс підключення;
    - доцільність використання для поставлених задач.

4. **Дослідження хмарних сервісів зберігання даних**

- Ознайомлення з хмарним сервісом
- Обрати один хмарний сервіс зберігання даних: Google Drive, Microsoft OneDrive, Dropbox.
- Визначити та зафіксувати:
    - обсяг безкоштовного сховища;
    - спосіб доступу (веб-браузер, десктопний клієнт, мобільний застосунок);
    - можливість спільного доступу до файлів.

Лабораторне заняття №19

Інтеграція сучасних технологій введення/виведення та мережевих інтерфейсів

Теоретичні відомості

Система введення/виведення (I/O - Input/Output) - це сукупність апаратних і програмних засобів, що забезпечують обмін даними між комп'ютером і зовнішніми пристроями (рис. 1). Пристрої введення: клавіатура, миша, сканер, камера, мікрофон, сенсорні панелі. Пристрої виведення: монітор, принтер, проєктор, акустика.

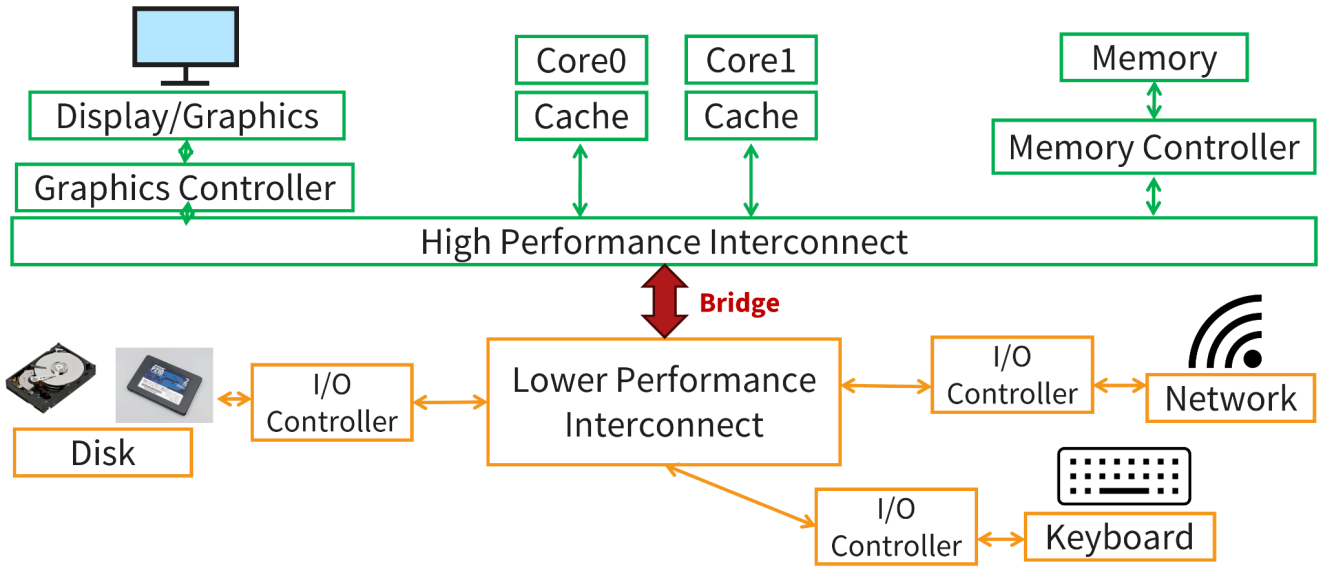


Рис. 1 - Загальна схема системи введення/виведення ПК (I/O): пристрої, інтерфейси, контролери

Інтерфейси введення/виведення умовно поділяють на:

- універсальні периферійні (USB, USB-C, Thunderbolt);
- відеоінтерфейси (HDMI, DisplayPort);
- аудіоінтерфейси (3.5 мм, S/PDIF);
- інтерфейси внутрішніх пристроїв (PCI Express, SATA, M.2 - згадується як зв'язок із попередніми роботами).

USB (Universal Serial Bus) - найпоширеніший інтерфейс периферії. Переваги: універсальність, підтримка «гарячого» підключення, можливість передавання даних і живлення.

