

## **Зміст**

<b>Зміст</b>	<b>1</b>
<b>Розділ 1. Комп'ютер, алгебра булевих функцій</b>	<b>2</b>
1.1. Комп'ютерна архітектура	2
1.2 Алгебра булевих функцій	5
1.3 Алгебри перемикальних функцій	12
1.4 Мінімізація перемикальних функцій	17
<b>Розділ 2. Типові вузли цифрових ЕОМ</b>	<b>21</b>
2.1 Тригери. Регістри. Лічильники	21
2.2 Шифратори. Дешифратори	34
2.3 Перетворювачі кодів	41
<b>Розділ 3. Комп'ютерна арифметика</b>	<b>46</b>
3.1 Кодування від'ємних чисел в ЕОМ	46
3.2 Форми представлення чисел у ЕОМ	46
3.3 Додавання та віднімання чисел	49
3.4 Зсуви в машинних кодах	52
3.5 Множення чисел	56
<b>Розділ 4. Мікроконтролери</b>	<b>61</b>
4.1 MSP	61
4.2 Arduino	64

## **Розділ 1. Комп'ютер, алгебра булевих функцій**

### **1.1. Комп'ютерна архітектура**

#### **Розвиток комп'ютерної архітектури**

Нульове покоління (1942 - 1945) - механічні комп'ютери

Перше покоління (1945 - 1955) - електронні лампи

Друге покоління (1955 - 1965) - транзистори

Третє покоління (1965 - 1980) - інтегральні схеми

Четверте покоління (1980 - 1990) - великі інтегральні схеми, мікропроцесори

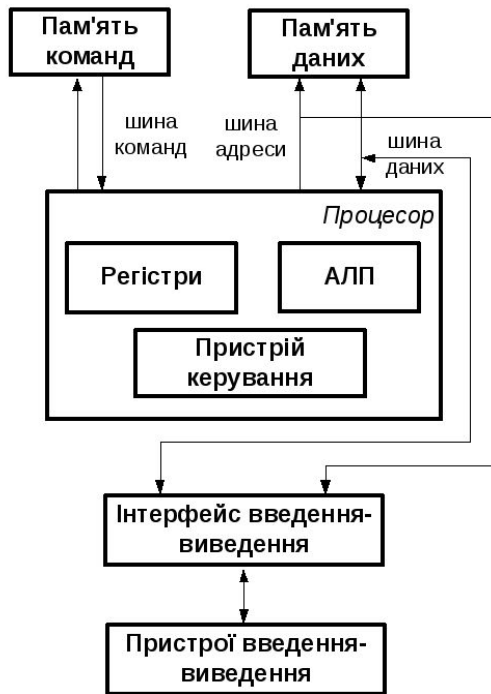
П'яте покоління (з 1990) - AI, нейрокомп'ютери

#### **Типи архітектур**

- 1) Гарвардська архітектура (команда та дані зберігаються в різних пристроях пам'яті)
- 2) Архітектура фон Неймана (команда та дані зберігаються в одній і тій самій пам'яті)

#### **Гарвардська архітектура**

**Гарвардська архітектура** - це архітектура, особливістю якої є фізичне розділення пам'яті на пам'ять команд (програм) і пам'ять даних. Тим самим розділяються шини передачі керуючої і оброблюваної інформації. При цьому підвищується продуктивність комп'ютера за рахунок суміщення в часі пересилання та обробки даних і команд. Гарвардська архітектура вперше була реалізована **Ховардом Айкеном** в ЕОМ Марк-1 у Гарварді.



Переваги Гарвардської архітектури:

- застосування невеликих за обсягом пам'яті даних сприяє прискоренню пошуку інформації в пам'яті і збільшує швидкодію мікропроцесора;
- наявність окремих шини даних і шини команд також дозволяє підвищити швидкодію мікропроцесора;
- з'являється можливість організувати паралельне виконання програм (кожна пам'ять з'єднується з процесором окремою шиною, що дозволяє одночасно з читанням/записом даних при виконанні поточної команди зробити вибірку і декодування наступної команди).

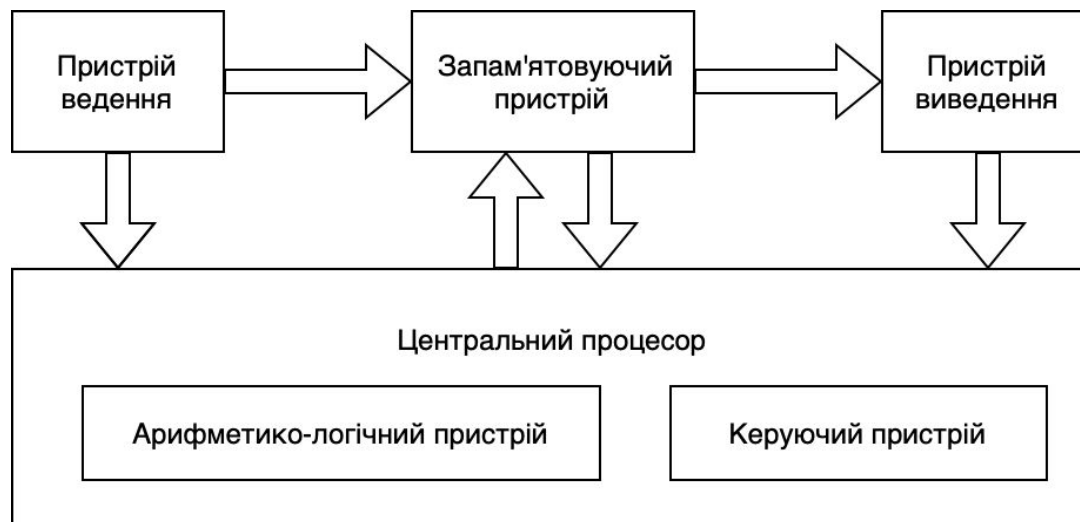
Недоліки гарвардської архітектури:

- ускладнення конструкції через використання окремих шин для команд і даних;
- фіксований обсяг пам'яті для команд і даних;
- збільшення загального обсягу пам'яті через неможливість її оптимального перерозподілу між командами і даними
- необхідність генерації додаткових керуючих сигналів для пам'яті команд і пам'яті даних.

## Схема архітектури фон Неймана

Фон Нейман виділив п'ять базових елементів комп'ютера:

- арифметико-логічний пристрій
- керуючий пристрій, який організовує виконання програми
- запам'ятовуючий пристрій
- пристрої для введення інформації
- пристрої для виведення інформації



**Арифметико-логічний пристрій** - пристрій, який виконує вказані командами операції над вказаними даними.

**Керуючий пристрій (КП)** керує всіма частинами комп'ютера.

**Запам'ятовуючий пристрій** призначений для тимчасового (оперативна пам'ять) та тривалого (постійна пам'ять) зберігання програм, вхідних і результатуючих даних та деяких проміжних результатів.

Пристрої введення-виведення, а також пристрої для попередньої підготовки інформації та її зберігання прийнято називати **периферійними пристроями**.

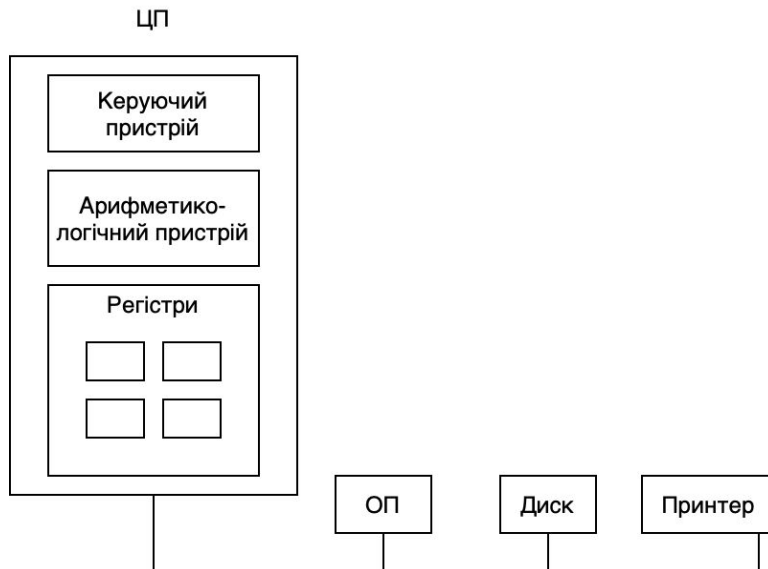
Основні принципи архітектури фон Неймана:

- двійкова система для кодування інформації в комп'ютері;
- програми і дані зберігаються в загальній пам'яті разом;
- програма, яка визначає дії комп'ютера, являє собою послідовність елементарних команд;
- адресація пам'яті;

- програмне керування роботою комп'ютера.

**Цифрові електронні обчислювальні машини (ЕОМ) або комп'ютери** – це система з програмним керуванням призначені для автоматизації оброблення цифрової інформації.

### Структурна схема комп'ютера



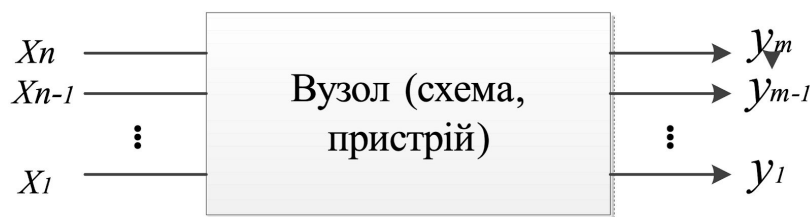
**Регістри** - високошвидкісна пам'ять невеликої ємності, використання якої дозволяє прискорити обмін інформацією між основною пам'яттю і процесором.

## 1.2 Алгебра булевих функцій

### Логічні елементи

Логічний елемент – це електронний пристрій, що реалізує одну з логічних операцій. Логічні елементи являють собою електронні пристрої, у яких оброблювана інформація закодована у вигляді двійкових чисел, відображуваних напругою (сигналом) високого і низького рівня.

На логічному та функціональному рівнях опису комп'ютер є системою, до складу якої входять відповідні функціональні вузли, блоки та пристрої.



Функція називається **перемикальною (логічною, булевою)**, якщо сама функція і кожен з її аргументів можуть набувати лише двох значень: 0 або 1.

Система перемикальних функцій:

$$\begin{cases} y_1 = f_1(x_n, x_{n-1}, \dots, x_1), \\ y_2 = f_2(x_n, x_{n-1}, \dots, x_1), \\ y_m = f_m(x_n, x_{n-1}, \dots, x_1). \end{cases}$$

Перемикальну функцію можна задати:

- таблицею істинності,
- словесним описом (вербально),
- геометричним зображенням,
- аналітично (формулою).

Номер набору	Набір значень аргументів			Значення функції у
	$x_3$	$x_2$	$x_1$	
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

Таблиця істинності перемикальної функції  $y = f(x_3, x_2, x_1)$

**Набором (кортежем)** називають впорядковану послідовність значень аргументів.

Перемикальну функцію, що залежить від  $n$  змінних, називають  **$n$ -місною**. Перемикальна функція є **повністю визначеною**, якщо відомі (задані) її значення на всіх без винятку наборах (кортежах) змінних (аргументів). Якщо перемикальна функція задана не на всіх наборах змінних, то її називають **неповністю визначеною функцією**.

Найпростіше і найкомпактніше є аналітичне (формульне) задання перемикальної функції, яке ґрунтується на використанні логічних операцій. Наприклад, перемикальну функцію  $y = f(x_3, x_2, x_1)$  від трьох змінних, якій відповідає таблиця істинності табл. 1.1, можна задати формулою:

$$f(x_3, x_2, x_1) = x_3 \overline{x_2} \vee x_3 \overline{x_1} \vee \overline{x_3} x_2 x_1$$

$$f(x_3, x_2, x_1) = (x_3 \vee x_2)(x_3 \vee x_1)(\overline{x_3} \vee \overline{x_2} \vee \overline{x_1})$$



Найпростішими є перемикальні функції від однієї та двох змінних.

$x$	$y_0 = f_0(x)$	$y_1 = f_1(x)$	$y_2 = f_2(x)$	$y_3 = f_3(x)$
0	0	0	1	1
1	0	1	0	1

Перемикальні функції від однієї змінної

Функція Константа нуль,  $y_0 = f_0(x) = 0$ , та функція Константа одиниця,  $y_3 = f_3(x) = 1$ , не залежать від значень аргументу, і тому називаються виродженими перемикальними функціями. Невиродженими перемикальними функціями є функція Повторення (функція ТАК),  $y_1 = f_1(x)$ , та функція Заперечення (функція Інверсія, функція НЕ),  $y_2 = f_2(x)$ .

На кресленнях (рисунках) логічний елемент зображають у вигляді умовного графічного позначення (УГП).



Табл. істинності	Назва ф-ї	Запис	УГП логічного елемента	Назва логічного елемента						
<table><tr><td>х</td><td>у</td></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	х	у	0	0	1	1	Повторення, ТАК	$y=x$		Повторювач
х	у									
0	0									
1	1									
<table><tr><td>х</td><td>у</td></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	х	у	0	1	1	0	Інверсія, заперечення, НІ	$y=\overline{x}$		Інвертор
х	у									
0	1									
1	0									



Невироджені перемикальні функції від однієї змінної


Можливі перемикальні функції від двох змінних, загальна кількість яких – 16.

Табл. істинності	Назва ф-ї	Запис	УГП логічного елемента	Назва логічного елемента
---------------------	-----------	-------	---------------------------	--------------------------------



<table><tr><th>X<sub>2</sub></th><th>X<sub>1</sub></th><th>y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	X <sub>2</sub>	X <sub>1</sub>	y	0	0	0	0	1	0	1	0	0	1	1	1	Кон'юнкція , І, логічний добуток	$y = x_2 x_1$ $y = x_2 \cdot x_1$ $y = x_2 \& x_1$		Елемент І, кон'юнкто р
X <sub>2</sub>	X <sub>1</sub>	y																	
0	0	0																	
0	1	0																	
1	0	0																	
1	1	1																	
<table><tr><th>X<sub>2</sub></th><th>X<sub>1</sub></th><th>y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	X <sub>2</sub>	X <sub>1</sub>	y	0	0	0	0	1	0	1	0	0	1	1	1	Диз'юнкція , АБО, логічний сума	$y = x_2 \vee x_1$ $y = x_2 + x_1$		Елемент АБО, диз'юнкто р
X <sub>2</sub>	X <sub>1</sub>	y																	
0	0	0																	
0	1	0																	
1	0	0																	
1	1	1																	

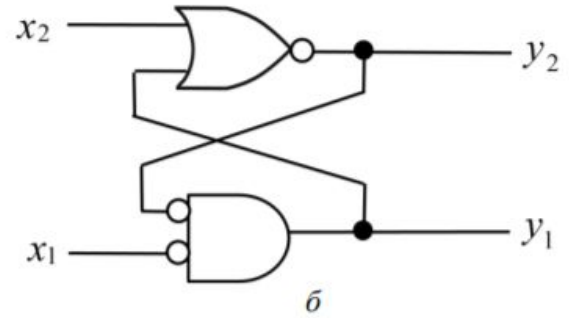
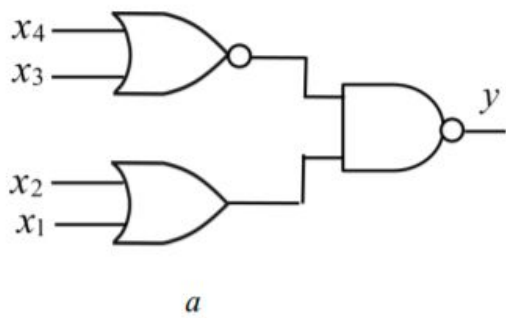
<table><tr><th>X<sub>2</sub></th><th>X<sub>1</sub></th><th>y</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	X <sub>2</sub>	X <sub>1</sub>	y	0	0	1	0	1	1	1	0	1	1	1	0	I-НЕ,  φ-я Шефера	$y = \overline{x_2 x_1}$ $y = \overline{x_2 \cdot x_1}$ $y = \overline{x_2 \& x_1}$		Элемент  I-НЕ,  элемент Шефера
X <sub>2</sub>	X <sub>1</sub>	y																	
0	0	1																	
0	1	1																	
1	0	1																	
1	1	0																	
<table><tr><th>X<sub>2</sub></th><th>X<sub>1</sub></th><th>y</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	X <sub>2</sub>	X <sub>1</sub>	y	0	0	1	0	1	0	1	0	0	1	1	0	АБО-НЕ,  φ-я Пірса	$y = \overline{x_2 \vee x_1}$ $y = \overline{x_2 + x_1}$		Элемент АБО-НЕ,  элемент Пірса
X <sub>2</sub>	X <sub>1</sub>	y																	
0	0	1																	
0	1	0																	
1	0	0																	
1	1	0																	

<table><tr><th>X<sub>2</sub></th><th>X<sub>1</sub></th><th>y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	X <sub>2</sub>	X <sub>1</sub>	y	0	0	0	0	1	1	1	0	1	1	1	0	ВИКЛЮЧНЕ АБО, сума по модулю два	$y = x_2 \oplus x_1$  $y = x_2 \neq x_1$		Елемент ВИКЛЮЧНЕ АБО
X <sub>2</sub>	X <sub>1</sub>	y																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	0																	

Функції двох аргументів

Розрізняють два види логічних схем – комбінаційні схеми та послідовнісні схеми. Якщо сукупність вихідних сигналів логічної схеми з  $n$  входами і  $m$  виходами в даний момент часу повністю визначається сукупністю вхідних сигналів у цей самий момент часу і не залежить від вхідних сигналів, що діяли на входах в попередні моменти часу, то таку логічну схему називають комбінаційною схемою.

