МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

І. П. Голубєва, В. А. Казміренко, Ю. В. Прокопенко

Інформатика

Збірник задач

Навчальний посібник

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра
за освітньою програмою «Електронні мікро- і наносистеми та технології»
спеціальності 176 Мікро- та наносистемна техніка

Електронне мережеве навчальне видання

Київ
КПІ ім. Ігоря Сікорського
2024

УДК 004.021 (045)
Г60

Автори: *Голубєва Ірина Петрівна*, канд. техн. наук, доц.
Казміренко Віктор Анатолійович, канд. техн. наук, доц.
Прокопенко Юрій Васильович, д-р техн. наук, проф.

Рецензент *Діденко Ю. В.*, канд. техн. наук, доц.,
доцент кафедри мікроелектроніки, КПІ ім. Ігоря Сікорського

Відповідальний редактор *Тимофєєв В. І.*, д-р техн. наук, проф.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № X від DD.MM.YYYY р.)
за поданням Вченої ради факультету електроніки
(протокол № X від DD.MM.YYYY р.)

Г60 **Голубєва І. П.**

Інформатика: збірник задач [Електронний ресурс]: навч. посіб. для здобувачів ступеня бакалавра за освіт. програмою «Електронні мікро- і наносистеми та технології» спец. 176 Мікро- та наносистемна техніка / І. П. Голубєва, В. І. Казміренко, Ю. В. Прокопенко; КПІ ім. Ігоря Сікорського. – Електрон. текст. дані (1 файл). – Київ : КПІ ім. Ігоря Сікорського, 2024. – 117 с.

Викладено відомості про системи числення та переходи між ними, способи зберігання цілих та дійсних чисел, правила арифметичних операцій над ними, поняття про логічні величини, логічні функції та основи булевої алгебри. Містить варіанти індивідуальних завдань, детальний опис методів їх розв'язання та приклади.
Призначений для здобувачів ступеня бакалавра за спеціальністю 176 Мікро- та наносистемна техніка. Буде також корисним здобувачам ступеня бакалавра за спеціальностями з галузі знань 17 Електроніка, автоматизація та електронні комунікації.

УДК 004.021 (045)

Реєстр. № НП XX/XX-XXX. Обсяг 5,32 авт. арк.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
проспект Перемоги, 37, м. Київ, 03056
https://kpi.ua
Свідоцтво про внесення до Державного реєстру видавців, виготовлювачів
і розповсюджувачів видавничої продукції ДК № 5354 від 25.05.2017 р.

Зміст

ВСТУП 5
1. Системи числення 7
1.1. Теоретичні відомості 7
1.1.1. Основні поняття 7
1.1.2. Основні алгоритми переходу до інших систем числення для цілих чисел 12
1.1.3. Алгоритм переходу від десяткової системи числення до двійкової для чисел, які мають дробову частину 25
1.2. Завдання 30
1.3. Контрольні питання 36
2. Зберігання цілих чисел в цифрових системах 37
2.1. Теоретичні відомості 37
2.1.1. Представлення цілих беззнакових чисел в цифрових системах 39
2.1.2. Представлення цілих знакових чисел в цифрових системах 43
2.1.3. Представлення цілих знакових чисел в цифрових системах: прямий, обернений та додатковий коди додатних чисел 45
2.1.4. Запис цілих знакових чисел в цифрових системах: прямий код від'ємних чисел 47
2.1.5. Представлення цілих знакових чисел в цифрових системах: обернений код від'ємних чисел 49
2.1.6. Представлення цілих знакових чисел в цифрових системах: додатковий код від'ємних чисел 51
2.2. Завдання 55
2.3. Контрольні питання 58
3. Арифметичні операції над цілими числами 59
3.1. Теоретичні відомості 59
3.1.1. Арифметичні операції над цілими числами без знаку 59
3.1.2. Арифметичні операції над цілими числами в прямому коді із знаком 61
3.1.3. Арифметичні операції над цілими числами в оберненому коді 63
3.1.4. Арифметичні операції над цілими числами в додатковому коді 67
3.1.5. Помилки у процесі виконання арифметичних операцій на комп'ютері 69
3.2. Завдання 71

3.3. Контрольні питання 74
4. Зберігання дійсних чисел в цифрових системах 75
4.1. Теоретичні відомості 75
4.2. Завдання 84
4.3. Контрольні питання 87
5. Машинна логіка 88
5.1. Теоретичні відомості 88
5.1.1. Інверсія 90
5.1.2. Кон'юнкція 92
5.1.3. Диз'юнкція 94
5.1.4. Виключне АБО 96
5.1.5. Пріоритет виконання логічних функцій 98
5.1.6. Основні закони алгебри логіки 99
5.2. Завдання 105
5.3. Контрольні питання 108
6. ВИСНОВКИ 109
7. ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 110
8. Додаток А. Основні положення стандарту IEEE 754 111
8.1.1. Формат половинної точності: Half-precision floating-point format 112
8.1.2. Формат одинарної точності: Single-precision floating-point format 114
8.1.3. Формат подвійної точності: Double-precision floating-point format 115
8.1.4. Формат чотирикратної точності: Octuple-precision floating-point format 117

ВСТУП

Обчислювальна техніка є незамінним робочим інструментом сучасного інженера у будь-якій галузі, зокрема і у галузі електроніки. Натепер розроблено велику кількість програмного забезпечення, як загального призначення, наприклад, для розв'язання задач лінійної алгебри (Linpack, LAPACK), так і спеціалізованих систем проектування провідних виробників (Synopsys, Cadence Design Systems, Mentor Graphics). Прикладна інформація у таких системах зберігається у числовому вигляді, а, відповідно, перетворення такої інформації зводяться до арифметичних та логічних операцій над числами. Як правило, інженеру немає потреби виконувати ті чи інші обчислення вручну, натомість слід скористатися готовим інструментом, виконавши належні налаштування.

Переважна більшість здобувачів ступеня бакалавра по закінченню шкільної програми з інформатики мають певне уявлення про системи числення, будову обчислювальної машини та загальні принципи її роботи. Разом з тим, поза увагою залишаються особливості зберігання чисел та дій над ними. Розуміння перетворень, які відбуваються під час введення інформації до машини, виконання обчислень та виводу результатів, дозволяє висувати обґрунтовані вимоги до обчислювальних процедур, виявляти їх обмеження, критично аналізувати отримані результати.

Окреслені питання є підґрунтям для подальшого успішного засвоєння інформатики та дисциплін, пов'язаних з інформаційними технологіями. Для перевірки засвоєння вступного модуля з інформатики пропонуються контрольні роботи. У цьому посібнику наведено теоретичний матеріал та довідкові відомості, які стануть у нагоді для підготовки до контрольних робіт, детально розглянуто велику кількість прикладів за усіма задачами, наведено індивідуальні завдання до виконання та контрольні запитання для самоперевірки.