Типові версії:

- **USB 2.0** - до 480 Мбіт/с
- **USB 3.2 Gen 1** - до 5 Гбіт/с
- **USB 3.2 Gen 2** - до 10 Гбіт/с
- **USB4** - до 40 Гбіт/с

USB Type-C - це **форма роз'єму**, а не швидкість. Через Type-C можуть працювати USB 2.0/3.x/USB4 і навіть Thunderbolt (залежить від контролера та підтримки плати) (рис. 2).

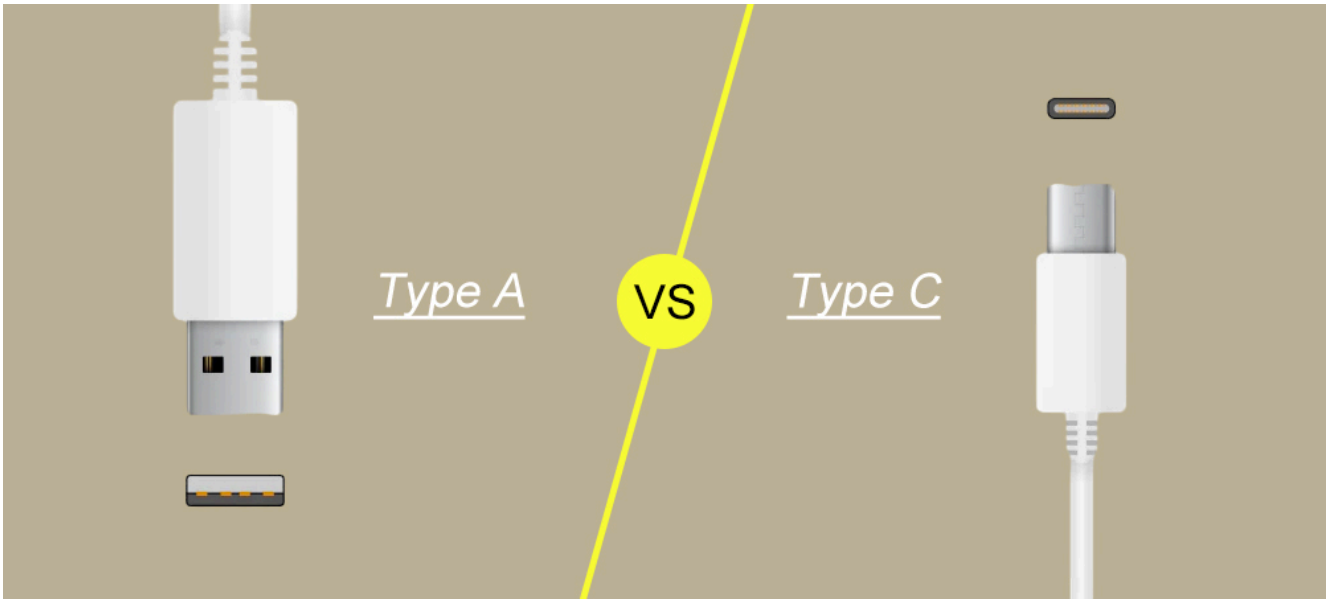


Рис. 2 - Порівняння роз'ємів USB Type-A та USB Type-C

Thunderbolt 3/4 - високошвидкісний інтерфейс, що може передавати дані, відео та живлення через один кабель (рис. 3). Типовий роз'єм - USB Type-C. Для Thunderbolt важливо:

- підтримка з боку **материнської плати/ноутбука**;
- наявність відповідного **контролера**;
- сумісність кабелів та пристроїв.