Якщо сукупність вихідних сигналів логічної схеми в даний момент часу залежить не лише від вхідних сигналів, що діють у цей самий момент часу, але й від стану, в якому перебуває схема (тобто й від сигналів, що діяли в попередній момент часу), то таку логічну схему називають послідовнісною схемою або схемою з пам'яттю.



Види логічних схем:

а) - комбінаційна схема; б) - послідовна схема (схема з пам'яттю )

Як комбінаційні, так і послідовні схеми називають цифровими автоматами. При цьому комбінаційні схеми є тривіальними автоматами (тобто автоматами, що не мають пам'яті), а послідовні схеми є автоматами з пам'яттю.

### 1.3 Алгебри перемикальних функцій

**4 алгебри: Буля, Шеффера, Пірса, Жегалкіна.** Усі перелічені алгебри ґрунтуються на одній і тій самій множині дискретних елементів  $B = \{0, 1\}$ , але використовують різні множини  $\Omega$  логічних операцій на множині елементів.

#### Алгебра Буля

Алгебру Буля використовують в теорії множин та теорії ймовірностей. Для перетворення аргументів (змінних) та виразів в алгебрі Буля використовують 3 логічні операції (функції) – І, АБО, НЕ (AND, OR, NOT). Операція НЕ – одномісна, операції І, АБО – n-місні.

У загальному випадку систему (множину) операцій (функцій) в алгебрі Буля можна записати у вигляді:

$$\left\{ \begin{array}{l} f_1 = x_n * x_{n-1} * \dots * x_1 \\ f_2 = x_n \vee x_{n-1} \vee \dots \vee x_1 \\ f_3 = \bar{x}_i, i = 1, 2, \dots, n \end{array} \right. \quad \begin{array}{ll} \text{(AND)} & \text{І} \\ \text{(OR)} & \text{АБО} \\ \text{(NOT)} & \text{НЕ} \end{array}$$

Алгебра Буля ґрунтується на наступних аксіомах:

$x * 0 = 0$	$x \vee 0 = x$	$\overline{\overline{x}} = x$
$x * 1 = x$	$x \vee 1 = 1$	$\overline{0} = 1$
$x * x = x$	$x \vee x = x$	$\overline{1} = 0$
$x * \overline{x} = 0$	$x \vee \overline{x} = 1$	

Основні закони (властивості) алгебри Буля:

1. Закон комутативності (переставний закон):

$$x_2 * x_1 = x_1 * x_2$$

$$x_2 \vee x_1 = x_1 \vee x_2$$

2. Закон асоціативності (сполучний закон):

$$x_3 x_2 x_1 = (x_3 x_2) x_1$$

$$x_3 \vee x_2 \vee x_1 = (x_3 \vee x_2) \vee x_1$$

3. Закон дистрибутивності (розподільний закон):

для кон'юнкції відносно диз'юнкції:  $x_3(x_2 \vee x_1) = x_3 x_2 \vee x_3 x_1$

для диз'юнкції відносно кон'юнкції:  $x_3 \vee (x_2 x_1) = (x_3 \vee x_2)(x_3 \vee x_1)$

4. Закон абсорбції (поглинання):

$$x_2(x_2 \vee x_1) = x_2$$

$$x_2 \vee x_2 x_1 = x_2$$

5. Закон склеювання:

$$x_2 x_1 \vee x_2 \overline{x}_1 = x_2$$

$$(x_2 \vee x_1)(x_2 \vee \overline{x}_1) = x_2$$

6. Закони (правила) де Морґана (De Morgan's laws) встановлюють зв'язок між функціями (операціями) І, АБО, І-НЕ, АБО-НЕ:

- |   |  |
|---|--|
| 1) $\overline{x_2 \vee x_1} = \overline{x_2} * \overline{x_1}$<br>( <i>NOR</i> $\rightarrow$ <i>AND</i> ) | - заперечення диз'юнкції еквівалентне<br>кон'юнкції заперечень |
| 2) $\overline{x_2 * x_1} = \overline{x_2} \vee \overline{x_1}$<br>( <i>NAND</i> $\rightarrow$ <i>OR</i> ) | - заперечення кон'юнкції еквівалентне<br>диз'юнкції заперечень |
| 3) $x_2 \vee x_1 = \overline{\overline{x_2} * \overline{x_1}}$  | - диз'юнкція еквівалентна кон'юнкції                           |

$(OR \rightarrow NAND)$

заперечень із загальним запереченням

4)  $x_2 x_1 = \overline{\overline{x_1} \vee \overline{x_1}}$

- кон'юнкція еквівалентна диз'юнкції

$(AND \rightarrow NOR)$

заперечень із загальним запереченням

Правила де Моргана допускають узагальнення на випадок довільної кількості змінних:

$$\overline{x_n \vee \dots \vee x_1} = \overline{x_n} * \dots * \overline{x_1},$$

$$\overline{x_n * \dots * x_1} = \overline{x_n} \vee \dots \vee \overline{x_1},$$

$$x_n \vee \dots \vee x_1 = \overline{\overline{x_n} * \dots * \overline{x_1}},$$

$$x_n * \dots * x_1 = \overline{\overline{x_n} \vee \dots \vee \overline{x_1}},$$

*Наслідок правил де Моргана:*

елемент АБО можна замінити на елемент І-НЕ з інверсними входами (і навпаки), а елемент І – на елемент АБО-НЕ з інверсними входами (і навпаки).

При виконанні перетворень в аналітичних виразах булевої алгебри необхідно дотримуватись пріоритету операцій:

1. Операції в дужках
2. Операція заперечення
3. Кон'юнкція
4. Диз'юнкція

Пріоритет операцій слід враховувати при застосуванні правил де Моргана.

### Канонічні форми в алгебрі Буля

В алгебрі Буля будь-яку перемикальну функцію можна подати (записати) у двох канонічних формах:

- у ДДНФ – досконалій диз'юнктивній нормальній формі
- у ДКНФ – досконалій кон'юнктивній нормальній формі.

Мінтермом називають кон'юнктивний терм, до складу якого входять всі без винятку змінні функції (змінна може бути прямою або інверсною). Так, для функції від 3-х змінних мінтермами є, наприклад,  $x_3 x_2 x_1$ ,  $\overline{x_3} x_2 x_1$ , а вирази (терми)  $x_2 x_3$ ,  $\overline{x_3} x_1$ ,  $\overline{x_2}$  мінтермами не є.

Диз'юнктивною нормальною формою називають диз'юнкцію мінтермів.

Наприклад, вираз  $f(x_4, x_3, x_2, x_1) = x_4 x_3 x_2 x_1 \vee \overline{x_4} x_3 \overline{x_2} x_1 \vee x_4 x_3 x_2 \overline{x_1}$  є нормальною диз'юнктивною формою, оскільки до складу кожного з термів входять всі без винятку змінні функції.

Досконала диз'юнктивна нормальна форма функції (ДДНФ) – це диз'юнкція конститuent одиниці (мінтермів), що відповідають наборам, на яких перемикальна функція набуває значення 1

ДДНФ – називають також формою І/АБО (AND/OR), підкреслюючи що в ДДНФ внутрішньою функцією є І, а зовнішньою – АБО.

Макстермом називають диз'юнктивний терм, до складу якого входять всі без винятку змінні функції (змінна може бути прямою або з інверсією). Для функції від 4-х змінних макстермами є, наприклад,  $x_4 \vee \overline{x_3} \vee \overline{x_2} \vee x_1$ ,  $x_4 \vee x_3 \vee x_2 \vee x_1$ , а вирази (терми)  $x_4 \vee \overline{x_2} \vee x_1$ ,  $\overline{x_4} \vee \overline{x_1}$  не є макстермами.

Кон'юнктивною нормальною формою називають кон'юнкцію макстермів. Наприклад, вираз  $f(x_3, x_2, x_1) = (x_3 \vee x_2 \vee x_1)(\overline{x_3} \vee \overline{x_2} \vee \overline{x_1})$  є нормальною кон'юнктивною формою, оскільки до складу кожного з термів входять всі без винятку змінні функції.

Досконала кон'юнктивна нормальна форма функції (ДКНФ) – це кон'юнкція контитuent нуля (макстермів), що відповідають наборам, на яких перемикальна функція набуває значення 0.

ДКНФ називають також формою АБО/І (OR/AND), де АБО – внутрішня функція, а І – зовнішня.

### Алгебра Шеффера

Для перетворення виразів в алгебрі Шеффера використовують лише одну логічну операцію – операцію І-НЕ (штрих Шеффера, функція Шеффера).

$$f = \overline{x_n \cdot x_{n-1} \cdot \dots \cdot x_1} = x_n / x_{n-1} / \dots / x_1$$

Аксиоми алгебри Шеффера:

$$\overline{x \cdot 0} = x/0 = 1,$$

$$\overline{\overline{x} \cdot \overline{x}} = x/x = \overline{\overline{x}},$$

$$\overline{x \cdot 1} = x/1 = \overline{x},$$

$$\overline{\overline{x} \cdot \overline{\overline{x}}} = x/\overline{\overline{x}} = 1$$

В алгебрі Шеффера виконується лише закон (властивість) комутативності (переставний закон):

$$\overline{x_2 x_1} = \overline{x_1 x_2},$$

$$x_2 / x_1 = x_1 / x_2$$

## Алгебра Пірса

Алгебра Пірса ґрунтується на застосуванні лише однієї логічної операції – АБО-НЕ (стрілка Пірса, функція Пірса).

$$f = \overline{x_n \vee x_{n-1} \vee \dots \vee x_1} = x_n \downarrow x_{n-1} \downarrow \dots \downarrow x_1$$

Аксіоми алгебри Пірса:

$$\overline{x \vee 0} = x \downarrow 0 = \bar{x},$$

$$\overline{x \vee x} = x \downarrow x = \bar{x},$$

$$\overline{x \vee 1} = x \downarrow 1 = 0,$$

$$\overline{x \vee \bar{x}} = x \downarrow \bar{x} = 0$$

В алгебрі Пірса виконується лише закон (властивість) комутативності (переставний закон):

$$\overline{x_2 \vee x_1} = \overline{x_1 \vee x_2},$$

$$x_2 \downarrow x_1 = x_1 \downarrow x_2$$

## Алгебра Жегалкіна

Алгебра Жегалкіна визначена на множині елементів  $B = \{0, 1\}$  та ґрунтується на множині логічних операцій  $\Omega = \{\&, \oplus, 1\}$  – кон'юнкція, додавання за модулем два та константа 1:

$$\begin{cases} f_1 = x_n \cdot x_{n-1} \cdot \dots \cdot x_1, & \text{I (AND),} \\ f_2 = x_n \oplus x_{n-1} \oplus \dots \oplus x_1, & \text{mod2 (XOR),} \\ f_3 = 1. \end{cases}$$

Операція кон'юнкції має вищий пріоритет, ніж операція додавання за модулем два.

Аксіоми алгебри Жегалкіна:

$$x \cdot 0 = 0,$$

$$x \oplus 0 = x,$$

$$x \cdot 1 = x,$$

$$x \oplus 1 = \bar{x},$$

$$x \cdot x = x,$$

$$x \oplus x = 0,$$

$$x \cdot \bar{x} = 0,$$

$$x \oplus \bar{x} = 1.$$

Розглянемо основні закони (властивості) алгебри Жегалкіна.

1. Закон комутативності (переставний закон):

$$x_2 \cdot x_1 = x_1 \cdot x_2$$

$$x_2 \oplus x_1 = x_1 \oplus x_2$$



2. Закон асоціативності (сполучний закон):

$$x_3 x_2 x_1 = (x_3 x_2) x_1$$

$$x_3 \oplus x_2 \oplus x_1 = (x_3 \oplus x_2) \oplus x_1$$

3. Властивість дистрибутивності (розподільний закон):

$$x_3 (x_2 \oplus x_1) = x_3 x_2 \oplus x_3 x_1$$

## 1.4 Мінімізація перемикальних функцій

Графічні методи мінімізації перемикальних функцій

Розрізняють два графічні методи мінімізації перемикальних функцій:

- метод мінімізації на основі діаграм Вейча;
- метод мінімізації на основі карт Карно.

Перевагами обох методів графічної мінімізації є:

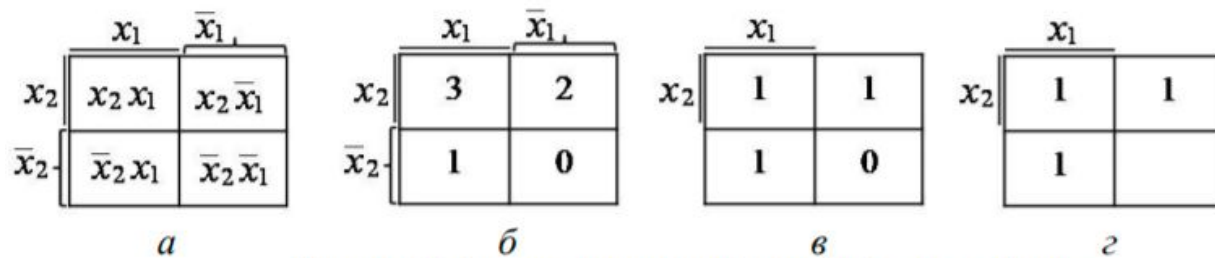
1) наочність – для отримання мінімальної форми функції використовується лише одна таблиця;

2) простота та швидкість отримання результату – мінімальну форму функції отримують з таблиці одразу, минаючи етап знаходження скороченої ДНФ та етап визначення тупикових ДНФ функції;

3) з таблиці можна отримати як мінімальну ДНФ, так і мінімальну КНФ функції.

Розглянемо метод мінімізації перемикальних функцій на основі діаграм Вейча. Діаграма Вейча перемикальної функції від  $n$  змінних є прямокутною (якщо  $n$  – непарне) або квадратною (якщо  $n$  – парне) таблицею, що складається з  $2^n$  клітинок

Кожній клітинці на діаграмі Вейча відповідає двійковий набір, для якого може бути записана конституента одиниці або конституента нуля. Якщо на деякому наборі аргументів функція дорівнює 1, то у клітинку, що відповідає даному набору, записують 1. Клітинки, що відповідають наборам, на яких функція дорівнює 0, або заповнюють нулями, або залишають незаповненими.

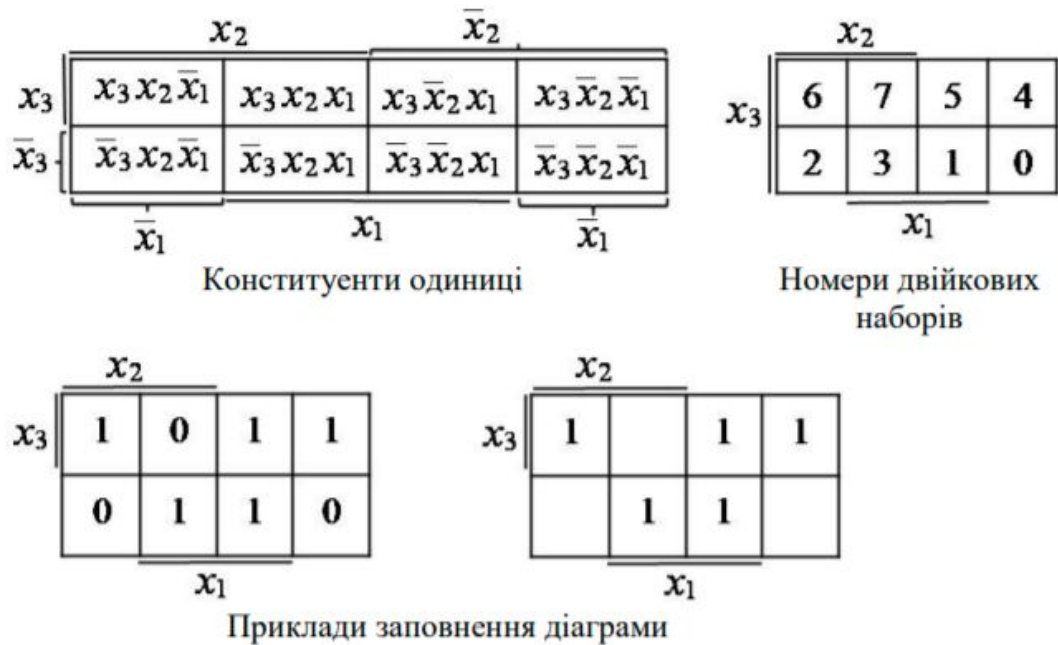


Діаграма Вейча перемикальної функції від двох змінних;

а) - конституенти одиниці; б) - номери двійкових наборів;

в) - приклад повного заповнення діаграми;

г) - приклад скороченого заповнення діаграми

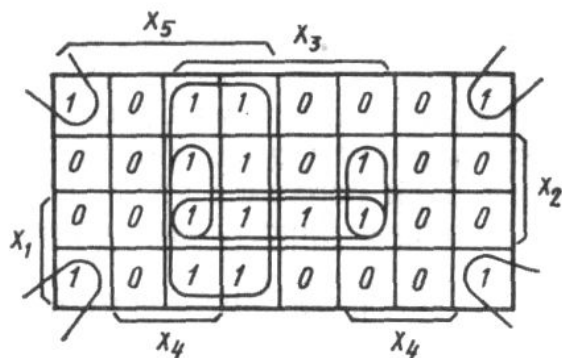


Діаграма Вейча перемикальної функції від трьох змінних

Процес мінімізації перемикальної функції полягає в наступному.