Зміст контрольних робіт не має надмірної трудомісткості, їх виконання не потребує застосування обчислювальної техніки.

Розглянуто питання про системи числення, загальні та спеціальні алгоритми переходу від однієї системи числення до іншої.

Наведено найпоширеніші способи кодування, зберігання цілих та дійсних чисел та операцій над ними. У додатку наведено довідковий матеріал щодо особливостей зберігання дійсних чисел.

1. Системи числення

Мета роботи: навчитися виконувати перехід із однієї системи числення в іншу.

1.1. Теоретичні відомості

1.1.1. Основні поняття

Число – це абстрактна сутність, міра для опису кількості чого-небудь.

Цифри – це система знаків для запису конкретних значень чисел. Оскільки чисел набагато більше ніж цифр, то для запису числа зазвичай використовується набір (комбінація) цифр. Існують різні системи знаків для запису чисел.

Система числення – це сукупність прийомів позначення чисел – мова, алфавітом якої є символи (цифри), а синтаксисом – правило, що дозволяє сформулювати запис чисел однозначно.

За весь час існування людства було винайдено багато різних систем числення. Однак усі системи числення можна поділити на два великих класи – позиційні та непозиційні системи числення.

Непозиційна система числення – це така система, в якій значення символа (цифри) чітко визначено і не залежить від її положення під час запису числа. Прикладами непозиційних систем числення є унарна система числення, римська система числення та інші.

В *позиційній системі* навпаки – значення кожної цифри залежить від її позиції (розряду) в числі і вага кожної цифри змінюється в залежності від її позиції. До позиційних систем числення відноситься звична людям десяткова система, яка базується на арабських цифрах.

Історично найпершою виникла так звана унарна система числення. Це непозиційна система числення з єдиною цифрою, яка означає 1. За одиницю використовували вузлики, насічки, камінці.

Унарна система числення була проста, але вона ставала незручною тоді, коли потрібно було рахувати велику кількість предметів. Тому з розвитком економіки стародавнього світу виникає потреба якось позначати певну кількість предметів компактним символом. З цієї причини у різних культурах виникають різні системи знаків запису чисел – нові непозиційні системи числення.

Непозиційними були системи числення були у древніх єгиптян, греків та римлян. Непозиційні системи числення були більш-менш придатні для виконання операцій додавання та віднімання, але зовсім не зручні для множення та ділення.

В позиційній системі значення кожної цифри залежить від її позиції та вага кожної цифри змінюється в залежності від її позиції.

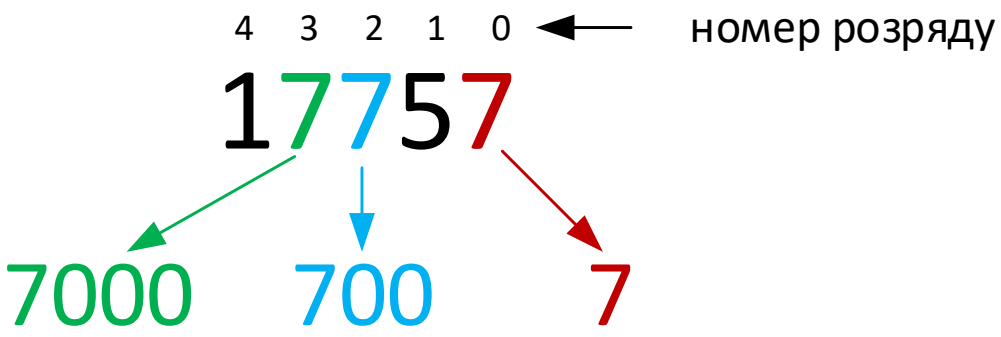


Рис. 1.1. Приклад запису числа в позиційній десятковій системі числення.

У наведеному на рис. 1.1 прикладі цифра «7» зустрічається три рази, причому у кожній з позицій ця цифра означає різну кількість:

Права (червона) цифра означає число 7;

Середня (синя) цифра означає 700;

Ліва (зелена) цифра означає 7000.

У позиційних системах числення окрему позицію в зображенні числа прийнято називати *розрядом*, а номер позиції – *номером розряду*. Число розрядів у запису числа називається *розрядністю* і збігається з довжиною числа.

Діапазоном представлення чисел D в системі числення – це інтервал числової осі між мінімальним та максимальним числами, представленими заданою кількістю розрядів.

Позиційні системи числення поділяються на два великих класи – однорідні та змішані. В *однорідних позиційних системах числення* для всіх розрядів числа набір допустимих символів – однаковий. Прикладом однорідної позиційної системи числення є загальноприйнята десяткова система числення, що використовує для запису чисел десять цифр від 0 до 9.

В *змішаній позиційній системі числення* – в кожному розряді числа набір допустимих символів (цифр) може відрізнятися від наборів інших розрядів. Прикладом змішаної системи числення може служити система вимірювання часу: у розряді секунд і хвилин можливі 60 різних символів (від «00» до «59»), а в розряді годин – 24 різних символи (від «00» до «23»),

В однорідних позиційних систем числення *основою системи числення (p)* називається кількість цифр та символів, що застосовуються для зображення числа. Наприклад, у десятковій системі числення $p = 10$. Оскільки найменшою цифрою є нуль, то найбільша цифра у системі числення (9) на одиницю менша, ніж кількість цифр ($p-1=10-1=9$).

Назва системи безпосередньо залежить від величини p (основи системи числення): якщо $p = 10$ маємо десяткову систему числення,

якщо $p = 2$ - двійкову систему числення і так далі.

Базою, або алфавітом системи числення називають послідовність цифр, які використовуються для запису числа. Якщо основа системи числення більше 10, тобто загальноприйнятих цифр не вистачає, використовують інші символи. У таблиці 1.1 наведені бази систем числення із основою 2, 4, 8, 10 та 16.

Таблиця 1.1. Бази систем числення із основою 2, 4, 8, 10 та 16.

	Основа системи числення (p)	База системи числення
Двійкова система числення	2	0, 1
Система числення із основою 4	4	0, 1, 2, 3
Система числення із основою 8	8	0, 1, 2, 3, 4, 5, 6, 7
Десяткова система числення	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Шістнадцяткова система числення	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

У сучасному світі найбільшого поширення набула десяткова система числення. Однак, в інженерії та комп'ютерних науках широко використовуються системи числення, основа яких кратна степеням двійки: $p = 2^1 = 2$, $p = 2^3 = 8$, $p = 2^4 = 16$. Це пов'язано з тим, що велика частина сучасної техніки побудована за цифровою технологією, де основним елементом побудови є цифровий ключ, що має 2 стійких положення, які прийнято позначати 0 і 1 (низький та високий рівень). Іншою незаперечною перевагою двійкової системи числення є те, що зберігання та передача інформації за допомогою тільки двох станів є надійною та стійкою до перешкод.