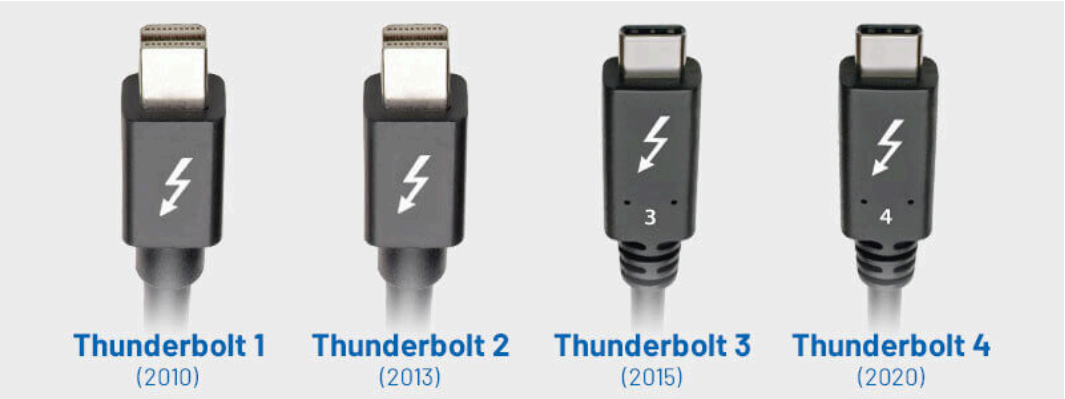


Рис. 3 - Порівняння роз'ємів Thunderbolt

HDMI і **DisplayPort** використовуються для передачі відео та аудіо (рис. 4).

- HDMI частіше зустрічається в побутових і навчальних рішеннях.
- DisplayPort поширений у професійних моніторах і багатомоніторних конфігураціях.

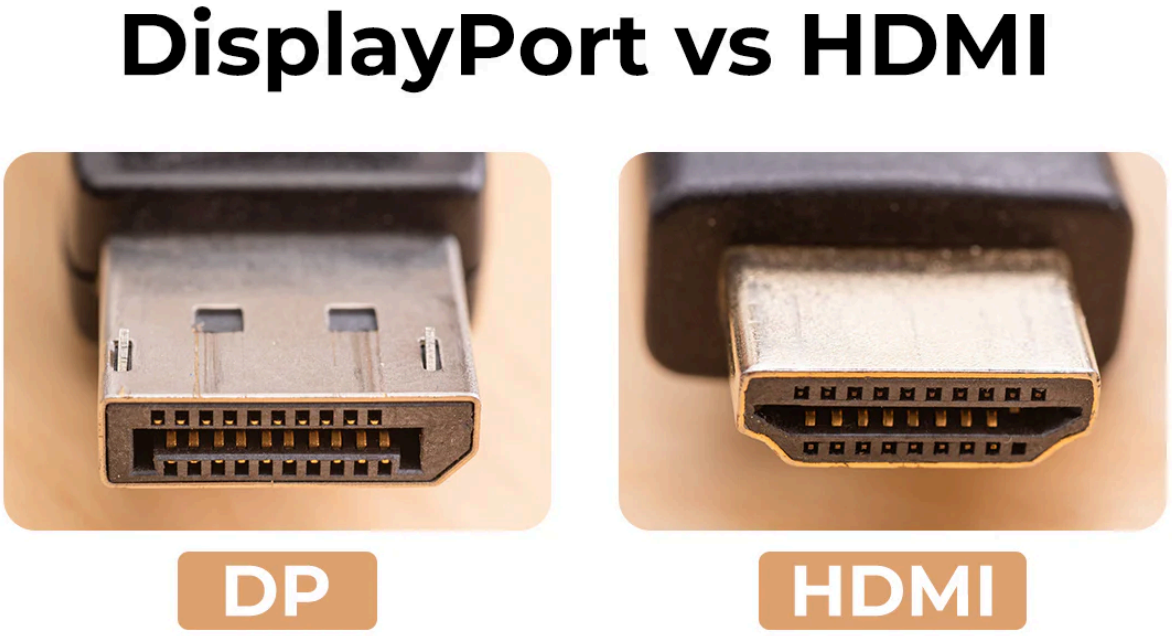


*Рис. 4 - Відеоінтерфейси HDMI та DisplayPort (приклади роз'є***мів)**

Мережеві інтерфейси

Ethernet (провідна мережа) - базовий стандарт дротових мереж (рис. 5). Найпоширеніші швидкості:

- 100 Мбіт/с (Fast Ethernet),
- 1 Гбіт/с (Gigabit),

- 2.5G / 10G Ethernet (високопродуктивні ПК, сервери).