1. Заповнити діаграму Вейча (на підставі ДДНФ, ДКНФ або таблиці істинності функції).

2. Об'єднати одиниці в прямокутники (групи) максимально можливого розміру (... , 16, 8, 4, 2 одиниці) так, щоб покрити всі одиниці на діаграмі, прагнучи при цьому, щоб кількість утворених прямокутників (груп) була якнайменшою.



3. Записати МДНФ – для кожного визначення в п.2 прямокутника сформулювати відповідний кон'юнктивний терм (просту імпліканту) та об'єднати їх знаком диз'юнкції.

Задача 3.7. Мінімізувати повністю визначену функцію  $z$  від трьох змінних, задану в ДКНФ:

$$z_{\text{ДКНФ}} = (x_3 \vee \overline{x_2} \vee x_1)(x_3 \vee \overline{x_2} \vee \overline{x_1})(\overline{x_3} \vee x_2 \vee x_1)(\overline{x_3} \vee \overline{x_2} \vee \overline{x_1})$$

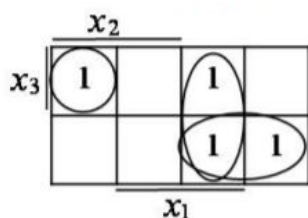
Розв'язування. Задану функцію запишемо в числовому вигляді:

$$z_{\text{ДКНФ}} = \overline{2} \cdot \overline{3} \cdot \overline{4} \cdot \overline{7}$$

Оскільки функція  $z$  - повністю визначена, то її ДДНФ має вигляд:

$$z_{\text{ДКНФ}} = 0 \vee 1 \vee 5 \vee 6$$

Тепер заповнимо діаграму Вейча та мінімізуємо функцію:



$$z_{\text{МДНФ}} = \overline{x_3} \overline{x_2} \vee \overline{x_2} x_1 \vee x_3 x_2 \overline{x_1}$$

Як і діаграма Вейча, карта Карно є розгорткою  $n$ -вимірного куба на площину. Карти Карно, як і діаграми Вейча, дозволяють отримувати як МДНФ, так і МКНФ перемикальних функцій. Рядки та стовпці карт Карно нумерують кодом Грея. Код Грея – це такий спосіб двійкової нумерації, коли наступне двійкове число відрізняється від попереднього лише в одному розряді.

$x_2 \backslash x_1$	0	1
0	0	1
1	2	3

*a*

$x_3 \backslash x_2 x_1$	00	01	11	10
0	0	1	3	2
1	4	5	7	6

*б*

$x_4 x_3 \backslash x_2 x_1$	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

*в*

$x_5 x_4 \backslash x_3 x_2 x_1$	000	001	011	010	110	111	101	100
00	0	1	3	2	6	7	5	4
01	8	9	11	10	14	15	13	12
11	24	25	27	26	30	31	29	28
10	16	17	19	18	22	23	21	20

*г*

Карти Карно для перемикальних функцій від:

а) - двох; б) - трьох;

в) - чотирьох; г) - п'яти змінних

Карта Карно відрізняється від діаграми Вейча лише порядком нумерації клітинок.

Діаграми Вейча

	$x_1$				
$x_2$	<table> <tr><td>3</td><td>2</td></tr> <tr><td>1</td><td>0</td></tr> </table>	3	2	1	0
3	2				
1	0				

	$x_2$								
$x_3$	<table> <tr><td>6</td><td>7</td><td>5</td><td>4</td></tr> <tr><td>2</td><td>3</td><td>1</td><td>0</td></tr> </table>	6	7	5	4	2	3	1	0
6	7	5	4						
2	3	1	0						
	$x_1$								

	$x_3$																
$x_4$	<table> <tr><td>12</td><td>13</td><td>9</td><td>8</td></tr> <tr><td>14</td><td>15</td><td>11</td><td>10</td></tr> <tr><td>6</td><td>7</td><td>3</td><td>2</td></tr> <tr><td>4</td><td>5</td><td>1</td><td>0</td></tr> </table>	12	13	9	8	14	15	11	10	6	7	3	2	4	5	1	0
12	13	9	8														
14	15	11	10														
6	7	3	2														
4	5	1	0														
	$x_1$																
	$x_2$																

Карти Карно

$x_1$	0	1
$x_2$	0	1
1	2	3

	$x_1$				
$x_2$	<table> <tr><td></td><td></td></tr> <tr><td></td><td></td></tr> </table>				

	$x_2 x_1$
$x_3$	00 01 11 10
0	0 1 3 2
1	4 5 7 6

	$x_2$								
$x_3$	<table> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> </table>								
	$x_1$								

	$x_2 x_1$
$x_4 x_3$	00 01 11 10
00	0 1 3 2
01	4 5 7 6
11	12 13 15 14
10	8 9 11 10

	$x_2$																
$x_4$	<table> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> </table>																
	$x_1$																
	$x_3$																

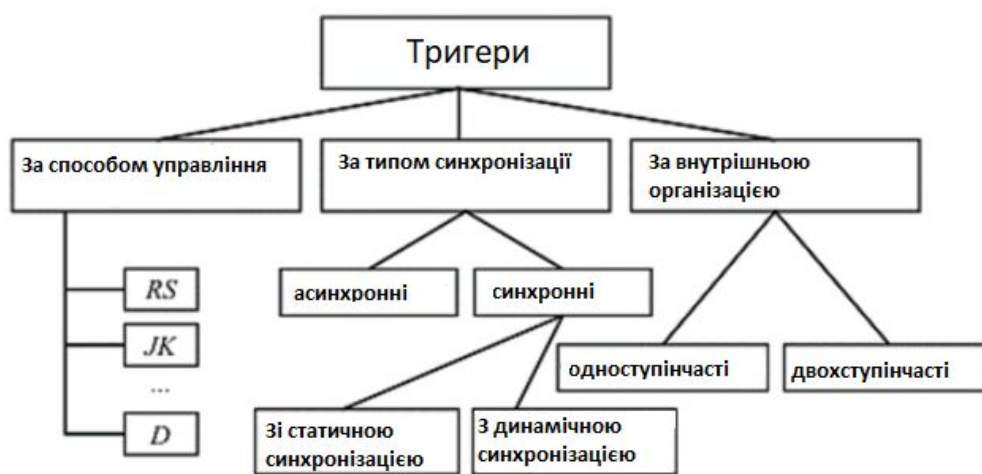
Порівняння діаграм Вейча та карт Карно

## Розділ 2. Типові вузли цифрових ЕОМ

### 2.1 Тригери. Регістри. Лічильники

**Три́гер** (англ. trigger, flip-flop) — електронна логічна схема, яка має два стійкі стани, в яких може перебувати, доки не зміняться відповідним чином сигнали керування. Напруги і струми на виході тригера можуть змінюватися стрибкоподібно. Їх можна використовувати для збереження інформації, розміром 1 біт. Тригери – це основний елемент для побудови різноманітних запам'ятовуючих пристроїв.

В арифметичних і логічних пристроях для збереження інформації найчастіше використовують тригери – пристрої з двома стійкими станами по виходу, які містять елементарну запам'ятовувальну комірку (бістабільна схема БС) і схему керування (СК). Схема керування перетворює інформацію, яка надходить, на комбінацію сигналів, що діють, безпосередньо на входи елементарної запам'ятовувальної комірки. Для забезпечення надійного перемикавання в точках А для деяких тригерів потрібні кола затримки. Схему такого тригера називають схемою типу М-S (master-slave), оскільки стан однієї БС, яку називають веденою, повторює стан додаткової БС, яку називають ведучою.



Класифікація тригерів

За способом організації логічних зв'язків розрізняють тригери з запуском (RS-тригери); з лічильним входом (Т-тригери); тригери затримки (D-тригери); універсальні (JK-тригери); комбіновані (наприклад, RST-, JKRS-, DRS-тригери).

### Асинхронний rs-тригер

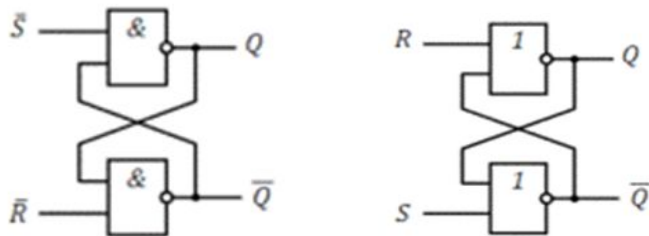
RS-тригер, або SR-тригер— тригер, який зберігає свій попередній стан при нульових входах та змінює свій вихідний стан при подачі на один з його входів одиниці.

При подачі одиниці (високої напруги) на вхід S (Set — встановити) вихідний стан стає рівним логічній одиниці – встановлення одиниці.

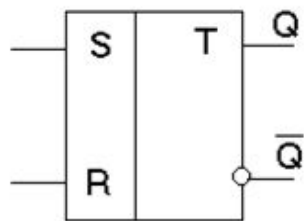
При подачі одиниці (високої напруги) на вхід R (Reset — скинути) вихідний стан стає рівним логічному нулю (низька напруга) – встановлення нуля.

Стан, при якому на обидва входи R і S одночасно подані логічні одиниці, є забороненим.

R	S	Q(t)	Q(t+1)	Режим
0	0	0	0	Збереження інформації
0	0	1	1	
0	1	0	1	Установка 1
0	1	1	1	
1	0	0	0	Установка 0
1	0	1	0	
1	1	0	1	Заборонено
1	1	1	0	



Реалізація асинхронного RS-тригера на логічних елементах І-НЕ та АБО-НЕ



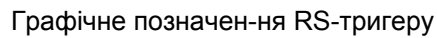
Графічне позначення RS-тригера

### Синхронний rs-тригер

Синхронний RS-тригер реагує на значення, подані на входи S та R тільки при наявності одиниці на синхронізуючому вході C. Він дозволяє уникнути перехідних станів, коли один сигнал на вхід може поступити раніше, ніж другий сигнал, і схема буде працювати неправильно. Для цього передбачений синхронний сигнал, який дозволяє вмикати тригер в потрібний момент часу.

Принцип роботи синхронного RS-тригера: схема тригера на елементах І-НІ керується за допомогою нуля, а на виході елементів І-НІ буде нуль тільки тоді, коли на входах будуть одиниці. При наявності одиниці на вході синхронізації, робота синхронного тригера від асинхронного нічим не відрізняється.

C	R	S	Q(t)	Q(t+1)	Режим
0	x	x	0	0	Збереження інформації
0	x	x	1	1	
1	0	0	0	0	Збереження інформації
1	0	0	1	1	
1	0	1	0	1	Запис 1
1	0	1	1	1	
1	1	0	0	0	Запис 0
1	1	0	1	0	
1	1	1	0	x	Заборонено
1	1	1	1	x	



D-тригер також називають тригером запам'ятовування інформації, або тригером-засувка (на рос. - защёлка). Він завжди синхронний. В раніше розглянутому RS-тригері було два вхідних сигнали, але для передачі двійкового коду достатньо одного входу з різними рівнями напруги (високий – 1, низький - 0). На два входи не можна було подати одиницю одночасно, тому в D-тригері два входи об'єднані за допомогою інверсії, що виключає можливість виникнення забороненого стану. Він має два входи – інформаційний вхід D і сигнал синхронізації C.

D	C	Q(t)	Q(t+1)	Режим
0	~	Q(t)	Q(t+1)	Збереження
1	1	~	1	Запис 1
1	0	~	0	Запис 0



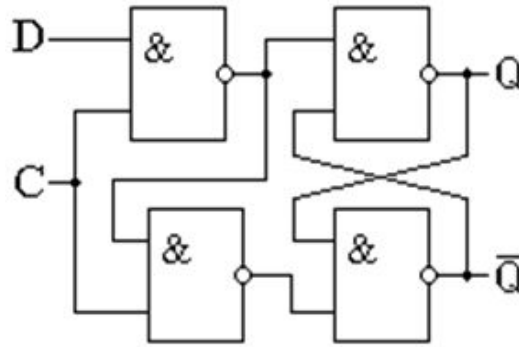
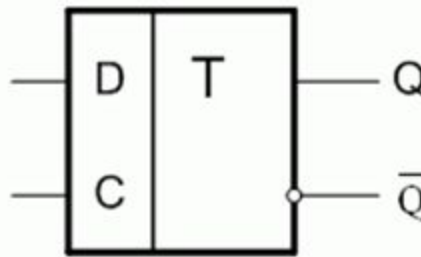


Схема синхронного D-тригера на елементах І-НЕ



Графічне позначення D-тригера

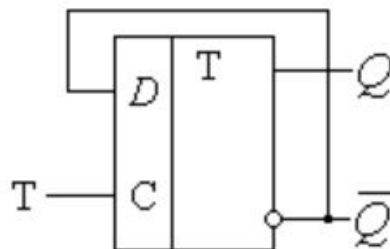
### Т-тригер

Т-тригер – рахунковий тригер. Він має тільки один вхід. Його можна побудувати на основі D-тригера чи RS-тригера. На основі RS-тригера, Т-тригер будується шляхом з'єднання інверсного виходу з входом Set, а прямого виходу з входом Reset. На основі D-тригера, Т-тригер будується шляхом об'єднання інверсного виходу з входом D.

Принцип роботи Т-тригера: після надходження імпульсу на вхід Т стан тригера змінюється на прямо протилежний. Рахунковим він називається тому, що Т-тригер підраховує, скільки імпульсів поступило на його вхід.

Якщо на вхід Т не поступив імпульс, то вихід Q буде таким самим, як і в попередньому стані. Як тільки на вхід Т поступить імпульс, вихід Q прийме протилежне значення. Режим зміни стану ще називається режимом перекидання (на рос. - переброс).

T	Q(t)	Q(t+1)	Режим
0	1	1	Збереження інформації
0	0	0	
1	1	0	Зміна стан
1	0	1	



Графічне позначення Т-тригера

### JK-тригер

JK-тригер називається універсальним. Він є модифікацією RS-тригера для вирішення проблеми забороненого режиму. Для того, щоб виключити заборонений стан, його схема змінена наступним чином:

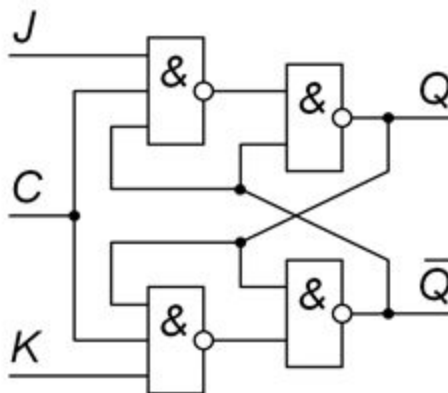
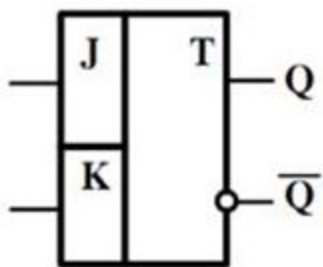


Схема JK-тригера на елементах I-HI

На основі JK-тригера можна побудувати D-тригер і T-тригер (він переходить в інверсний стан кожний раз, коли одночасно поступають одиниці на входи J і K). Ця властивість дозволяє створити на основі JK-тригера T-тригер, об'єднуючи входи J і K.

C	J	K	Q(t)	Q(t+1)	Режим
0	~	~	0	0	Збереження інформації
0	~	~	1	1	
1	0	0	0	0	
1	0	0	1	1	
1	0	1	~	1	Запис 1
1	1	0	~	0	Запис 0
1	1	1	0	1	Рахунковий режим
1	1	1	1	0	



Графічне позначення JK-тригера

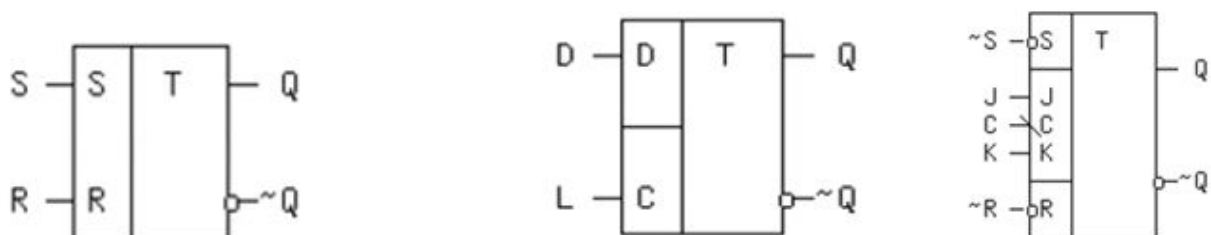
**Регістр** (англ. Register) являє собою пристрій, головним призначенням якого є зберігання інформації у вигляді багаторозрядних двійкових чисел (двійкового коду).