Однак двійкова система числення має недолік – швидке зростання кількості розрядів, необхідних для запису чисел, тому в інформатиці також широко використовують вісімкову та шістнадцяткову системи числення, оскільки запис чисел в цих системах значно коротший, ніж запис чисел в двійковій системі. До того ж, перехід від двійкової системи числення до

вісімкової та шістнадцяткової систем та назад дуже простий та інтуїтивно зрозумілий.

Формула (1.1) – це загальний формат запису чисел в позиційних системах числення:

$$x_p = a_{n-1}a_{n-2}\ldots a_0\,, \tag{1.1}$$

де x_p – число, представлене в системі числення із основою p ,

p – основа системи числення,

a_i – i – тий розряд числа,

n – загальна кількість розрядів в числі.

Приклад 1.1.

$x_p = 375_{10}$

$p = 10 \quad n = 3$

$a_0 = 5, \quad a_1 = 7, \quad a_2 = 3$

В цьому прикладі 375 – це число, записане в системі числення з основою p . Індекс 10 знизу праворуч від числа означає що p – основа системи числення, в якій записане число, дорівнює 10, тобто число представлене в десятковій системі числення; n =3, тобто число що розглядається має 3 розряди: нульовий розряд дорівнює 5, перший – 7 та другий – 3.

Формула (1.2) розкриває зміст формули (1.1):

$$x_p = \sum_{k=0}^{n-1} a_k p^k\,. \tag{1.2}$$

Розглянемо декілька прикладів щоб зрозуміти зміст формули (1.2).

Приклад 1.2.

$$375_{10} = 3\cdot 10^2 + 7\cdot 10^1 + 5\cdot 10^0.$$

375_{10} означає, що 5 береться з ваговим множником 10 (це основа системи числення) в нульовому степені, 7 – з ваговим множником 10 в першому степені, а 3 – із множником 10 в другому степені.

Приклад 1.3.

$$B9_{16} = 11 \cdot 16^1 + 9 \cdot 16^0$$

Наведене у цьому прикладі число $B9_{16}$ представлене в шістнадцятковій системі числення: 9 береться з ваговим множником 16 в нульовому степені. На відміну від попереднього прикладу основою є 16, а не 10, оскільки число подане в шістнадцятковій системі числення. Далі додаємо B, тобто 11, з ваговим множником 16 в першому степені.

Приклад 1.4.

$$1011101_2 = 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0.$$

В розглянутому двійковому числі одиниці розташовані на позиціях 0, 2, 3, 4 та 6, тому ці розряди множаться на 2^0, 2^2, 2^3, 2^4, 2^6 відповідно. На інших позиціях розташовані нулі, тому ми їх не враховуємо.

1.1.2. Основні алгоритми переходу до інших систем числення для цілих чисел

На формулі (1.2) базується ***алгоритм переходу від чисел у довільній системі до десяткової системи числення***:

1. Пронумерувати розряди числа, записаного в довільній системі числення.
2. Записати суму, скориставшись формулою (1.2).
3. Виконати обчислення та записати отриманий результат (вказавши основу системи числення 10).

Розглянемо декілька прикладів:

Приклад 1.5.

$1265_8 = ?_{10}$ $\underset{3\ 2\ 1\ 0}{1265_8} = 1\cdot 8^3 + 2\cdot 8^2 + 6\cdot 8^1 + 5\cdot 8^0 = 1\cdot 512 + 2\cdot 64 + 6\cdot 8 + 5 = 693_{10}$

По-перше, пронумеруємо розряди числа починаючи з крайнього правого та будемо вважати, що цей розряд має номер нуль, наступний розряд буде мати номер один і так далі. Тобто в нашому прикладі

Розряд із значенням 5 буде мати номер 0,

Розряд із значенням 6 буде мати номер 1,

Розряд із значенням 2 буде мати номер 2,

Розряд із значенням 1 буде мати номер 3,

Тепер скористаємося формулою (1.2) – будемо додавати розряди із ваговими коефіцієнтами:

5 із коефіцієнтом 8 в нульовому степені,

6 із коефіцієнтом 8 в першому степені,

2 із коефіцієнтом 8 в другому степені,

1 із коефіцієнтом 8 в третьому степені,

Отриманий після додавання результат і буде значенням числа в десятковій системі числення. В нашому прикладі це 693.

Приклад 1.6.

$121_3 = ?_{10}$ $\underset{2\ 1\ 0}{121_3} = 1\cdot 3^2 + 2\cdot 3^1 + 1\cdot 3^0 = 1\cdot 9 + 2\cdot 3 + 1 = 16_{10}$

В цьому прикладі крайню праву одиницю беремо із ваговим коефіцієнтом 3 в нульовому степені, двійку – із коефіцієнтом 3 в першому степені та крайню ліву одиницю беремо із ваговим коефіцієнтом 3 в другому степені – загалом отримаємо 16 в десятковому форматі.

Алгоритм переходу числа із десяткової системи числення у систему числення із довільною основою можна сформулювати наступним чином:

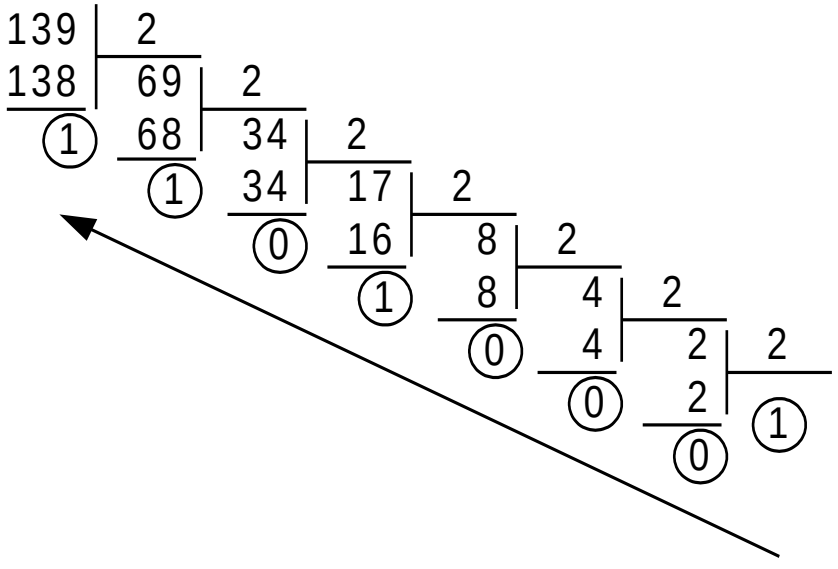
Беремо число, яке потрібно перевести в іншу систему числення, та ділимо це число на основу НОВОЇ системи числення. Операція ділення виконується за правилами «старої», десяткової системи числення. Після виконання цієї операції ми отримаємо частку та залишок. Залишок запам’ятовуємо, а частку знову ділимо на основу нової системи числення. Цю операцію потрібно повторювати до ти пір, доки частка не стане меншою нової основи системи числення.