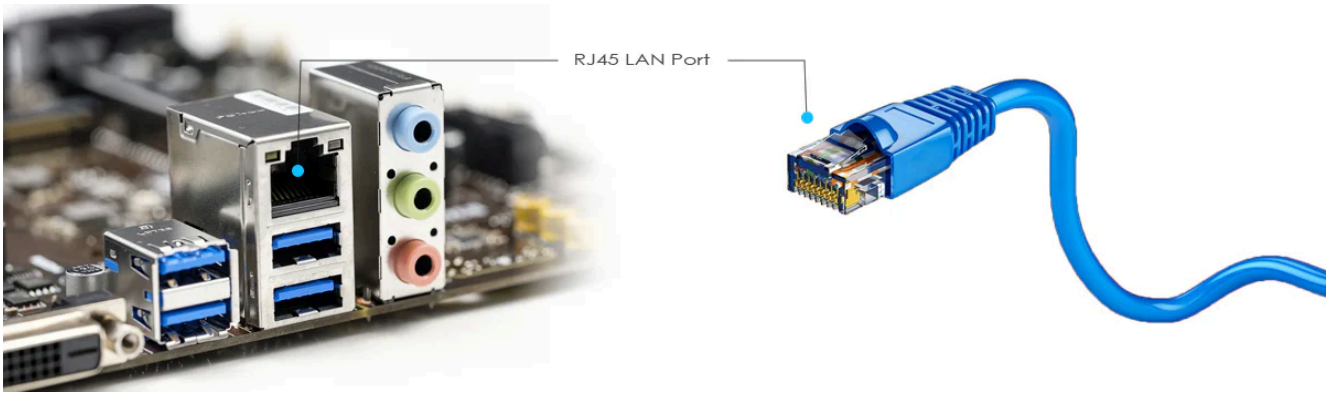


Рис. 5 - Роз'єм RJ-45 (Ethernet) на материнській платі та приклад мережевого кабелю

Wi-Fi забезпечує бездротовий доступ до мережі. Сучасні стандарти:

- Wi-Fi 5 (802.11ac),
- Wi-Fi 6/6E (802.11ax),
- Wi-Fi 7 (802.11be).

Якість Wi-Fi залежить від стандарту, ширини каналу, перешкод, відстані та якості антени/адаптера.

Bluetooth використовується для підключення периферії малого радіуса дії (миша, клавіатура, навушники). Типові версії: 5.0–5.3. Переваги: низьке енергоспоживання, зручність. Обмеження: дальність і швидкість (порівняно з Wi-Fi).

Інтеграція інтерфейсів у сучасних ПК

Інтеграція інтерфейсів у сучасному ПК означає, що переважна більшість портів введення/виведення та мережевих функцій реалізована "на борту" материнської плати (або в ноутбуках - на системній платі), а не у вигляді окремих плат розширення. Це зменшує вартість, спрощує складання, підвищує надійність та енергоефективність.

Рівні інтеграції: У сучасній архітектурі ПК інтерфейси розподілені між кількома вузлами:

Процесор (CPU)

- Має вбудований **контролер пам'яті** (IMC).
- Надає частину **ліній PCI Express** (особливо для відеокарти і/або NVMe).
- У деяких платформах частково бере участь у маршрутизації високошвидкісних шин.

Чіпсет (PCH / Chipset)

- Це "концентратор" більшості периферійних інтерфейсів:
  - USB (2.0/3.x/частково USB-C логіка),
  - SATA,
  - додаткові PCIe-лінії для периферії,
  - інколи частина мережевих функцій.
- Зв'язується з CPU через швидкісний канал (наприклад DMI в Intel-платформах).

Окремі контролери на материнській платі

Часто виробник плати додає спеціалізовані мікросхеми:

- **USB-контролери** (для додаткових портів або кращих режимів),
- **Thunderbolt/USB4 контролер** (якщо підтримується),
- **мережевий контролер** (LAN),
- **Wi-Fi/Bluetooth модуль** (часто у вигляді M.2 E-key),
- **аудіокодек** (Realtek/аналогічні),
- контролер підсвітки, датчики, fan-controller тощо.

Роз'єми/фізична частина (connectors) Навіть якщо контролер є, потрібна фізична реалізація:

- порти на задній панелі,
- внутрішні колодки (headers) для передньої панелі корпуса,
- розводка доріжок і дотримання вимог сигналу (особливо для USB 3.x/USB-C/DP).