Регістром називають операційний вузол, призначений для запису, зберігання та видачі слів, а також для виконання мікрооперацій над ними.

В регістрах інформація запам'ятовується короткочасно, тобто на період одного або кількох циклів роботи всієї системи. Регістри, перш за все, призначені для запису, зберігання і читання одного двійкового числа. Крім цих основних операцій регістри можуть виконувати додаткові операції: інвертування числа, скидання в нульовий стан, перетворення послідовного коду в паралельний і навпаки, зсуву числа вліво або вправо тощо.

Регістр складається із запам'ятовувальних елементів (тригерів). Кількість розрядів (тригерів) регістра називають його довжиною.

Запам'ятовувальні елементи регістру виготовляють на основі RS-, D- або JK-тригерів у кількості, що відповідає кількості розрядів двійкового числа. Для допоміжних операцій (введення до регістру або виведення з нього числа, яке зберігається, перетворення коду двійкового числа, зсуву числа на певне число розрядів вліво або вправо) застосовують комбінаційні схеми на основі логічних елементів.

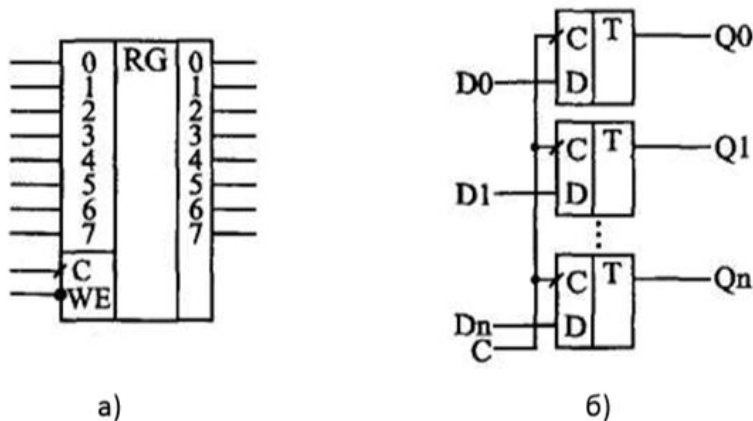


Всі регістри діляться на дві великі групи за способом запису і зчитування двійкової інформації:

- паралельні регістри;
- регістри зсуву (або зсувні регістри).

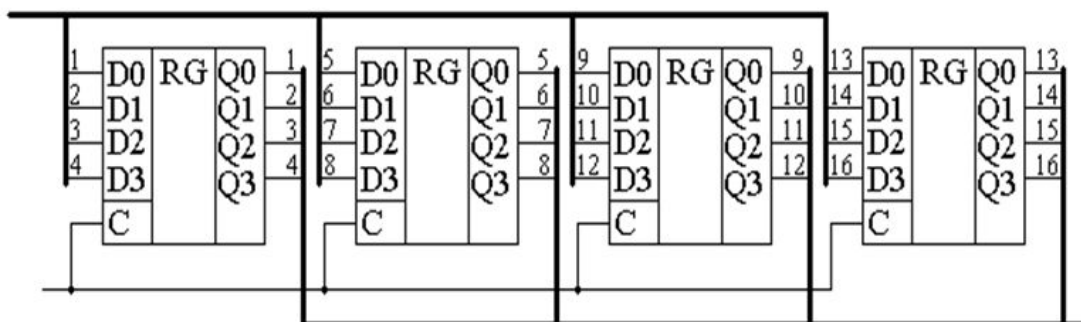
### **Паралельні регістри**

В паралельних регістрах інформація (двійкове число) записується одночасно до всіх розрядів и одночасно виводиться назовні (паралельний код)



Схематичне позначення (а) і принцип побудови (б) паралельного регістру

При вирішенні практичних завдань часто потрібно розрядність паралельних регістрів більше восьми. В такому випадку можна збільшувати їх розрядність паралельним з'єднанням готових мікросхем. Принципова схема паралельного з'єднання чотирьох регістрів наведена на малюнку:

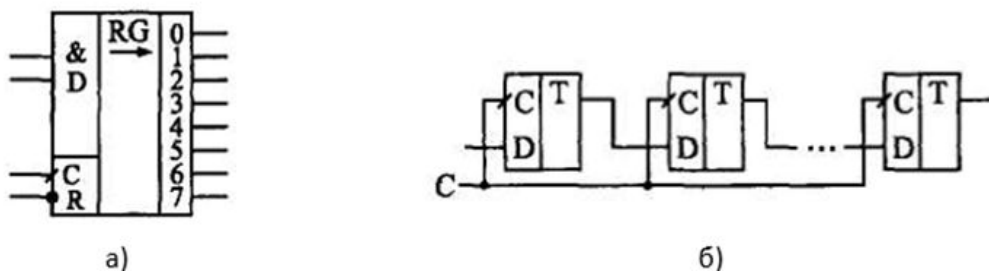


Збільшення розрядності паралельного регістра

В даний час паралельні регістри зазвичай є частиною більш складних цифрових пристроїв, таких як цифрові фільтри, ОЗУ, синтезатори частот або схеми прямого цифрового синтезу DDS. Подібні схеми не реалізуються на мікросхемах середньої інтеграції, а є частиною великих інтегральних мікросхем (BIC), таких як мікропроцесори, ASIC або FPGA.

### Зсувні регістри

В зсувних регістрах використовуються D-тригери, сполучені в послідовний ланцюжок (вихід кожного попереднього тригера сполучений з входом D наступного тригера).



Схематичне позначення (а) і принцип побудови (б) зсувного регістру

Основний режим їх роботи - це зсув розрядів коду, записаного в ці тригери, Тобто по тактовому сигналу вміст кожного попереднього тригера переписується в наступний по порядку в ланцюжку тригер. Зсув буває двох видів: вправо - це основний режим, який є у всіх зсувних регістрів, і вліво - цей режим є тільки у деяких реверсивних зсувних регістрів. Зсув інформації регістром вправо є зсув в бік розрядів, що мають великі номери, а зсув інформації регістром вліво - це зсув в бік розрядів, що мають менші номери. Відповідно зсув двійкового числа вправо буде зсувом в бік молодших розрядів, а зсув вліво - зсувом в бік старших розрядів. Більшість регістрів зсуву має вісім розрядів. Зсувні регістри допускають каскадування - спільне включення для збільшення розрядності.

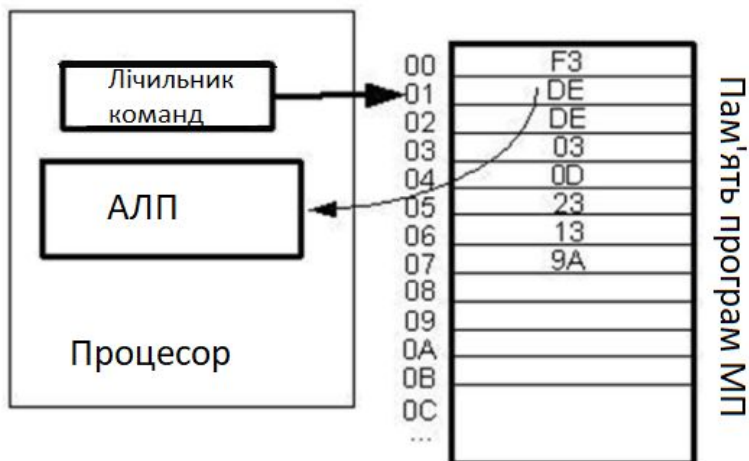
**Лічильник** - послідовнісна схема, яка перетворює поступаючі на вхід імпульси в паралельний двійковий код, відповідний їх кількості.



Команди програми розташовані в пам'яті одна за одною, і тим організовується вибірка ланцюжка команд з послідовно розташованих елементів пам'яті.

Якщо ж після виконання команди потрібно перейти не до наступної, а до якоїсь іншої, використовуються команди умовного або безумовного переходів, які заносять в лічильник команд номер елементу пам'яті, що містить наступну команду.

Вибірка команд з пам'яті припиняється після досягнення і виконання команди «стоп».



## Команди умовного переходу

В командах умовного переходу рішення про примусовий перехід до іншої адреси (шляхом перезавантаження лічильника команд) або продовження звичайного режиму виконання програми приймається в залежності від виконання або невиконання певної умови.

Такою умовою може бути, наприклад, рівність певних значень, або їхня нерівність. Для обчислення цього факту часто використовують арифметичні команди.



Формат RISC-команди умовного переходу за рівністю значень двох регістрів  
(PC:=адреса переходу ЯКЩО R1=R2)

## Команди безумовного переходу

Безумовний перехід примусово перезавантажує лічильник команд адресою (повністю або лише молодшу частину адреси), яка міститься у відповідному полі команди. Діапазон можливих адрес для переходу визначається розрядністю поля команди.



Команди виклику підпрограм і повернення з підпрограм — передають керування підпрограмі, зберігаючи адресу повернення й, можливо, контекст процесора (інформації про стан системи), а також організують коректне відновлення після виходу з підпрограми.

Команди виклику оброблювачів переривань — передають керування оброблювачу переривань. Іноді ці команди розглядаються як особливий випадок команд виклику підпрограм.





Програма може починатися та закінчуватися в будь-якому місці діапазону від 0 до 65535. Щоб звернутися до будь-якої із цих адрес лічильник повинен володіти 16 двійковими розрядами.

Всі команди слідують одна за одною в строго визначеному порядку, тому коли програма починає виконуватися, першим вмістом лічильника є початкова адреса програми (адреса першої команди). Потім в лічильник команд додається по 1 (2 або 3), в залежності від довжини команди в байтах.

За способом перемикавання тригерів лічильники поділяються на асинхронні і синхронні. У асинхронних лічильниках тригери перемикаються послідовно (асинхронно) від розряду до розряду, а в синхронних одночасно.

## **2.2 Шифратори. Дешифратори**

**Дешифратор** (декодер) в цифровій електроніці - комбінаційна схема, яка перетворює  $n$ -розрядний двійковий, трійчастий або  $k$ -ічний код в  $k^n$ -ічний унітарний код, де  $k$  - основа системи числення.

Унітарний код - послідовність біт, що містить тільки один активний біт, а інші біти послідовності неактивні.

Активний біт - біт, що дорівнює або одиниці, або нулю (залежить від реалізації дешифратор).

Логічна функція дешифратора позначається буквами DC (decoder).

В комп'ютерах дешифратори використовують для виконання таких операцій:

- дешифрації коду операції, записаного в регістр команд процесора, що забезпечує вибір потрібної мікропрограми;
- перетворення коду адреси операнда в команді в керуючі сигнали вибору заданої комірки пам'яті в процесі записування або читання інформації;
- забезпечення візуалізації на зовнішніх пристроях;
- реалізації логічних операцій та побудови мультиплексорів і демультиплексорів.

### **Двійковий дешифратор**

Двійковий дешифратор працює наступним чином:

- на вхід дешифратора подається двійкове слово з  $n$  біт. Кількість допустимих вхідних комбінацій з  $n$  біт дорівнює  $2^n$ ;

- на виході у дешифратора формується бінарне слово з числа бітів, меншого або рівного  $2^n$ ;
- У вихідному слові завжди є один активний біт, що дорівнює 1 або 0, інші біти неактивні. Активність 0 або 1 залежить від конкретної реалізації дешифратора.

### Види дешифраторів

Дешифратор називається:

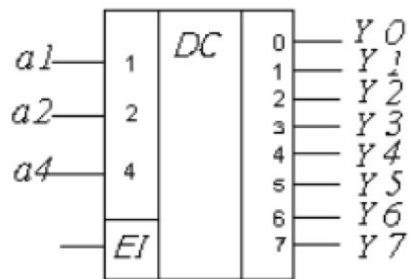
- повним, якщо число виходів одно максимально можливої розрядності вихідного слова  $2^n$ ;
- неповним, якщо частина вхідних розрядів не використовується (тобто число виходів менше  $2^n$ ).

У **лінійних дешифраторів**  $n$  змінних представляють сукупність не зв'язаних між собою  $2^n$  систем збігу на  $n$  входів, кожна з яких реалізує відповідну конститuentу одиниці.

Пірамідальні дешифратори будуються за принципом послідовних каскадів: на першому каскаді реалізуються конститuentи одиниці для 2 змінних, на  $n-1$  реалізуються конститuentи одиниці для  $n$  змінних, при цьому, на вході отримується вихід з попереднього каскаду.

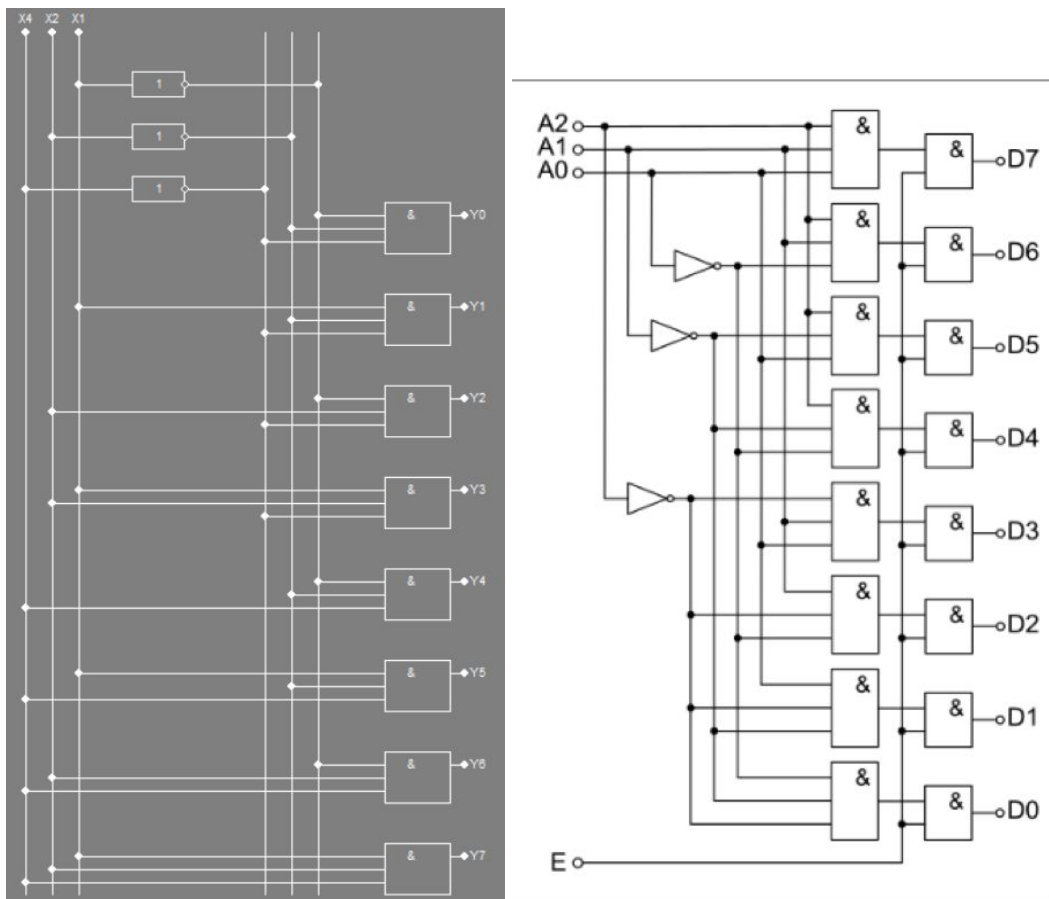
### Принцип роботи повного дешифратора 3 на 8

x4	x2	x1	y0	y1	y2	y3	y4	y5	y6	y7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

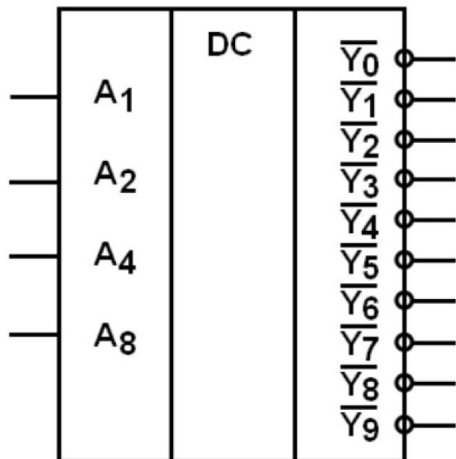


Представимо число 010 в десятковому вигляді  $010 = 0 \cdot 4 + 1 \cdot 2 + 0 \cdot 1 = 2$ .  
 Тепер подивимось на схему і бачимо, що вага розряду відповідає цифрі, на яку множиться 0 або 1 у формулі.