Отримані остання частка та усі залишки складають розряди числа в новій системі числення. Причому частка буде найстаршим розрядом нового числа, а залишки записуються у зворотному порядку, тобто перший залишок буде останнім розрядом, другий – передостаннім і так далі.

Приклад 1.7.

Виконати наступне перетворення: $139_{10} = ?_2$, тобто знайти представлення десяткового числа 139 в двійковому форматі.

В цьому прикладі перетворення виконується у двійкову систему числення, тому 139 потрібно ділити на 2. Операція ділення виконується за правилами десяткової системи числення:



Результатом ділення 139 на 2 буде 69 із залишком 1 – це молодший (крайній правий розряд представлення числа в двійковому форматі). Результат

ділення 69 > 2, тому операцію ділення потрібно повторити. Новий результат ділення дорівнює 34 із залишком 1 – цей розряд записується лівіше від попереднього розряду. 34 > 2 , тому знову повторюємо ділення і так далі.

	Результат	Залишок	Результат менше 2	Номер розряду в двійковому представлення
139 : 2	69	1	Так	0
69 : 2	34	1	Так	1
34 : 2	17	0	Так	2
17 : 2	8	1	Так	3
8 : 2	4	0	Так	4
4 : 2	2	0	Так	5
2 : 2	1	0	Ні (1)	6 (7)

Отже, результат в цьому прикладі виглядатиме наступним чином:

$$139_{10} = 10001011_2$$

Хоча у комп'ютерних системах використовується двійкова система числення, для скорочення запису адрес та вмісту оперативної пам'яті комп'ютера зазвичай використовують шістнадцяткове та вісімкове відображення, оскільки перехід між цими системами числення простий та інтуїтивно зрозумілий.

Існують ***швидкі алгоритми переходу від двійкової системи числення до систем числення, основою яких є степені двійки***.

Оскільки $4 = 2^2$, то кожні два двійкових розряди числа утворюють один розряд в системі числення з основою 4. Аналогічно, оскільки $8=2^3$ та $16 = 2^4$, то

кожні три двійкових розряди відображення числа утворюють один вісімковий розряд, а кожних чотири двійкових розряди - один шістнадцятковий розряд.

Швидкий алгоритм переходу із двійкової системи числення у систему числення із основою 4 має наступний вигляд:

1. Розбити двійкове число в групи по 2 розряди починаючи від крайнього правого розряду. Якщо загальна кількість розрядів двійкового представлення числа непарна, то доповнити це число нулем ліворуч до парної кількості розрядів.
2. Перетворити кожну групу двійкових розрядів в одну цифру в системі числення із основою 4 відповідно до таблиці 1.2.

Таблиця 1.2. Таблиця відповідності чисел системи числення із основою 4 із їх двійковими представленнями.

Двійкове представлення цієї цифри	00	01	10	11
Цифра в системі числення із основою 4	0	1	2	3

Приклад 1.8.

Виконати наступне перетворення: $1110101_2=?_4$.

Спочатку потрібно розбити усі розряди двійкового представлення числа в групи по 2 розряди починаючи із крайнього правого розряду:

$$1110101_2= 1\quad 11\quad 01\quad 01_2$$

Оскільки загальна кількість розрядів дорівнює 7 (непарна кількість), то потрібно додати ще один розряд ліворуч:

$$1110101_2= 1\quad 11\quad 01\quad 01_2= \mathbf{0}1\quad 11\quad 01\quad 01_2$$

Після цього кожну групу із двох розрядів потрібно замінити єдиним розрядом в системі числення із основою 4 відповідно до значень в таблиці 1.2:

$$1110101_2 = 01\ \ 11\ \ 01\ \ 01_2 = 1\ 3\ 1\ 1_4$$

<u>Швидкий алгоритм переходу із двійкової системи числення у вісімкову систему числення</u> має наступний вигляд:

1. Розбити двійкове число в групи по 3 розряди починаючи від крайнього правого розряду. Якщо загальна кількість розрядів двійкового представлення числа не кратна 3, то доповнити це число необхідною кількістю нулів ліворуч так, щоб кількість розрядів стала кратною 3.
2. Перетворити кожну групу двійкових розрядів в одну цифру в системі числення із основою 8 відповідно до таблиці 1.3.

Таблиця 1.3 Таблиця відповідності чисел системи числення із основою 8 із їх двійковими представленнями.

Двійкове представлення цієї цифри	000	001	010	011	100	101	110	111
Цифра в системі числення із основою 8	0	1	2	3	4	5	6	7

Приклад 1.9.

Виконати наступне перетворення: $1110101_2 = ?_8$.

Оскільки $8 = 2^3$, то розбивати двійкові розряди потрібно вже в групи по 3 розряди починаючи із крайнього правого розряду:

$$1110101_2 = 1\ \ 110\ \ \ 101_2$$

Оскільки загальна кількість розрядів дорівнює 7 (не кратне 3), нам потрібно додати ще два розряди ліворуч:

1110101_2= 1 110 101_2= **00**1 110 101_2

Після цього кожну групу із трьох розрядів потрібно замінити єдиним розрядом в системі числення із основою 8 відповідно до значень в таблиці 1.3:

1110101_2= 1 110 101_2= **00**1 110 101_2= 1 6 5_8

Швидкий алгоритм переходу із двійкової системи числення у систему числення із основою 16 має наступний вигляд:

1. Розбити двійкове число в групи по 4 розряди починаючи від крайнього правого розряду. Якщо загальна кількість розрядів двійкового представлення числа не кратна 4, то доповнити це число необхідною кількістю нулів ліворуч так, щоб кількість розрядів стала кратною 4.
2. Перетворити кожну групу двійкових розрядів в одну цифру в системі числення із основою 16 відповідно до таблиці 1.4.

Таблиця 1.4 Таблиця відповідності чисел системи числення із основою 16 із їх двійковими представленнями.

Двійкове представлення цієї цифри	0000	0001	0010	0011	0100	0101	0110	0111
Цифра в системі числення із основою 16	0	1	2	3	4	5	6	7

Двійкове представлення цієї цифри	1000	1001	1010	1011	1100	1101	1110	1111
Цифра в системі числення із основою 16	8	9	10	11	12	13	14	15

Приклад 1.10.