Завдання для виконання роботи

1. Складіть перелік наявних портів на вашому ПК
2. На основі зібраних даних коротко поясніть:

- які інтерфейси, імовірно, реалізовані через **чіпсет** (типово USB/SATA/частина PCIe);
- які залежать від **CPU** (лінії PCIe для GPU/NVMe, iGPU-відеовиходи);
- які забезпечуються **окремими контролерами** (Thunderbolt/USB4, LAN 2.5G/10G, Wi-Fi/BT модуль, аудіокодек).

ЛІТЕРАТУРА

1. Tanenbaum Andrew S. Computerarchitektur - Strukturen, Konzepte, Grundlagen, 6th Edition. Pearson Education, 2016. 829 p.
2. Tanenbaum A. S. Structured computer organization. Todd Austin. – 6th ed. 2018. 801p.
3. Тарарака В.Д. Архітектура комп'ютерних систем: навчальний посібник. Житомир : ЖДТУ, 2018. 383 с.
4. Борисенко О. А. Цифрова схемотехніка : підручник. Суми : Сумський державний університет, 2016. 200 с.
5. Матвієнко М.П. Архітектура комп'ютера. Навчальний посібник. / Матвієнко М.П., Розен В.П. Закладний О.М. Київ : Видавництво Ліра-К. 2016. 256 с.
6. Матвієнко М. П., Розен В. П. Комп’ютерна схемотехніка. Навчальний посібник. Київ : Видавництво Ліра-К, 2016. 192 с.
7. Мельник А. О. Архітектура комп'ютера: підруч. для студ. вузів. 3-вид. Луцьк : Волинська обласна друкарня, 2018. 470 с.
8. Рябенький В.М. Жуйков В.Я. Ямненко Ю.С. Заграничний А.В. „Схемотехніка: Пристрої цифрової електроніки‖. Електронний підручник для вищих навчальних закладів, 2016 р.
9. Тонкошкур О.С. Архітектура комп’ютерів. Машинні команди та програмування на асемблері: навчальний посібник / О.С. Тонкошкур, О.Б. Гниленко, Н.О. Матвєєва, О.С. Морозов. Дніпро : Вид-во «Нова Ідеологія», 2018. 179 с.
10. Тотосько О.В., Стухляк П.Д. Цифрова обробка сигналів та зображень: навчальний посібникдля студентів спеціальності 151 «Автоматизація та комп’ютерно-інтегровані технології» – Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя , 2016. –140с
11. Computer Architecture / Charles Fox., No Starch Press, 2024
12. Code: The Hidden Language of Computer Hardware and Software / Charles Petzold., Microsoft Press, 2023

13. Саварин, П., Редько, О., Редько, Р., & Великий, О. (2024). ЛЮДИНО-КОМП'ЮТЕРНА ВЗАЄМОДІЯ НА ОСНОВІ ARDUINO. Automation of Technological and Business Processes, 15(4), 98-105. https://doi.org/10.15673/atbp.v15i4.2724
14. P. Savaryn, V. Strekha, M. Brych, L. Brych, V. Kabak and M. Polishchuk, «The Original Method of Controlling a Computer Using Distance Sensors», 2022 IEEE 16th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), 2022, pp. 683-688, doi: 10.1109/TCSET55632.2022.9767011.

Кодування інформації та архітектура комп’ютера : Методичні вказівки до лабораторних занять для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Професійна освіта (комп’ютерні технології)» галузі знань А Освіта спеціальності А5.39 Професійна освіта (Цифрові технології) денної та заочної форм навчання / уклад. А. ЛОБАЦЬКИЙ, П. САВАРИН. Луцьк: ЛНТУ, 2026. 195 с.

Комп’ютерний набір А. ЛОБАЦЬКИЙ
Редактор А. ЛОБАЦЬКИЙ

Підп. до друку «__»_____2026 р. Формат 60x84/16. Папір офс.
Гарн. Таймс. Ум. друк. арк. 12.
Тираж 50 прим.

Луцький національний технічний університет
43018, м. Луцьк, вул. Львівська, 75