Для реалізації повного дешифратора на  $m$  входів потрібно  $n = 2^m$  елементів кон'юнкції (кількість входів кожного елемента "І" дорівнює  $m$ ) і  $m$  елементів заперечення.



Дешифратор К555ІД6 серії К555



### Семисегментний дешифратор

Індикатори призначені для відображення різних видів інформації для людини. Використання семисегментних індикаторів дозволяє сформувати всі десяткові цифри і частину букв.



**Шифратор** (кодер) – це електронний пристрій, в даному випадку мікросхема, яка перетворює код однієї системи числення в код іншої системи.

Число входів і виходів в повному  $k$ -ковому шифраторі задається співвідношенням  $n = 2^m$ , де

$n$  — кількість входів,

$m$  — кількість виходів.

Основне призначення шифратора - перетворення номера джерела сигналу в код. Наприклад, уявімо, що ми тримаємо в руках звичайний калькулятор. Оскільки всі дії в калькуляторі виконуються з двійковими числами (згадаємо основи цифрової електроніки), то після клавіатури стоїть шифратор, який перетворює введені числа в двійкову форму.

Всі кнопки калькулятора з'єднуються із загальним проводом і, натиснувши, наприклад, кнопку 5 на вході шифратора, ми відразу одержимо двійкову форму даного числа на його виході.

Шифратор виконує обернену функцію до функції дешифратора.

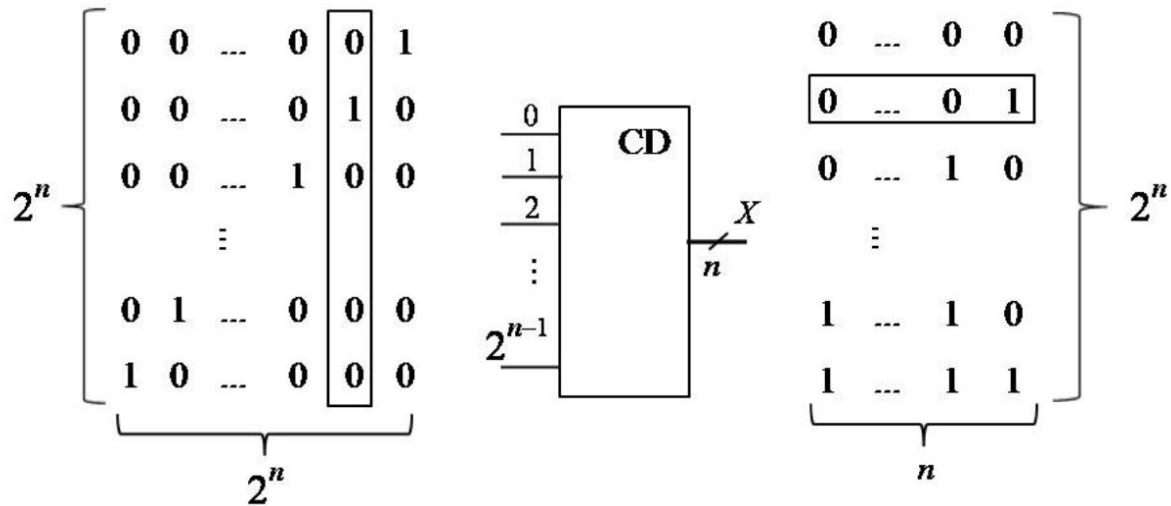
### **Види шифраторів**

1. Двійковий шифратор виконує логічну функцію перетворення унітарної  $n$ -ічного однозначного коду в двійковий. При подачі сигналу на один з  $n$  входів (обов'язково на один, не більше) на виході з'являється двійковий код номера активного входу.
2. Трійчастий шифратор виконує логічну функцію перетворення унарна  $n$ -ічного однозначного (одне одиничного або одне нульового) коду в трійчастий. При подачі сигналу («1» в одне одиничному коді або «0» в одне нульовому коді) на один з  $n$  входів на виході з'являється потрібний код номера активного входу ( $n = 3^m$ )
3. Повний  $k$ -тий шифратор ( $n = 2^m$ )  
Якщо кількість входів настільки велике, що в шифраторі використовуються всі можливі комбінації сигналів на виході, то такий шифратор називається повним, якщо не все, то неповним.
4. Пріоритетний шифратор відрізняється від шифратора наявністю додаткової логічної схеми виділення активного рівня старшого входу

для забезпечення умови працездатності шифратора (тільки один рівень на вході активний). Рівні сигналів на інших входах схемою ігноруються.

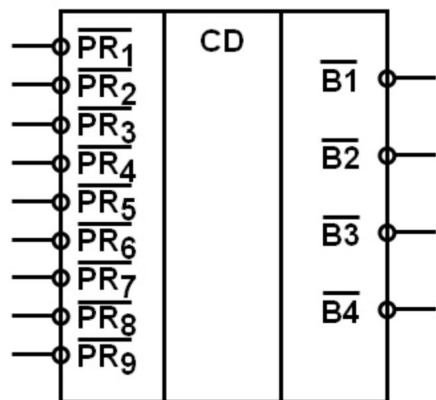
На входи шифратора в деякий момент часу надходить код, що містить одиницю лише в одному розряді, а в решті розрядах – нулі. Функція шифратора полягає в тому, щоб розпізнати, в якому саме розряді вхідного унарного коду міститься одиниця, і видати на вихід номер цього розряду (нумерація розрядів починається з нуля).

Рисунок. Перетворення  $2^n$ -розрядного унарного коду в  $n$ -розрядний позиційний код (число)  $X$



**Принцип роботи шифратора**

**Шифратор з пріоритетом (пріоритетний шифратор) K555IB3 серії мікросхем K555 (ТТЛШ)**



Шифратор має 9 інверсних входів, позначених через  $\overline{PR_1}$ , ...,  $\overline{PR_9}$ .  
 Аббревіатура PR позначає «пріоритет». Шифратор має чотири інверсних виходу  $\overline{B_1}$ , ...,  $\overline{B_8}$ . Аббревіатура B означає «шина» (від англ. Bus). Цифри визначають значення активного рівня (нуля) у відповідному розряді двійкового числа. Наприклад,  $\overline{B_8}$  позначає, що нуль на цьому виході відповідає числу 8. Очевидно, що це неповний шифратор.

Якщо на всіх входах - логічна одиниця, то на всіх виходах також логічна одиниця, що відповідає числу 0 в так званому інверсному коді (1111). Якщо хоча б на одному вході є логічний нуль, то стан вихідних сигналів визначається найбільшим номером входу, на якому є логічний нуль, і не залежить від сигналів на входах, що мають менший номер.

Наприклад, якщо на вході  $\overline{PR_1}$  - логічний нуль, а на всіх інших входах - логічна одиниця, то на виходах є наступні сигнали:  $\overline{B_1} = 0$ ,  $\overline{B_2} = 1$ ,  $\overline{B_4} = 1$ ,  $\overline{B_8} = 1$ , що відповідає числу 1 в інверсному коді (1110).

Якщо на вході  $\overline{PR_9}$  логічний нуль, то незалежно від інших вхідних сигналів на виходах є наступні сигнали:  $\overline{B_1} = 0$ ,  $\overline{B_2} = 1$ ,  $\overline{B_4} = 1$ ,  $\overline{B_8} = 0$ , що відповідає числу 9 в інверсному коді (0110).

### Шифратор 8 на 3

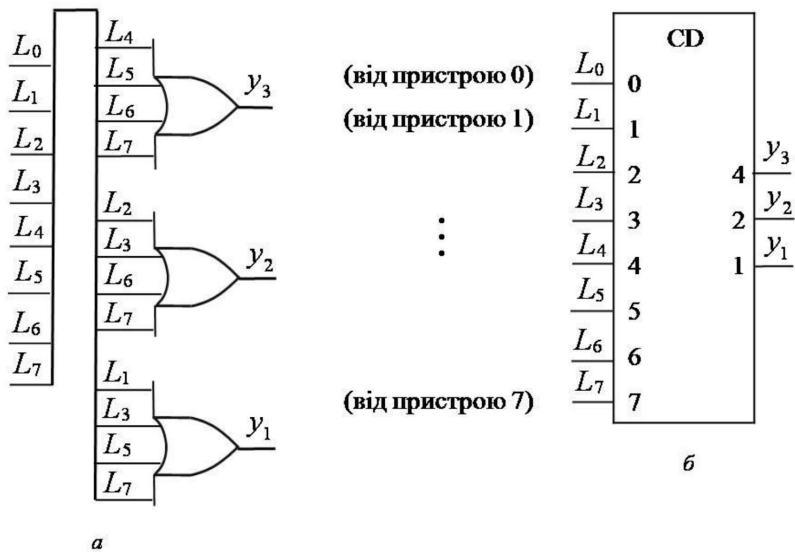
Входи								Виходи		
L7	L6	L5	L4	L3	L2	L1	L0	y3	y2	y1
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1



0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

З таблиці функціонування (істинності) визначаємо функції виходів шифратора:

$$\begin{cases} y_3 = L_4 \vee L_5 \vee L_6 \vee L_7, \\ y_2 = L_2 \vee L_3 \vee L_6 \vee L_7, \\ y_1 = L_1 \vee L_3 \vee L_5 \vee L_7. \end{cases}$$



Шифратор 8×3 з прямими входами:

*a* – функціональна схема; *б* – умовне графічне позначення

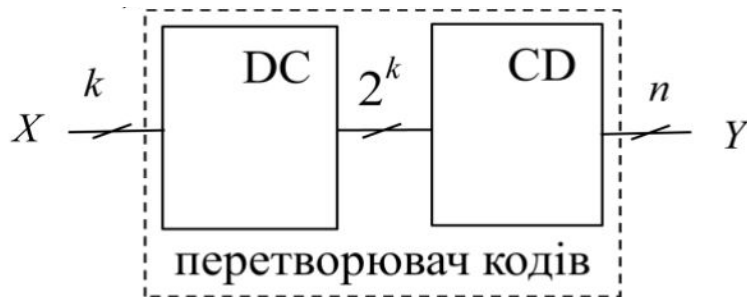
### 2.3 Перетворювачі кодів

Перетворювач коду (code converter) – це комбінаційна схема, що має  $k$  входів та  $n$  виходів, яка реалізує перетворення  $k$ -розрядного слова, що надходить на входи схеми, в  $n$ -розрядне слово на виходах схеми.

Прикладами перетворювачів є схеми для:

- перетворення чисел з однієї системи числення в іншу,
- керування символьними індикаторами,
- перетворення одного двійково-десятькового коду (ДДК) в інший ДДК,
- перетворення позиційного коду в унарний код (дешифратор),
- перетворення унарного коду в позиційний (шифратор) тощо.

Універсальним способом реалізації перетворювачів кодів є використання схеми побудови «дешифратор-шифратор» (схема побудови «DC-CD»).



Іншим способом реалізації перетворювачів є побудова (синтез) комбінаційної схеми на основі лише логічних елементів.

Застосуємо схему побудови «дешифратор-шифратор» для реалізації перетворювача двійково-десятькового коду «8-4-2-1» в код Грея (табл. 5.13), де 8, 4, 2, 1 – ваги двійкових розрядів.

Таблиця функціонування перетворювача ДДК «8-4-2-1» в код Грея

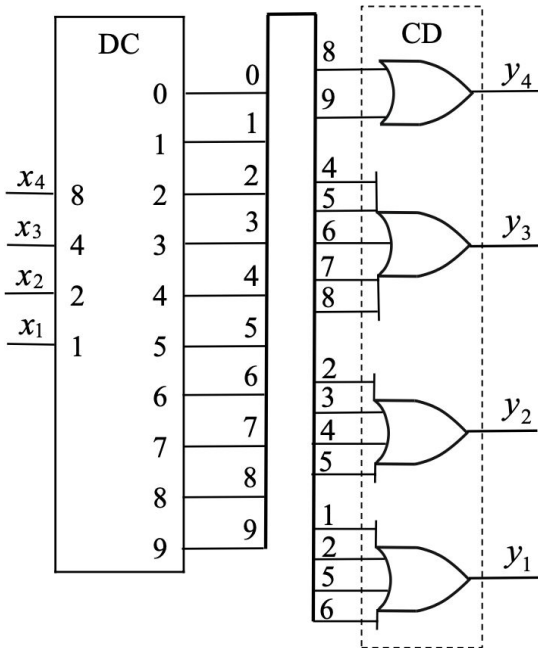
Десяткова цифра	Двійково-десятковий код «8-4-2-1»				Код Грея			
	$x_4$	$x_3$	$x_2$	$x_1$	$y_4$	$y_3$	$y_2$	$y_1$
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	0	0	0

**Код Грея**, який також називають рефлексним кодом, використовують у пристроях, що перетворюють лінійні або кутові механічні переміщення в цифрову форму. У коді Грея двом будь-яким сусіднім десятковим цифрам відповідають двійкові тетради, що відрізняються лише в одному розряді.

Для побудови перетворювача використаємо неповний дешифратор  $4 \times 10$  з прямими виходами. Тоді функції виходів  $y_4$ ,  $y_3$ ,  $y_2$ ,  $y_1$  перетворювача мають вигляд:

$$\begin{cases} y_4 = 8 \vee 9, \\ y_3 = 4 \vee 5 \vee 6 \vee 7 \vee 8, \\ y_2 = 2 \vee 3 \vee 4 \vee 5, \\ y_1 = 1 \vee 2 \vee 5 \vee 6, \end{cases}$$

, де цифрами 1-9 позначено відповідні набори змінних  $x_4 - x_1$ , тобто номери виходів дешифратора.



Функціональна схема перетворювача ДДК «8-4-2-1» в код Грея

Реалізуємо перетворювач за схемою побудови «DC-CD».

Оскільки таблиця функціонування перетворювача (стовпці  $y_7 - y_1$ ) містить мало нулів (21 з 70 цифр), то перетворювач доцільно побудувати на основі дешифратора з інверсними виходами. Це істотно знижує складність схеми.

Таблиця істинності перетворювача кодів для керування семисегментним цифровим індикатором

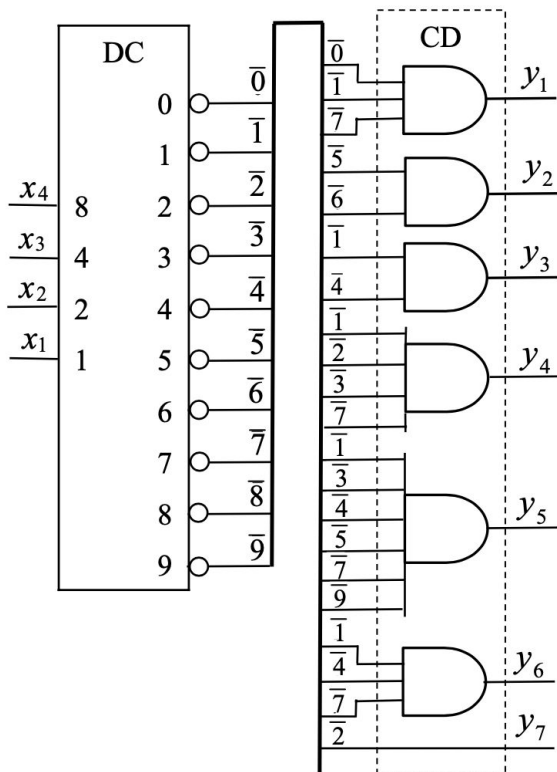
Входи				Виходи						
$x_4$	$x_3$	$x_2$	$x_1$	$y_7$	$y_6$	$y_5$	$y_4$	$y_3$	$y_2$	$y_1$
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	1	0	0	0	0	1	0
0	0	1	0	0	1	1	0	1	1	1
0	0	1	1	1	1	0	0	1	1	1
0	1	0	0	1	0	0	1	0	1	1
0	1	0	1	1	1	0	1	1	0	1
0	1	1	0	1	1	1	1	1	0	1
0	1	1	1	1	0	0	0	1	1	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1

Роботу перетворювача описують системою перемикальних функцій:

$$\begin{cases} y_7 = \bar{2}, \\ y_6 = \bar{1} \cdot \bar{4} \cdot \bar{7}, \\ y_5 = \bar{1} \cdot \bar{3} \cdot \bar{4} \cdot \bar{5} \cdot \bar{7} \cdot \bar{9}, \\ y_4 = \bar{1} \cdot \bar{2} \cdot \bar{3} \cdot \bar{7}, \\ y_3 = \bar{1} \cdot \bar{4}, \\ y_2 = \bar{5} \cdot \bar{6}, \\ y_1 = \bar{0} \cdot \bar{1} \cdot \bar{7}, \end{cases}$$

де  $\bar{1} - \bar{9}$  – номери інверсних виходів дешифратора.

Застосування схеми побудови «дешифратор-шифратор» є універсальним способом реалізації перетворювачів кодів, але часто отримувані за цією схемою структури перетворювачів не є найбільш економічними та швидкодіючими. Тому для побудови конкретно заданого перетворювача застосовують окремі методики та прийоми.