Виконати наступне перетворення: $1110101_2=?_{16}$.

Число 16 = 2^4 тому в цьому прикладі будемо виконувати групування двійкових розрядів по 4 розряди починаючи із крайнього правого розряду:

$$1110101_2= 111\ \ 0101_2$$

Наявні 7 розрядів, як і в попередніх прикладах, не кратні 16, тому нам потрібно додати ще один розряди ліворуч:

$$1110101_2= 111\ \ 0101_2= 0111\ \ 0101_2$$

Далі кожну групу із 4 розрядів заміняємо єдиним розрядом в системі числення із основою 16 відповідно до значень в таблиці 1.4:

$$1110101_2= 0111\ \ 0101_2= 75_{16}$$

З точністю до навпаки будуються ***швидкі алгоритми переходу від систем числення, основою яких є степені двійки до двійкової системи числення***.

Швидкий алгоритм переходу із системи числення із основою 4 в двійкову систему числення:

1. Представити кожен розряд числа, представленого в системі числення із основою 4, у вигляді 2 двійкових розрядів відповідно до таблиці 1.2.
2. Якщо отримане число в двійковому форматі починається із нуля, то потрібно прибрати цей не значущий лівий розряд.

Приклад 1.11.

Виконати наступне перетворення: $132_4 =?_2$.

Відповідно до таблиці 1.2:

$$1_4 \to 01_2$$

$$3_4 \to 11_2$$

$$2_4 \to 10_2$$

Отже, $132_4 = \underline{\mathbf{0}}1\ \ 11\ \ 10_2$

Ліворуч маємо незначущий нуль, його можна не писати:

$$132_4 = \underline{\mathbf{0}}11110_2 = 11110_2$$

<u>Швидкий алгоритм переходу із системи числення із основою 8 в двійкову систему числення</u>:

1. Представити кожен розряд числа, представленого в системі числення із основою 8, у вигляді 3 двійкових розрядів відповідно до таблиці 1.3.
2. Якщо отримане число в двійковому форматі починається із нулів, то потрібно прибрати ці незначущі розряди ліворуч.

Приклад 1.12

Виконати наступне перетворення: $132_4 = ?_2$.

Відповідно до таблиці 1.3:

$$1_4 \to 001_2$$

$$3_4 \to 011_2$$

$$2_4 \to 010_2$$

Отже, $132_4 = \underline{\mathbf{00}}1\ \ 011\ \ 010_2$

Два найстарші розряди є нулями, і вони не впливають на значення, тому прибираємо їх:

$$132_4 = \underline{\mathbf{00}}1011010_2 = 1011010_2$$

<u>Швидкий алгоритм переходу із системи числення із основою 16 в двійкову систему числення</u>:

1. Представити кожен розряд числа, поданого в системі числення із основою 16, у вигляді 4 двійкових розрядів відповідно до таблиці 1.4.

2. Якщо отримане число в двійковому форматі починається із нулів, то потрібно прибрати ці незначущі розряди ліворуч.

Приклад 1.13.

Виконати наступне перетворення: $3F1_{16} = ?_2$.

Скористаємося таблицею 1.4:

$$3_4 \to 0011_2$$

$$F_4 \to 1111_2$$

$$1_4 \to 0001_2$$

Отже, $3F1_{16} = \underline{\mathbf{00}}11\ \ 1111\ \ 0001_2$

Два найстарші розряди є нулями, і вони не впливають на значення, тому прибираємо їх:

$$3F1_{16} = \underline{\mathbf{00}}1111110001_2 = 1111110001_2$$

Якщо потрібно перейти із системи числення яка має основу 2^{α} в систему числення із основою 2^{β}, можна скористатися двійковою системою числення як проміжною та скористатися раніш описаними швидкими алгоритмами переходу.

Приклад 1.14.

Виконати наступне перетворення: $3F1_{16} = ?_8$.

В цьому прикладі потрібно виконати перехід від системи числення із основою $16 = 2^4$ ($\alpha = 4$) до системи числення із основою $8 = 2^3$ ($\beta = 3$). Як раніше згадувалось, простіш за все це зробити по наступній схемі:

$$3F1_{16} = ?_2 = ?_8$$

1) Перехід від шістнадцяткового представлення до двійкового:

$$3_4 \to 0011_2$$

$$F_4 \to 1111_2$$

$$1_4 \to 0001_2$$

$$3F1_{16} = \mathbf{00}1111110001_2 = 1111110001_2$$

2) Перехід від двійкового представлення до вісімкового:

$$1111110001_2 = 1\ \ 111\ \ 110\ \ 001_2 = 1761_8$$

Отже відповідь має вигляд: $3F1_{16} = 1761_8$

В загальному випадку, ***для перетворення числа A із основою системи числення p в число із основою системи числення q*** можна скористатися простим алгоритмом:

> Число А потрібно поділити на НОВУ основу системи числення q. Отриманий залишок буде молодшим розрядом представлення числа в новій системі числення. Якщо частка більша основи нової системи числення q, то ми виконуємо ділення ще раз. І так до тих пір, доки ми не отримаємо залишок менше q.

Одне важливе зауваження – **усі операції виконуються в системі числення з основою *q***. Але це може бути досить важко без певних навичок. Тому таке перетворення зручніше виконати у два етапи **через проміжну десяткову систему числення**.

Приклад 1.15.

Виконати перетворення: $121_3 = ?_8$.

Оскільки виконувати розрахунок в системі числення із основою 3 не зручно, виконаємо перехід із додатковим кроком:

$$121_3 = ?_{10} = ?_8$$

1. На першому кроці скористаємося формулою (1.2) для переходу до десяткової системи числення:

$$121_3 = 1\cdot3^2 + 2\cdot3^1 + 1\cdot3^0 = 9 + 6 + 1 = 16_{10}.$$

2. Виконаємо послідовне ділення отриманого десяткового числа на нову основу системи числення ($p = 8$):

$$\begin{array}{r|l} 16 & 8 \\ \hline 16 & 2 \\ \hline 0 & \end{array} \longrightarrow 16_{10} = 20_8 \longrightarrow 121_3 = 20_8$$

Відповідь: $121_3 = 20_8$.

Приклад 1.16.

Виконати наступне перетворення: $100111_2 = ?_3$.

Як ми вже згадували, зручніш за все виконати таке перетворення через десяткову систему числення. Тобто виконати перехід від двійкової системи числення до десяткової системи, а потім від десяткової системи числення до системи з основою 3:

$$100111_2 = ?_{10} = ?_3$$

1. На першому кроці скористаємося формулою (1.2) для переходу до десяткової системи числення:

   По-перше, пронумеруємо розряди числа починаючи з крайнього правого та будемо вважати, що цей розряд має номер нуль, наступний розряд буде мати номер один і так далі. Усього в нашому числі 6 розрядів із номерами від нуля (молодший, самий правий) до п'яти (старший, самий лівий):

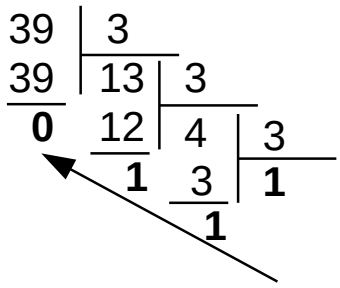
   $$\underset{5\ 4\ 3\ 2\ 1\ 0}{100111}$$

   Тепер скористаємося формулою (1.2) – будемо додавати розряди із ваговими коефіцієнтами: молодші три розряди дорівнюють одиниці із коефіцієнтами 2 в нульовому степені, 2 в першому степені та 2 в другому степені, Наступні два розряди дорівнюють нулям і остання (ліва) одиниця із коефіцієнтом 2 в п'ятому степені:

   $$100111_2 = 1\cdot 2^0 + 1\cdot 2^1 + 1\cdot 2^2 + 0\cdot 2^3 + 0\cdot 2^4 + 1\cdot 2^5 = 1 + 2 + 4 + 32 = 39_{10}.$$

2. Другий крок у разі виконання цього прикладу – перехід від десяткової системи числення до систему числення із основою 3. Оскільки ми переходимо від десяткової системи числення, то всі подальші операції будемо виконувати в десятковій системі числення.

Ми переходимо в систему числення із основою 3, тому будемо ділити число 39 на 3 до тих пір, поки залишок від ділення не стане менше трьох:



Записуємо розряди числа в системі числення з основою 3. Старшим розрядом напишемо частку від останнього ділення, дали залишки у зворотному порядку їх отримання. Отримуємо:

$$39_{10} = 1110_3$$

Перевіримо виконання цього завдання. Для цього скористаємося формулою (1.2).

$$1110_3 = 0 \cdot 3^0 + 1 \cdot 3^1 + 1 \cdot 3^2 + 1 \cdot 3^3 = 3 + 9 + 27 = 39_{10}.$$

Відповідь: $100111_2 = 1110_3$.

1.1.3. Алгоритм переходу від десяткової системи числення до двійкової для чисел, які мають дробову частину

В загальному випадку число, записане в позиційній системі числення, і яке має дробову чистину, можна записати як показано формулі (1.3) і це є узагальненням формули (1.1):

$$x_p = a_{n-1}a_{n-2}\ldots a_0, a_{-1}a_{-2}\ldots a_{-m}. \quad (1.3)$$

Кома, або точка переважно у англомовних країнах, відділяє розряди цілої та дробової частин числа. Відповідно, зміст формули (1.3) розкривається формулою (1.4), яка є узагальненням формули (1.2):

$$x_p = \sum_{k=-m}^{n-1} a_k p^k . \quad (1.4)$$

У такій системі запису розділювач завжди знаходиться між розрядами a_0 та a_{-1}, тому її також називають системою з фіксованою комою (точкою), а власне сам розділювач інколи називають десятковою комою (точкою).

Алгоритм переходу із десяткової системи числення до двійкової для чисел, які мають дробову частину, складається із таких кроків:

1. Спочатку реалізується перехід із десяткової системи числення до двійкової тільки цілої частини числа.
2. Далі тільки дробова частина десяткового числа множиться на нову основу системи числення (тобто, для двійкової – на 2).
3. У результат множення виділяється ціла частина та її значення приймається як $i-$тий розряд після коми $\left(a_{-i}\right)$ в новій системі числення $(i = 1, 2, \ldots)$.
4. Якщо дробова частина результату не дорівнює нулю, то обчислення повторюється починаючи із кроку 2 до тих пір, доки ми не отримаємо

нуль в дробовій частині результату множення чи не буде отримано потрібну кількість розрядів.

Приклад 1.17.

Виконати перетворення десяткового дробу до двійкової системи числення: $7{,}625_{10} = ?_2$.

Відповідно до раніш описаного алгоритму, спочатку переведемо в двійкову систему числення цілу частину числа:

$$7_{10} = 111_2.$$

Далі реалізуємо перехід в двійкову систему числення для дробової частини числа. Домножуємо дробову частину на 2 і розряд цілої частини добутку запам'ятовуємо як перший розряд після коми:

$$0,625 \cdot 2 = 1,25.$$

Тобто першій двійковий розряд після коми в двійковому представленні дробової частини буде $a_{-1} = 1$. Дробова частина результату $(0,25)$ не дорівнює нулю, тому множення на 2 виконуємо ще раз:

$$0,25 \cdot 2 = 0,5.$$

Новий розряд дорівнює $a_{-2} = 0$ а дробова частина результату не дорівнює нулю, тому ще раз повторюємо множення:

$$0,5 \cdot 2 = 1,0.$$

Ще один розряд знайдено: $a_{-3}=1$. Причому, на цьому кроці дробова частина числа стала рівною 0, а це означає, що вже знайдено точне двійкове представлення дробової частини числа.

Відповідь: $7{,}625_{10} = 111{,}101_2$.

Не завжди вдається абсолютно точно представити дробову частину десяткового числа в двійковій системі числення. Насамперед це стосується ірраціональних чисел, проте існують і раціональні числа, які можна записати точно у одній системі числення, але не можна записати точно у іншій. Розглянемо приклад.

Приклад 1.18.

Виконати перетворення: $3{,}116_{10} = ?_2$, вважаючи, що для зберігання дробової частини числа відведено 10 двійкових розрядів.

Спочатку знайдемо цілу частину числа:

$$3_{10} = 11_2.$$

Для знаходження дробової частини числа будемо виконувати послідовне множення на 2 і запам’ятовувати розряд цілої частини добутку:

$$a_{-1}: \quad 0{,}116 \cdot 2 = 0{,}232;$$
$$a_{-2}: \quad 0{,}232 \cdot 2 = 0{,}464;$$
$$a_{-3}: \quad 0{,}464 \cdot 2 = 0{,}928;$$
$$a_{-4}: \quad 0{,}928 \cdot 2 = 1{,}856;$$
$$a_{-5}: \quad 0{,}856 \cdot 2 = 1{,}712;$$
$$a_{-6}: \quad 0{,}712 \cdot 2 = 1{,}424;$$

$$a_{-7}: \quad 0,424 \cdot 2 = 0,848;$$
$$a_{-8}: \quad 0,848 \cdot 2 = 1,696;$$
$$a_{-9}: \quad 0,696 \cdot 2 = 1,392;$$
$$a_{-10}: \quad 0,392 \cdot 2 = 0,784.$$

Множення було виконано 10 разів і було отримано 10 розрядів двійкового представлення дробової частини числа, тобто всі відведені для зберігання дробової частини розряди вже заповнені. Але навіть після останнього множення дробова частина результату все ще не дорівнює нулю. Позаяк вільних розрядів для запису дробової частини вже нема, то на цьому алгоритм перетворення буде зупинено, а неперетворений залишок – втрачено. Це означає, що двійкове представлення дробової частини в двійковому форматі буде виконано із округленням.