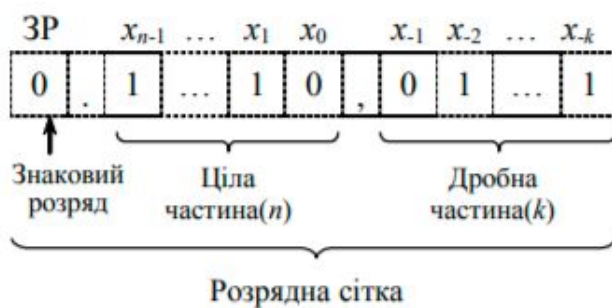


## Розділ 3. Комп'ютерна арифметика

### 3.1 Кодування від'ємних чисел в ЕОМ

Для спрощення арифметичних операцій, числа в цифрових пристроях ЕОМ подаються спеціальними кодами - прямим, оберненим і доповненим.

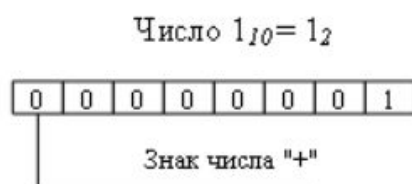
Розрядна сітка містить число, яке має  $n$ -розрядну цілу частину,  $k$ -розрядну дробову частину і знаковий розряд зліва від основних розрядів. Всі три способи використовують самий лівий (старший) розряд для кодування знака числа: знак "плюс" кодується нулем, а "мінус" - одиницею.



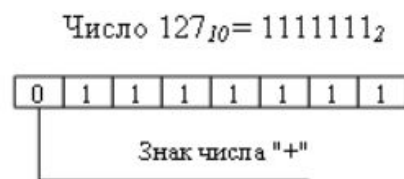
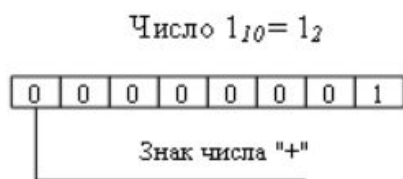
Від'ємні числа в прямому, зворотному і доповненому кодах мають різне представлення.

### 3.2 Форми представлення чисел у ЕОМ

Додатні числа у прямому, оберненому та додатковому кодах записуються однаково - двійковим кодом числа з цифрою 0 у знаковому розряді. Наприклад, при розмірі розрядної сітки  $n=8$ :



**Прямий код від'ємного числа** відрізняється від прямого коду додатного числа тим, що значення його знакового розряду дорівнює не 0, а 1. Наприклад, прямий код чисел -1 і -127 у 8-розрядній сітці:

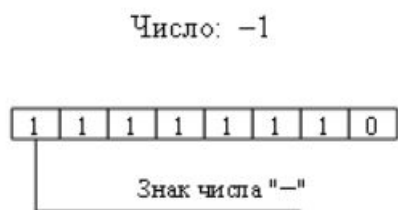


$$[X]_{\text{ПК}} = \begin{cases} X, & \text{якщо } X \geq 0; \\ 2^n + |X|, & \text{якщо } X \leq 0. \end{cases}$$

Для чисел  $A=10,101011$ ;  $B=-10,0111010$ :

$[A]_{\text{ПК}} = 0,10,101011$ ;  $[B]_{\text{ПК}} = 1,10,0111010$

**Обернений код від'ємного числа** отримується із прямого коду шляхом заміни його цифр на їх доповнення до 1. Знаковий розряд зберігається без змін. Заміна цифр їх доповненням для двійкової системи співпадає з операцією інверсії, тобто у всіх розрядах нулі потрібно замінити на 1, а одиниці - на 0. Наприклад, обернений код чисел -1 і -127 у 8-розрядній сітці:



$$[A]_{\text{ОК}} = \begin{cases} A, & \text{якщо } A \geq 0; \\ 2^{n+1} - 2^{-k} - |A| = 2^{n+1} - 2^k + A, & \text{якщо } A \leq 0. \end{cases}$$

Для чисел  $A=10,1011$ ;  $B=-01,11010$ :

$[A]_{\text{ОК}} = 0,10,1011$ ;  $[B]_{\text{ОК}} = 1,10,00101$

**Доповнений код від'ємного числа** отримується із оберненого коду збільшенням на 1 його молодшого розряду. При цьому перенос із знакового розряду ігнорується. Наприклад, додатковий код чисел -1 і -127 у 8-розрядній сітці:



$$[A]_{\text{ДК}} = \begin{cases} A, & \text{якщо } A \geq 0; \\ 2^{n+1} - |A| = 2^{n+1} + A, & \text{якщо } A < 0. \end{cases}$$

Для чисел A=-011,100; B=010,010:

[A] <sub>ПК</sub> = 1.011,100	B = 0.010,010
[A] <sub>ОК</sub> = 1.100,011	B = 0.010,010
+            1	
[A] <sub>ДК</sub> = 1.100,100	B = 0.010,010





Подання від'ємного числа в оберненому і додатковому кодах прямому

### 3.3 Додавання та віднімання чисел

Операції алгебраїчного підсумовування і віднімання неможливо виконувати в прямому коді із використанням звичайного суматора, оскільки знакові розряди і основні розряди повинні оброблятися по-різному. З використанням ОК та ДК операції додавання і віднімання можна виконувати за допомогою звичайних багаторозрядних суматорів, на яких оброблюються як основні, так і знакові розряди. Операція віднімання замінюється операцією додавання з числом, що має протилежний знак. Наприклад, операція  $S = A - B$  виконується як  $S = A + (-B)$ .

Під час додавання чисел із однаковими знаками може виникнути переповнення розрядної сітки, що приводить до втрати знака числа.

Формули подання модифікованих кодів мають вигляд:

$$[A]_{OK} = \begin{cases} A, & \text{якщо } A \geq 0; \\ 2^{n+2} - 2^{-k} + A, & \text{якщо } A < 0. \end{cases}$$

$$[A]_{DK} = \begin{cases} A, & \text{якщо } A \geq 0; \\ 2^{n+2} + A, & \text{якщо } A < 0. \end{cases}$$

В модифікованих кодах знакові розряди і основні розряди обробляються як єдине ціле. Правильний знак суми утворюється автоматично в процесі підсумовування цифр знакових і основних розрядів з урахуванням переносів.

**Характерною рисою ОК** є циклічний перенос зі старшого знакового розряду в молодший розряд суми. Перенос виникає автоматично, коли корекція результату потрібна.

Приклад 1. Задано  $A = 0,10101$ ;  $B = -0,01001$ . Знайти суму  $C$  з використанням ОК.

Виконання завдання:

$$\begin{array}{r} [A]_{\text{ЗК}} = 00,10101 \\ + [B]_{\text{ЗК}} = 11,10110 \\ \hline 00,01011 \\ + \quad \quad 1 \text{ (перенос)} \\ \hline [C]_{\text{ЗК}} = 00,01100 \end{array}$$

Приклад 2. Задано  $A = 0,01001$ ;  $B = -0,10101$ . Знайти суму  $C$  з використанням ОК.

Виконання завдання:

$$\begin{array}{r} [A]_{\text{ЗК}} = 00,01001 \\ + [B]_{\text{ЗК}} = 11,01010 \\ \hline [C]_{\text{ЗК}} = 11,10011 \\ \text{(Перенос відсутній)} \end{array}$$

Приклад 3. Задано обернені коди чисел  $A = -0,10101$  і  $B = -0,01001$ . Знайти обернений код суми  $C$ .

Виконання завдання:

$$\begin{array}{r} [A]_{\text{ЗК}} = 11,01010 \\ + [B]_{\text{ЗК}} = 11,10110 \\ \hline 11,00000 \\ + \quad \quad 1 \text{ (перенос)} \\ \hline [C]_{\text{ЗК}} = 11,00001 \end{array}$$

Приклад 4. Задано  $A = -0,10101$ ;  $B = -0,01110$ . Знайти суму  $C$ .

Виконання завдання:

$$\begin{array}{r} [A]_{\text{ЗК}} = 11,01010 \\ + [B]_{\text{ЗК}} = 11,10001 \\ \hline 10,11011 \\ + \quad \quad 1 \text{ (перенос)} \\ \hline [C] = 10,01100 - \text{від'ємне переповнення} \end{array}$$

**Підсумовування операндів у ДК** автоматично формує суму в ДК з урахуванням знаку. За застосування модифікованого коду корекції роботи не треба при будь-якому сполученні знаків доданків. Факт переповнення розрядної сітки можна визначити за розбіжністю значень знакових розрядів. Старший знаковий розряд завжди зберігає знак результату.

Приклад 5. Задано  $A = 0,10101$ ;  $B = 0,01001$ . Знайти суму  $C$  в ДК.

Виконання завдання:

$$\begin{array}{r} [A]_{\text{ДК}} = 00,10101 \\ + [B]_{\text{ДК}} = 00,01001 \\ \hline [C]_{\text{ДК}} = 00,11110 \end{array}$$

Приклад 6. Задано  $A = 0,10101$ ;  $B = -0,01001$ . Знайти суму  $C$  в ДК.

Виконання завдання:

$$\begin{array}{r} [A]_{\text{ДК}} = 00,10101 \\ + [B]_{\text{ДК}} = 11,10111 \\ \hline [C]_{\text{ДК}} = 00,01100 \end{array}$$

Приклад 7. Задано  $A = -0,10101$ ;  $B = -0,01001$ . Знайти  $C$  в ДК.

Виконання завдання:

$$\begin{array}{r} [A]_{\text{ДК}} = 11,01011 \\ + [B]_{\text{ДК}} = 11,10111 \\ \hline [C]_{\text{ДК}} = 11,00010 \end{array}$$

**Правила додавання двійкових значень розрядів** чисел  $X$  і  $Y$  наведені нижче. Тут показані правила додавання двійкових значень однойменних розрядів  $x_i$  та  $y_i$  чисел  $X$  і  $Y$  з урахуванням можливого переносу  $P_{i-1}$  із попереднього молодшого розряду і можливого переносу  $P_i$  в наступний старший розряд.

Значення $i$ -х двійкових розрядів чисел $X$ , $Y$ та $P_{i-1}$			Значення розряду суми $S_i$	Значення переносу в наступний розряд $P_i$
$x_i$	$y_i$	$P_{i-1}$		
0	0	0	0	0

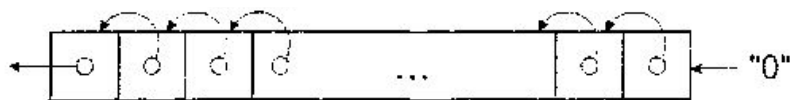
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

### 3.4 Зсуви в машинних кодах

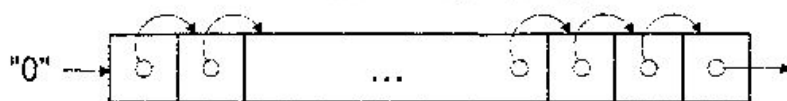
Існують два різновиди машинних зсувів:

- Логічний зсув;
- Арифметичний зсув;
- Циклічний зсув.

**Логічний зсув** це зміщення розрядів машинного слова у просторі із втратою розрядів, що виходять за межі розрядної сітки. Розряди, що звільняються заповнюються нулями.



Логічний зсув ліворуч



Логічний зсув праворуч

При виконанні логічного зсуву праворуч або ліворуч всі розряди слова зсуваються на один розряд у відповідну сторону, в перший розряд записується нуль, а останній розряд випадає. Для формату представлення даних без знаків

зсув на один розряд ліворуч еквівалентний множенню на два, а на один розряд праворуч - відповідно цілочисельному діленню на два.

*Приклад 1. Виконати логічний зсув двійкового числа вліво і вправо на один розряд.*

$$A_{(2)} = 0.0101,1101.$$

*Виконання завдання:*

$$\begin{array}{l} A \\ A \rightarrow \end{array} \left| \begin{array}{l} 1.0101,1101 \\ 0.1010,1110 \end{array} \right| *$$

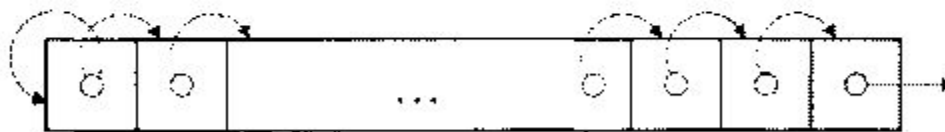
$$\begin{array}{l} A \\ \leftarrow A \end{array} \left| \begin{array}{l} 1.0101,1101 \\ *0.1011\ 1010 \end{array} \right|$$

**Арифметичний зсув** виконується з урахуванням знакового розряду.

Правила зсуву чисел поданих у ПК, ЗК та ДК відрізняються. Арифметичний зсув аналогічний логічному, але значення слова вважається знаковим числом, представленим в додатковому коді. Так, при правому зсуві старший біт зберігає своє значення. Лівий арифметичний зсув ідентичний логічному.



Арифметичний зсув ліворуч



Арифметичний зсув праворуч

### **Арифметичний зсув чисел поданих у ПК**

При цьому типі зсуву знаковий розряд не зсувається. Основні розряди зсуваються ліворуч або праворуч із заповненням нулями розрядів, що звільнилися при зсуві. Розряди, що вийшли за межі розрядної сітки втрачаються. За

арифметичного зсуву вліво можлива втрата значимості числа. За зсуву праворуч виникає похибка.

*Приклад 2. Виконати арифметичний зсув вліво і вправо на один розряд двійкових чисел, поданих у ПК.*

$$A_{(2)} = 1.10101; B_{(2)} = 0.11011$$

Виконання завдання:

$A$	1.	1	0	101	
$A \rightarrow$	1.	0	1	010	* Похибка
$\leftarrow A$	1.	*0	1	010	Втрата значимості

$B$	0.	1	1	011	
$B \rightarrow$	0.	0	1	101	* Похибка
$\leftarrow B$	0.	*1	0	110	Втрата значимості

### Арифметичний зсув ліворуч чисел, поданих у ОК.

Для визначення переповнення розрядної сітки при арифметичному зсуві використовують модифіковані коди.

Правило. Якщо під час зсуву від'ємного числа ліворуч за розрядну сітку виходить одиниця із знакового розряду, то необхідно виконати корекцію  $K = +2^{-k}$ .

*Приклад 1. Виконати арифметичний зсув ліворуч на один розряд двійкових чисел, поданих у ОК.*

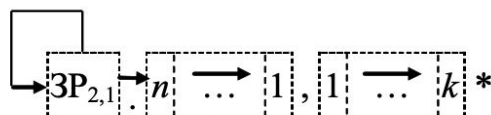
$$A_{(2)} = 11.1101,1010; B_{(2)} = 11.0011,0011.$$

Виконання завдання:

$A$	11.1101,0101	
$\leftarrow A$	11.1010,1011	←
$B$	11.0011,0011	
$\leftarrow B$	10.0110,0111	← Переповнення

### Арифметичний зсув праворуч чисел, поданих у ОК.

Правило. За зсуву праворуч від'ємного числа знаковий розряд переходить у поле основних розрядів і знов заповнюється тим самим значенням.



Операційна схема арифметичного зсуву праворуч від'ємних чисел у ОК

*Приклад 2. Виконати арифметичний зсув праворуч на один розряд двійкових чисел, поданих у ЗК.*

$$A_{(2)} = 00.1011,1001; B_{(2)} = 11.1010,0011.$$

*Виконання завдання:*

$$\begin{array}{l} A \quad 00.1011,1001 \\ \rightarrow A \quad 00.0101,1100^* \end{array}$$

$$\begin{array}{l} B \quad 11.1010,0011 \\ \rightarrow B \quad 11.1101,0001^* \end{array}$$

### Арифметичний зсув ліворуч чисел, поданих у ДК

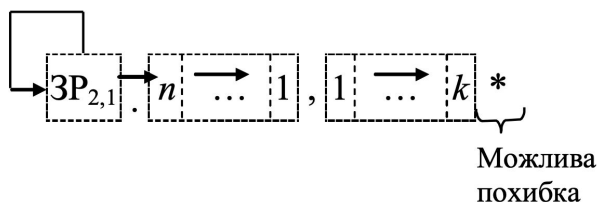
Правило. За зсуву ліворуч числа поданого у ДК розряди, що звільнились заповнюються нулями. Якщо знаковий розряд змінює значущість виникає втрата значимості числа.



Операційна схема арифметичного зсуву ліворуч від'ємних чисел у ДК

### Арифметичний зсув праворуч чисел, поданих у ДК

Правило. За зсуву праворуч числа поданого у ДК знаковий розряд розповсюджується у поле основних розрядів і знов заповнюється тим самим значенням. В результаті може виникнути похибка.



Операційна схема арифметичного зсуву праворуч від'ємних чисел у ДК

*Приклад 3. Виконати арифметичний зсув праворуч і ліворуч на один розряд двійкових чисел, поданих у ДК.*

$$A_{(2)} = 0.1011,0111; B_{(2)} = 1.1011,1001.$$

*Виконання завдання:*

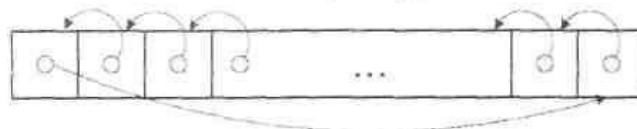
$$\begin{array}{l} A \\ \leftarrow A \\ A \rightarrow \end{array} \left[ \begin{array}{l} 00.1011,0111 \\ 01.0110,1110 \\ 00.0101,1011^* \end{array} \right]$$

$$\begin{array}{l} B \\ \leftarrow B \\ \rightarrow B \end{array} \left[ \begin{array}{l} 11.1011,1001 \\ 11.0111,0010 \\ 11.1101,1100^* \end{array} \right]$$

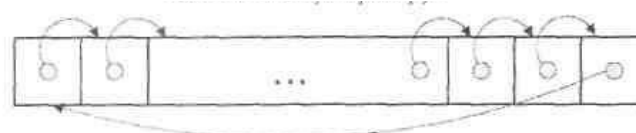
### Циклічний зсув

При циклічному зсуві, значення останнього біта за напрямом зсуву копіюється в перший біт (і копіюється в біт переносу).

Також розрізняють циклічний зсув через біт переносу — при ньому перший біт за напрямом зсуву отримує значення з біта переносу, а значення останнього біта зсувається в біт переносу.



Циклічний зсув ліворуч



Циклічний зсув праворуч

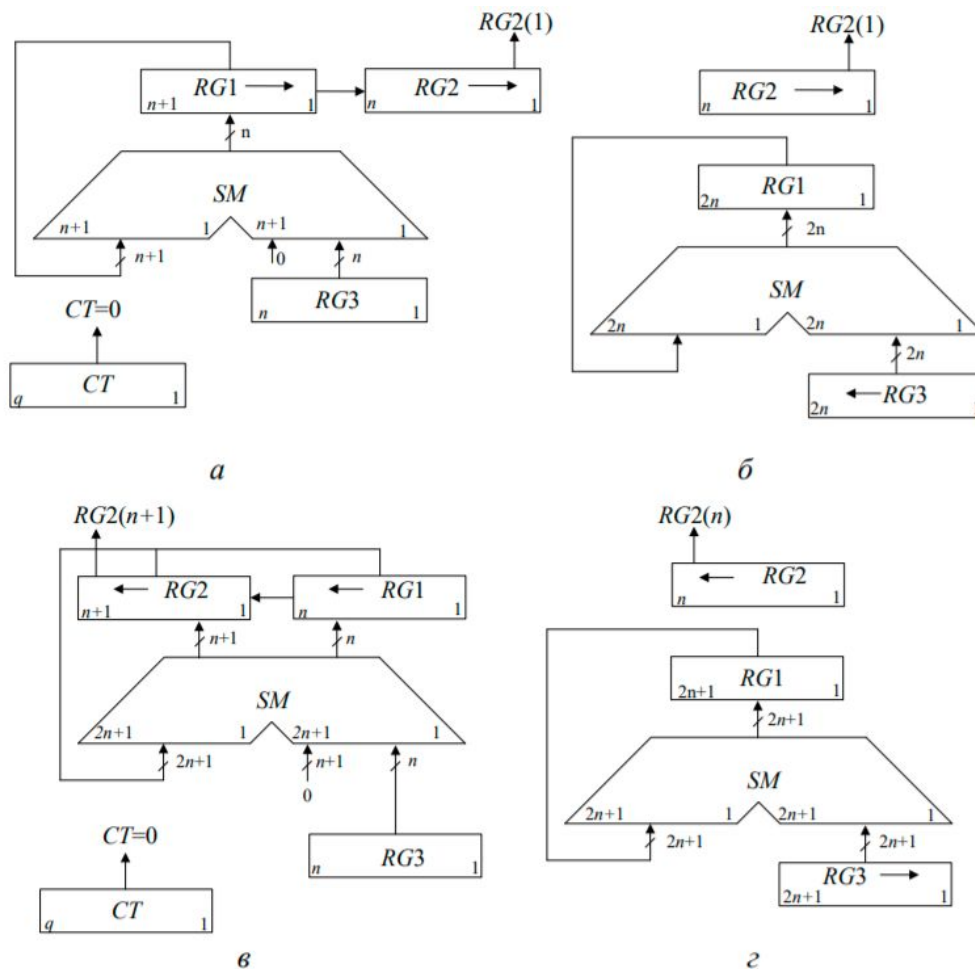
## 3.5 Множення чисел

При множенні чисел у прямих кодах знакові та основні розряди обробляються роздільно. Для визначення знака добутку здійснюють підсумовування за модулем 2 цифр, записаних в знакових розрядах співмножників.

Розрізняють чотири способи множення.



Множення виконується зі старших розрядів множника, сума часткових добутків залишається нерухомою, а множене зсувається вправо. Принцип побудови пристроїв, що реалізують різні способи множення, показаний на рисунку, де RG3 – реєстр множеного, RG1 – реєстр добутку, RG2 – реєстр множника. Цифрами зазначені номери розрядів SM і реєстрів, а стрілками показаний напрямок зсуву кодів у реєстрах.



Операційні схеми пристроїв для множення чисел: а – перший спосіб; б – другий спосіб; в – третій спосіб; г – четвертий спосіб

Під час **множення першим способом (шкільний спосіб)** в першому такті  $i$ -го циклу аналізується значення  $RG2[1]$  – молодшого ( $n$ -го) розряду реєстру RG2, в якому знаходиться чергова цифра множника. Вміст RG3 додається до суми часткових добутків, що знаходяться в реєстрі RG1, якщо  $RG2[1]=1$ , або не додається, якщо  $RG2[1]=0$ . В другому такті здійснюється правий зсув у реєстрах

RG1 і RG2, що еквівалентно множенню їхнього вмісту на  $2^{-1}$ . При зсуві цифра молодшого розряду регістру RG1 записується у вивільнюваний старший розряд регістру RG2. Після виконання  $n$  циклів молодші розряди  $2n$ -розрядного добутку будуть записані в регістр RG2, а старші – у RG1.

### Вираз множення 1 способом

$$\Pi = A \times B = A \times (b_{n-1} p^{n-1} + \dots + b_1 p^1 + b_0 p^0) = A p^{n-1} b_{n-1} + \dots + A p^1 b_1 + A p^0 b_0$$

Приклад

		A	1	0	1	1		
		B	1	1	0	1		
$b_0 = 1$			1	0	1	1		$A \cdot 2^0 b_0$
$b_1 = 0$		0	0	0	0			$A \cdot 2^1 b_1$
$b_2 = 1$		1	0	1	1			$A \cdot 2^2 b_2$
$b_3 = 1$		1	0	1	1			$A \cdot 2^3 b_3$
$\Pi =$	1	0	0	0	1	1	1	1
		$\vdots$		$n$	$\vdots$		$n$	$\vdots$

Перед початком **множення другим способом** множник  $X$  записують в регістр RG2, а множене  $Y$  – в молодші розряди регістру RG3 (тобто в регістрі RG3 установлюють  $Y_0 = Y 2^{-n}$ ). В кожному  $i$ -му циклі множення додаванням кодів RG3 і RG1 управляє цифра RG2[1], а в регістрі RG3 здійснюється зсув вліво на один розряд, в результаті чого формується величина  $Y_i = 2Y_{i-1}$ . Оскільки сума часткових добутків в процесі множення нерухома, зсув в регістрі RG3 можна виконати суміщення в часі з підсумовуванням (як правило,  $t_p \geq t_z$ ). В цьому випадку  $t_m = n t_p$ . Завершення операції множення визначається за нульовим вмістом регістру RG2, що також приводить до збільшення швидкодії, якщо множник ненормалізований.

### Вираз множення 2 способом

$$\Pi = A \times B = p^n \{((\dots((0 + A \cdot b_0) p^{-1} + A \cdot b_1) p^{-1} + \dots + A \cdot b_{n-2}) p^{-1} + A \cdot b_{n-1}) p^{-1}\}$$

Приклад

[illegible]

Результат множення  $\Pi = \{\Pi_4 \cdot 2^{-1}\} \cdot 2^4 = \{1000, 1111\} \cdot 2^4 = 10001111$ .

Тут множення починається з молодших розрядів і зсувається праворуч сума часткових добутків.

При **множенні третім способом** множник  $X$  записується в старші розряди  $RG2$ , при цьому  $RG2[1]=0$ . Вага молодшого розряду  $RG3$  дорівнює  $2^{-2n}$ , тому код в регістрі  $RG3$  являє собою значення  $Y 2^{-n}$ . В кожному циклі множення підсумовування виконується при  $RG2[n+1]=1$ . В регістрах  $RG1$  і  $RG2$  виконується лівий зсув. В результаті підсумовування вмісту  $RG3$  і  $RG1$  може виникнути перенос в молодший розряд регістру  $RG2$ , що реалізується на  $SM$ . Збільшення довжини  $RG2$  на один розряд усуває можливість поширення переносу в розряди множника. Після виконання  $n$  циклів молодші розряди добутку будуть знаходитися в регістрі  $RG1$ , а старші – в регістрі  $RG2$ . Час множення третім способом визначається аналогічно першому способу.

### Вираз множення 3 способом

$$\Pi = A \times B = p^n \{A \cdot p^{-1} b_{n-1} + A \cdot p^{-2} b_{n-2} + \dots + A \cdot p^{-n+1} b_1 + A \cdot p^{-n} b_0\}$$

## Приклад

A	1	0	1	1				
B	1	1	0	1				
$b_3 = 1$	0	1	0	1	1			$A \cdot 2^{-1} b_3$
$b_2 = 1$	0	0	1	0	1	1		$A \cdot 2^{-2} b_2$
$b_1 = 0$	0	0	0	0	0	0	0	$A \cdot 2^{-3} b_1$
$b_0 = 1$	0	0	0	0	1	0	1	$A \cdot 2^{-4} b_0$
	1	0	0	0	1	1	1	
	n				n			

Результат множення  $\Pi = \{1000, 1111\} \cdot 2^4 = 10001111$

Перед **множенням четвертим способом** множник записують в регістр RG2, а множене – в старші розряди регістру RG3 (тобто в RG3 установлюють  $Y_0 = Y \cdot 2^{-1}$ ). В кожному циклі цифра RG2[n+1], що знаходиться в старшому розряді регістру RG2, управляє підсумовуванням, а в RG3 здійснюється правий зсув на один розряд, що еквівалентно множенню вмісту цього регістра на  $2^{-1}$ . Час виконання множення четвертим способом складає  $t_m = n \cdot t_p$ , визначається аналогічно другому способу.

### Вираз множення 4 способом

$$\Pi = A \times B = (\dots((0 + A \cdot b_{n-1}) p^{+1} + A \cdot b_{n-2}) p^{+1} + \dots + A \cdot b_1) p^{+1} + A \cdot b_0$$

A	1	0	1	1				
B	1	1	0	1				
		1	0	1	1			$0 + A \cdot b_3 = \Pi_1$
	1	0	1	1	0			$\Pi_1 \cdot 2$
		1	0	1	1			$A \cdot b_2$
	1	0	0	0	0	1		$\Pi_1 \cdot 2 + A \cdot b_2 = \Pi_2$
	1	0	0	0	0	1	0	$\Pi_2 \cdot 2$
		0	0	0	0			$A \cdot b_1$
	1	0	0	0	0	1	0	$\Pi_2 \cdot 2 + A \cdot b_1 = \Pi_3$
	1	0	0	0	0	1	0	$\Pi_3 \cdot 2$
		1	0	1	1			$A \cdot b_0$
	1	0	0	0	1	1	1	$\Pi = \Pi_3 \cdot 2 + A \cdot b_0$
	n				n			

## **Розділ 4. Мікроконтролери**

### **4.1 MSP**

Серійний протокол MultiWii є широко розповсюдженим стандартом для зв'язку з диспетчерами польоту через UART. MSP спочатку був частиною прошивки контролера польоту MultiWii. Цей протокол передачі даних використовується в дронах, радіокерованих літаках, коптерах і інших пристроях

MSP - це протокол, розроблений спільнотою MultiWii, з ідеєю бути легким, загальним, ефективним, безпечним. Протокол створений як частина програмного забезпечення multiwii. Дозволяє передавати до 8 каналів по одному дроту.

MSP є видом протоколу послідовної передачі даних. Послідовна передача цифрових даних використовує 3 дроти (сигнал, земля і споживання) для передачі безлічі каналів (та й взагалі будь-яких даних). Цей тип передачі вимагає наявності послідовного порту як на приймачі, так і на польотному контролері.

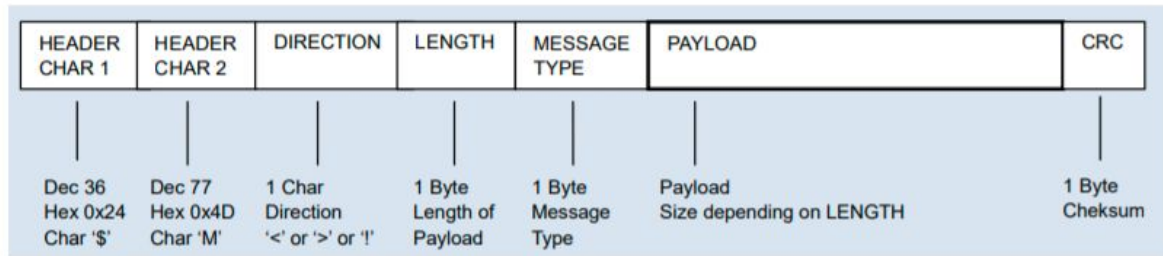
#### **Основні характеристики протоколу MSP (Multiwii Serial Protocol)**

Серійний протокол MultiWii має такі ключові особливості:

- Компактний - використовує 8-бітові двійкові дані.
- Загальний - його можна прозоро використовувати за допомогою графічного інтерфейсу, екранного меню, телеметрії або саморобних інструментів конфігурації, тобто жоден конкретний код екранного меню не повинен кодуватися в MultiWii
- Захищена контрольна сума - дані надсилаються з контрольною сумою, що запобігає введенню пошкодженої конфігурації.
- Чутливий заголовок - оскільки він розроблений з певним заголовком, його можна змішувати з іншим кадром, наприклад, рамкою GPS, тобто можна буде підключити або графічний інтерфейс, або GPS на одному послідовному порту без зміни конфігурації.

#### **Структура пакетів MSP**

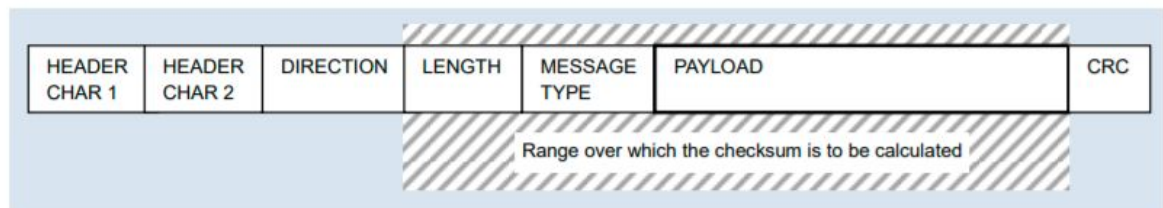
Базовий пакет MSP виглядає так:



- Кожне повідомлення починається з двох символів: "\$" і "M"
- Далі наводиться поле 'Direction', яке містить один символ. Це поле визначає напрямок повідомлення.
- Наступне поле довжиною в 1 байт. 'Length' визначається як довжина лише корисного навантаження. Він не включає поля Header Chars, Direction, Length, MessageType і CRC. Може становити від 0 до 255. Програмне забезпечення MultiWii версії 2.1 не може обробити більше 64 байт.
- Корисне навантаження - це поле змінної довжини, воно може бути порожнім, якщо поле Довжина дорівнює нулю.
- CRC - це 8-бітова контрольна сума, обчислення якої визначено нижче.

### Контрольна сума MSP

Контрольна сума обчислюється за пакетом, починаючи і включаючи поле LENGTH, аж до поля контрольної суми, але без його урахування:



Контрольна сума обчислюється з використанням XOR-функції



### Потік повідомлень MSP

Залишається головне правило: MultiWii ніколи не надсилає щось самостійно. У кожному випадку потрібно зробити запит на отримання або встановлення даних. Кожне отримане повідомлення підтверджується, навіть якщо всередині даних немає.

Навіть у випадку помилки CRC або поганого формату повідомлення, MultiWii не надсилає щось назад.

### Запит повідомлення до MultiWii

Для запиту простих даних без параметрів / надсилання певної команди / введення нових параметрів у MultiWii потрібно надіслати повідомлення з полем напрямку, що містить ASCII char '<' (Dec 60, Hex 0x3C).

### Вихідне повідомлення MultiWii

Це повідомлення надсилається в результаті повідомлення-запиту. Коли MultiWii отримає правильне повідомлення із запитом, воно буде оброблено, і MultiWii надішле назад повідомлення з полем Напрямку, яке містить ASCII char '>' (Dec 62, Hex 0x3E), те саме поле Типу повідомлення, що й у повідомленні-запиті, поле нульової довжини та відсутність корисного навантаження.

### Невідомий тип повідомлення

Це повідомлення надсилається в результаті повідомлення-запиту з невідомим типом повідомлення. Коли MultiWii отримав правильне

повідомлення-запит, але з невідомим полем типу повідомлення, MultiWii надсилає назад повідомлення з полем напрямку, що містить ASCII char '!' (Dec 33, Hex 0x21), те саме поле Типу повідомлення, що й у запиті повідомлення, поле нульової довжини та відсутність корисного навантаження.

Приклад MSP\_STATUS:

<b>Message</b>		MSP_STATUS		
<b>Description</b>		Request cycletime, errors_count, sensor present, box activation, current setting number		
<b>Type</b>		Get		
<b>Comment</b>		No payload		
<b>Offset</b>	<b>Data</b>	<b>Name</b>	<b>Unit</b>	<b>Description</b>
0	'\$'	Header char 1		Dec 36, Hex 0x24
1	'M'	Header char 2		Dec 77, Hex 0x4D
2	'<'	Direction		Dec 60, Hex 0x3C
3	0	Length		No payload
4	101	Message Code		MSP_STATUS, Dec 101, Hex 0x65
5	101	CRC		Dec 101, Hex 0x65

## 4.2 Arduino

**Arduino** є платформою прототипування електроніки з відкритим вихідним кодом, заснована на гнучких, легких у використанні апаратних засобах і програмному забезпеченні.

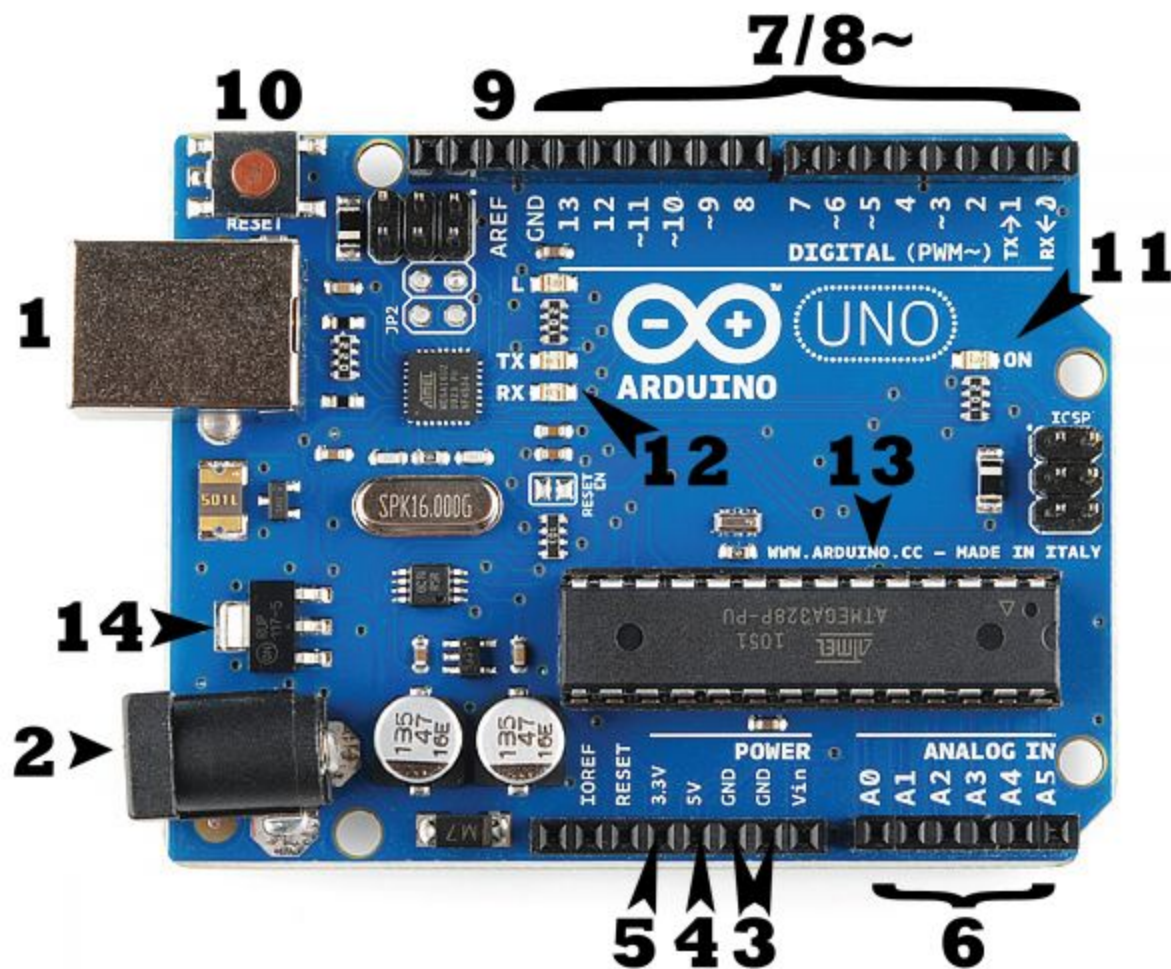
Протягом багатьох років Arduino був мозком тисячі проектів, від повсякденних об'єктів до складних наукових інструментів. Всесвітнє співтовариство виробників – студенти, любителі, художники, програмісти та професіонали - зібралися навколо цієї платформи з відкритим кодом, їх внесок доповнив неймовірний обсяг доступних знань, які сьогодні можуть бути корисними як для новачків, так і для експертів.

Arduino народився в Інституті дизайну взаємодії Ivrea як простий інструмент для швидкого створення прототипів, орієнтований на студентів, які не мають досвіду в галузі електроніки та програмування. Як тільки платформа Arduino потрапила до ширшої спільноти, вона почала змінюватися, щоб адаптуватися до нових потреб і викликів, диференціюючи свою пропозицію від простих 8-бітних плат до продуктів для додатків IoT, носимих пристроїв, пристроїв для 3D-друку та вбудованих середовищ. Усі плати Arduino повністю відкриті, що дає можливість



користувачам створювати їх самостійно та врешті-решт адаптувати їх до їхніх конкретних потреб. Програмне забезпечення теж є відкритим, і воно зростає завдяки внеску користувачів по всьому світу.

### Основні елементи апаратного забезпечення Arduino Uno



- Мікроконтролер аналог мікропроцесора в звичайному ПК;
- Кнопка скидання здійснює скидання мікроконтролера і повторити запуск програми;
- Порт USB забезпечує зв'язок з ПК і живлення пристрою;
- Світлодіод No13 світлодіод, з'єднаний з цифровим виходом No13;
- Живлення +9 додаткове живлення від зовнішнього джерела (батарея, блок живлення)

## **Живлення (USB / Barrel Jack)**

Кожна плата Arduino потребує способу підключення до джерела живлення. Arduino UNO може харчуватися від USB-кабелю, що надходить від вашого комп'ютера, або від настінного джерела живлення (наприклад, такого), що закінчується в гнізді. На малюнку вище з'єднання USB позначено (1), а гніздо стовбура - (2).

Підключення через USB - це також спосіб завантаження коду на плату Arduino.

**ПРИМІТКА.** НЕ використовуйте джерело живлення більше 20 Вольт, оскільки ви пересилите (і тим самим знищите) ваш Arduino. Рекомендована напруга для більшості моделей Arduino становить від 6 до 12 Вольт.

## **Штифти (5V, 3,3V, GND, Аналог, Цифрові, PWM, AREF)**

Штифти на вашому Arduino - це місця, де ви підключаєте дроти для побудови схеми (можливо, у поєднанні з макетною дошкою та деяким дротом). Вони, як правило, мають чорні пластикові "заголовки", які дозволяють просто підключити провід прямо до плати. Має кілька різних видів шпильок, кожен з яких позначений на дошці та використовується для різних функцій.

**GND (3):** скорочення від "Ground". На Arduino є кілька штифтів GND, будь-який з яких можна використовувати для заземлення ланцюга.

**5V (4) та 3,3V (5):** Як ви можете здогадатися, 5-контактний штир подає 5 вольт, а 3,3V - 3,3 вольта. Більшість простих компонентів, що використовуються з Arduino, радісно працюють від 5 або 3,3 вольт.

**Аналог (6):** Площа штифтів під міткою „Analog In” (від A0 до A5 на UNO) є штифтами Analog In. Ці висновки можуть зчитувати сигнал з аналогового датчика (наприклад, датчика температури) і перетворювати його в цифрове значення, яке ми можемо прочитати.

**Цифрові (7):** навпроти аналогових штифтів розташовані цифрові штифти (від 0 до 13 на UNO). Ці висновки можуть використовуватися як для цифрового

введення (наприклад, для повідомлення про натискання кнопки), так і для цифрового виводу (наприклад, для живлення світлодіода).

**PWM (8):** Можливо, ви помітили тильду (~) біля деяких цифрових штифтів (3, 5, 6, 9, 10 та 11 на UNO). Ці висновки діють як звичайні цифрові висновки, але їх також можна використовувати для чогось, що називається імпульсно-ширинна модуляція (PWM).

**AREF (9):** Підставки для аналогових посилок. Найчастіше ви можете залишати цю шпильку в спокої. Іноді його використовують для встановлення зовнішньої опорної напруги (від 0 до 5 Вольт) як верхньої межі для аналогових вхідних контактів.

### **Кнопка скидання**

Як і оригінальний Nintendo, Arduino має кнопку скидання (10). Якщо його натиснути, тимчасово з'єднати штифт скидання із землею та перезапустити будь-який код, завантажений на Arduino. Це може бути дуже корисно, якщо ваш код не повторюється, але ви хочете перевірити його кілька разів. На відміну від оригінального Nintendo, видування на Arduino зазвичай не вирішує жодних проблем.

### **Світлодіодний індикатор живлення**

Неподалік і праворуч від слова "UNO" на друкованій платі поруч із словом "ON" (11) є крихітний світлодіод. Цей світлодіод повинен загорятися кожного разу, коли ви підключаєте Arduino до джерела живлення. Якщо це світло не вмикається, є велика ймовірність, що щось не так. Час перевірити свою схему!

### **Світлодіоди TX і RX**

TX - це скорочення для передачі, RX - скорочення для прийому. Цього маркування досить мало зустрічається в електроніці для позначення контактів, відповідальних за послідовний зв'язок. У нашому випадку на Arduino UNO є два місця, де з'являються TX і RX - один раз цифровими штифтами 0 і 1, а вдруге поруч із світлодіодними індикаторами TX і RX (12). Ці світлодіоди дадуть нам кілька приємних візуальних вказівок кожного разу, коли наш Arduino отримує або передає дані (наприклад, коли ми завантажувемо нову програму на плату).

### **Головна IC**

Чорна річ з металевими ніжками - це IC, або інтегральна схема (13). Подумайте про це як про мізки нашого Arduino. Основна мікросхема на Arduino дещо відрізняється від типу плати до типу плати, але, як правило, від лінійки мікросхем ATmega від компанії ATMEL. Це може бути важливим, оскільки вам може знадобитися знати тип мікросхеми (разом із типом вашої плати) перед завантаженням нової програми із програмного забезпечення Arduino. Цю інформацію зазвичай можна знайти в письмовій формі на верхній стороні IC.

### **Регулятор напруги**

Регулятор напруги (14) насправді не є тим, з чим ви можете (або повинні) взаємодіяти на Arduino. Але потенційно корисно знати, що воно є і для чого воно потрібне. Регулятор напруги робить саме те, що він говорить - він контролює величину напруги, яка надходить на плату Arduino. Сприймайте це як свого роду воротаря; це відверне додаткову напругу, яка може завдати шкоди ланцюгу. Звичайно, це має свої межі, тому не підключайте ваш Arduino до нічого, що перевищує 20 вольт.

У платі Arduino UNO доступно 14 цифрових входів/виходів, 6 із яких можуть видавати PWM сигнал, і 6 аналогових входів. Ці сигнали доступні на платі через контактні площадки або штирьові роз'єми. Також існує багато видів зовнішніх плат розширення, які називаються "shields" ("щити"), які приєднуються до плати Arduino через штирьові роз'єми.

Arduino може використовуватися як для створення автономних інтерактивних об'єктів, так і підключатися до програмного забезпечення, яке виконується на комп'ютері (наприклад: Adobe Flash, Processing, Max/MSP, Pure Data, SuperCollider).

### **Переваги Arduino**

Завдяки простому та доступному користувачеві досвіду Arduino використовується в тисячах різних проектів та додатків. Програма Arduino проста у використанні для початківців, але досить гнучка для досвідчених користувачів. Він працює на Mac, Windows і Linux. Викладачі та студенти використовують його для побудови недорогих наукових інструментів, для доведення принципів хімії та

фізики або для початку роботи з програмуванням та робототехнікою. Дизайнери та архітектори створюють інтерактивні прототипи, музиканти та художники використовують її для інсталяцій та експериментів з новими музичними інструментами. Виробники, звичайно, використовують її для побудови багатьох проектів, виставлених на Maker Faire, наприклад. Arduino - ключовий інструмент для вивчення нових речей.

Є багато інших мікроконтролерів та платформ мікроконтролерів, доступних для фізичних обчислень. Parallax Basic Stamp, Netmedia BX-24, Phidgets, MIT's Handyboard та багато інших пропонують подібну функціональність. Усі ці інструменти мають безладні деталі програмування мікроконтролера та обертають їх у простий у використанні пакет. Arduino також спрощує процес роботи з мікроконтролерами, але він пропонує певну перевагу для вчителів, студентів та зацікавлених аматорів перед іншими системами:

- *Невисока ціна*

Плати Arduino відносно недорогі в порівнянні з іншими платформами мікроконтролера. Найдешевшу версію модуля Arduino можна зібрати вручну, і навіть попередньо зібрані модулі Arduino коштують менше 50 доларів.

- *Кроссплатформенність*

Програмне забезпечення Arduino (IDE) працює на операційних системах Windows, Macintosh OSX та Linux. Більшість систем мікроконтролера обмежені Windows.

- *Просте та зрозуміле середовище програмування*

Програмне забезпечення Arduino (IDE) є простим у користуванні для початківців, але при цьому досить гнучким для просунутих користувачів. Для вчителів це зручно базуватися на середовищі програмування Processing, тому студенти, які навчаються програмуванню в цьому середовищі, будуть знайомі з тим, як працює Arduino IDE.

- *Розширюване програмне забезпечення з відкритим кодом*

Програмне забезпечення Arduino публікується як інструмент з відкритим кодом, доступне для розширення досвідченими програмістами. Мову можна

розширити за допомогою бібліотек C ++, і люди, які хочуть зрозуміти технічні деталі, можуть зробити перехід від Arduino до мови програмування AVR C, на якій вона базується. Подібним чином ви можете додати код AVR-C безпосередньо у свої програми Arduino, якщо хочете.

- *Розширюване апаратне забезпечення з відкритим кодом*

Плати Arduino публікуються під ліцензією Creative Commons, тому досвідчені дизайнери схем можуть зробити власну версію модуля, розширивши або удосконаливши його. Навіть відносно не досвідчені користувачі можуть створити макетну версію модуля, щоб зрозуміти, як він працює, і заощадити гроші.

### **Програмне забезпечення Arduino**

Інтегроване середовище розробки Arduino – це багатоплатформний додаток на Java, що включає в себе редактор коду, компілятор і модуль передачі прошивки в плату. Середовище розробки засноване на мові програмування Processing та спроектована для програмування новачками, незнайомими близько з розробкою програмного забезпечення. Мова програмування аналогічна мові Wiring. Строго кажучи, це C ++, доповнений деякими бібліотеками. Програми обробляються за допомогою препроцесора, а потім компілюється за допомогою AVR-GCC.

Програми Arduino пишуться на мові програмування C або C++. Середовище розробки Arduino поставляється разом із бібліотекою програм «Wiring» (бере початок від проекту Wiring, який дозволяє робити багато стандартних операцій вводу/виводу набагато простіше). Користувачам необхідно визначити лише дві функції для того, щоб створити програму, яка буде працювати за принципом циклічного виконання:

- `setup()`: функція виконується лише раз при старті програми і дозволяє задати початкові параметри
- `loop()`: функція виконується періодично, доки плата не буде вимкнена

Типова найпростіша програма для мікроконтролера, яка посилає команду блимати світловому діоду в середовищі Arduino, буде виглядати так:

```
#define LED_PIN 13

void setup () {
  pinMode (LED_PIN, OUTPUT); // Ввімкнути контакт 13 для цифрового виводу
}

void loop () {
  digitalWrite (LED_PIN, HIGH); // Ввімкнути світлодіод
  delay (1000); // Зачекати одну секунду (1000 мілісекунд)
  digitalWrite (LED_PIN, LOW); // Вимкнути світлодіод
  delay (1000); // Зачекати одну секунду
}
```

У прикладі програми використовується конструктивна особливість більшості плат Arduino, які мають вбудований світлодіод з резистором навантаження, підключений між 13-м контактом і землею, що є зручним для багатьох простих тестів.