Відповідь: $3,116_{10} = 11,0001110110_2$ (із округленням представлення дробової частини до 10 двійкових розрядів).

Виконаємо зворотне перетворення із двійкової системи числення до десяткової. $11_2 = 3_{10}$. Для дробової частини також скористаємося методом додавання розрядів, пам'ятаючи, що степені праворуч від коми змінюються у порядку –1, –2,…:

$$0001110110_2 = 0 \cdot 2^{-1} + 0 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4} + 1 \cdot 2^{-5} +$$
$$1 \cdot 2^{-6} + 0 \cdot 2^{-7} + 1 \cdot 2^{-8} + 1 \cdot 2^{-9} + 0 \cdot 2^{-10} =$$
$$= \frac{1}{16} + \frac{1}{32} + \frac{1}{64} + \frac{1}{256} + \frac{1}{512} =$$
$$= 0,0625 + 0,03125 + 0,015625 + 0,00390625 + 0,001953125 =$$
$$= 0,115\left[234375\right].$$

Отже, у пам'яті машини буде збережено $3,115_{10}$ замість потрібного 3,116. Іншими словами, число збережено у пам'яті машини наближено, бо виникла похибка округлення при переході від однієї системи числення до іншої.

Зрозуміло, що під час зворотного переходу від двійкової системи до десяткової, наприклад, для відображення результату операції користувачеві, також можуть виникати подібні похибки округлення. Детальне вивчення похибок округлення та їх впливу на результати обчислень різними методами є предметом вивчення обчислювальної математики.

1.2. Завдання

Номер варіанта потрібно обирати за наступною формулою:

$$N = K + M ,$$

де N – номер варіанту,

$K = 0$ для груп ДМ-х1 та ДП-х1 та $K = 20$ для груп ДМ-х2 та ДП-х2,

M – номер із залікової книжки.

У разі оформлення роботи обов'язково наводити усі розрахунки та коментувати усі етапи роботи.

1. Виконайте наступні перетворення з однієї системи числення в іншу:

Номер варіанта	Завдання 1	Завдання 2
1	$10011010_2=?_3$	$12011010_7=?_5$
2	$10110100_2=?_3$	$10310100_7=?_5$
3	$10010110_2=?_3$	$10040110_7=?_5$
4	$11111001_2=?_3$	$11115001_7=?_5$
5	$10000100_2=?_3$	$10000600_7=?_5$
6	$11001000_2=?_3$	$11001020_7=?_5$
7	$10110101_2=?_3$	$10110103_7=?_5$
8	$11110111_2=?_3$	$41110111_7=?_5$
9	$10000010_2=?_3$	$15000010_7=?_5$
10	$10111101_2=?_3$	$10611101_7=?_5$
11	$10011000_2=?_3$	$10021000_7=?_5$
12	$11110001_2=?_3$	$11113001_7=?_5$
13	$10110110_2=?_3$	$10110410_7=?_5$
14	$10000011_2=?_3$	$10000051_7=?_5$
15	$11110110_2=?_3$	$11110116_7=?_5$

Номер варіанта	Завдання 1	Завдання 2
16	$10111110_2=?_3$	$20111110_7=?_5$
17	$11001101_2=?_3$	$13001101_7=?_5$
18	$10011001_2=?_3$	$10411001_7=?_5$
19	$10110111_2=?_3$	$10150111_7=?_5$
20	$11110010_2=?_3$	$11116010_7=?_5$
21	$11010010_2=?_3$	$11010210_7=?_5$
22	$10011011_2=?_3$	$10011031_7=?_5$
23	$10100000_2=?_3$	$10100004_7=?_5$
24	$11111001_2=?_3$	$51111001_7=?_5$
25	$10000101_2=?_3$	$16000101_7=?_5$
26	$11001110_2=?_3$	$11201110_7=?_5$
27	$10001001_2=?_3$	$10031001_7=?_5$
28	$11001001_2=?_3$	$11004001_7=?_5$
29	$10111000_2=?_3$	$10111500_7=?_5$
30	$10000110_2=?_3$	$10000160_7=?_5$
31	$11110100_2=?_3$	$11110102_7=?_5$
32	$10001000_2=?_3$	$30001000_7=?_5$
33	$10111100_2=?_3$	$14111100_7=?_5$
34	$10011100_2=?_3$	$10511100_7=?_5$
35	$11001111_2=?_3$	$11061111_7=?_5$
36	$10011101_2=?_3$	$10012101_7=?_5$
37	$11110101_2=?_3$	$11110301_7=?_5$
38	$10000111_2=?_3$	$10000141_7=?_5$
39	$11010000_2=?_3$	$11010005_7=?_5$
40	$10011111_2=?_3$	$60011111_7=?_5$
41	$10111111_2=?_3$	$14131201_7=?_5$

2. Перетворіть вісімкові числа на двійкові та шістнадцяткові:

Номер варіанта	Завдання 1
1	1023_8
2	2311_8
3	1771_8
4	6336_8
5	1056_8
6	4321_8
7	6665_8
8	3756_8
9	3055_8
10	6446_8
11	5630_8
12	4512_8
13	2271_8
14	7032_8
15	1551_8
16	5000_8
17	5366_8
18	4701_8
19	6233_8
20	5775_8
21	2541_8
22	4513_8
23	4444_8
24	2764_8
25	2552_8
26	3015_8

Номер варіанта	Завдання 1
27	1663_8
28	4565_8
29	4721_8
30	5225_8
31	4500_8
32	2744_8
33	3150_8
34	7113_8
35	5643_8
36	6666_8
37	4517_8
38	7702_8
39	4664_8
40	7745_8
41	1773_8

3. Перетворіть десяткове число із дробовою частиною у двійкове, якщо під зберігання дробової частини відведено 12 двійкових розрядів:

Номер варіанта	Завдання 1	Завдання 2
1	311,9410	$355,6616_{10} = ?_2$
2	4306,7853	$8799,5142_{10} = ?_2$
3	8278,9963	$2280,3060_{10} = ?_2$
4	922,5687	$4274,4971_{10} = ?_2$
5	5376,5312	$4859,7316_{10} = ?_2$
6	4065,1919	$6340,9618_{10} = ?_2$

Номер варіанта	Завдання 1	Завдання 2
7	8511,8415	$7425,6774_{10} = ?_2$
8	$1380,5641_{10} = ?_2$	$9953,4383_{10} = ?_2$
9	$4292,7517_{10} = ?_2$	$8358,0950_{10} = ?_2$
10	$5186,6181_{10} = ?_2$	$1739,2317_{10} = ?_2$
11	$8767,2100_{10} = ?_2$	$8821,5014_{10} = ?_2$
12	$7650,4262_{10} = ?_2$	$4255,6159_{10} = ?_2$
13	$6913,1949_{10} = ?_2$	$604,9082_{10} = ?_2$
14	$2742,6644_{10} = ?_2$	$5688,6753_{10} = ?_2$
15	$1759,2166_{10} = ?_2$	$5340,7666_{10} = ?_2$
16	$8158,9670_{10} = ?_2$	$6935,3118_{10} = ?_2$
17	$8713,0068_{10} = ?_2$	$5442,4513_{10} = ?_2$
18	$7389,8739_{10} = ?_2$	$1075,6908_{10} = ?_2$
19	$6315,2295_{10} = ?_2$	$6143,4753_{10} = ?_2$
20	$6563,0716_{10} = ?_2$	$8474,3753_{10} = ?_2$
21	$9613,1796_{10} = ?_2$	$306,1885_{10} = ?_2$
22	$7801,7620_{10} = ?_2$	$1633,3688_{10} = ?_2$
23	$5679,1642_{10} = ?_2$	$1804,5352_{10} = ?_2$
24	$3181,2787_{10} = ?_2$	$7638,2822_{10} = ?_2$
25	$5900,5402_{10} = ?_2$	$8211,9000_{10} = ?_2$
26	$3934,9858_{10} = ?_2$	$6320,5932_{10} = ?_2$
27	$5269,4809_{10} = ?_2$	$6751,2304_{10} = ?_2$
28	$680,8668_{10} = ?_2$	$5616,7708_{10} = ?_2$
29	$6705,9021_{10} = ?_2$	$9629,3621_{10} = ?_2$
30	$2591,9407_{10} = ?_2$	$1978,2407_{10} = ?_2$
31	$1439,0212_{10} = ?_2$	$1689,1766_{10} = ?_2$
32	$9496,1654_{10} = ?_2$	$9555,2330_{10} = ?_2$
33	$7103,5011_{10} = ?_2$	$6466,4638_{10} = ?_2$

Номер варіанта	Завдання 1	Завдання 2
34	$5198{,}3918_{10} = ?_2$	$5467{,}0224_{10} = ?_2$
35	$8778{,}3393_{10} = ?_2$	$3537{,}7771_{10} = ?_2$
36	$2269{,}3519_{10} = ?_2$	$2564{,}6000_{10} = ?_2$
37	$336{,}1306_{10} = ?_2$	$7166{,}6505_{10} = ?_2$
38	$949{,}7566_{10} = ?_2$	$3308{,}1037_{10} = ?_2$
39	$2340{,}4567_{10} = ?_2$	$4389{,}6905_{10} = ?_2$
40	$8332{,}0630_{10} = ?_2$	$489{,}4768_{10} = ?_2$
41	$8550{,}7455_{10} = ?_2$	$5863{,}0232_{10} = ?_2$

1.3. Контрольні питання

1. Яка система числення – позиційна чи непозиційна є більш зручною для виконання операцій множення та ділення?
2. Які символи використовуються в системах числення, основа яких більша ніж 10, окрім звичайних цифр?
3. Яка система числення є найпоширенішою в сучасних цифрових системах?
4. Якщо потрібно перейти від системи числення з основою 5 до системи числення із основою 9, то це зручно зробити із використанням додаткового етапу чи безпосередньо? Якщо потрібно додатковий етап, то який?
5. В чому основний недолік двійкової системи числення у разі використання її в цифрових системах? Як із цим можуть допомогти вісімкова та шістнадцяткова системи числення?

2. Зберігання цілих чисел в цифрових системах

Мета роботи: навчитися записувати цілі числа в прямому, оберненому та додатковому кодах.

2.1. Теоретичні відомості

Як раніше вже згадувалося, сучасна цифрова техніка використовує двійкове представлення чисел для їх зберігання, отже і арифметичні операції над числами також виконуються у двійковому вигляді.

Одиницею вимірювання інформації в цифровій техніці є біт (BInary digiT – bit) – саме така кількість інформації міститься у відповіді на питання: нуль або один?

У комп'ютерній техніці часто використовується величина, яка називається байтом (byte). Назва «байт» byte є скороченням словосполучення BinarY TErm - «двійковий терм». Цей термін було вперше використано в 1956 році Бухгольцом у процесі проектування першого суперкомп'ютера IBM 7030 для шести бітів. Пізніше, в рамках того ж проекту, байт було розширено до восьми біт. Зараз під байтом розуміють групу з 8 біт. Байт дозволяє закодувати одне із 256 можливих значень.

Машинним словом називають найбільшу кількість бітів, яку обчислювальна мишина може обробляти як єдине ціле. Довжина машинного слова залежить від розрядності процесора і може бути рівною 16, 32, 64 чи більше бітам.

Поряд з байтами для вимірювання кількості інформації використовуються кратні одиниці:

1 Кбайт (один кілобайт) = 2^{10} байт (два в десятому степені) = 1024 байта;

1 Мбайт (один мегабайт) = 2^{10} Кбайт (два в десятому степені)= 1024 Кбайта;

1 Гбайт (один гігабайт) = 2^{10} Мбайт (два в десятому степені)= 1024 Мбайт, і так далі.

Цікаво, що у метричній системі префікси кіло, мега, гіга означають множники 10^3, 10^6 та 10^9 відповідно. У теперішній час метричні префікси часто використовуються для вимірювання швидкості обміну та ємності носіїв інформації, а раніші префікси у вигляді «дцятих степенів двійки» називаються двійковими і позначаються як кібібайти, мебібайти, гібібайти.

На рис. 2.1 наведено одну з можливих класифікацій чисел, які використовуються у обчислювальній техніці.

Рис. 2.1. Типи чисел з огляду на формати їх запису в двійковому представленні.

Цілі числа можуть мати знак, або бути беззнакові. В свою чергу, знакові числа можуть бути додатними та від’ємними.

Хоча в математичних задачах не так часто зустрічаються величини, що не мають негативних значень, беззнакові типи даних досить поширені в комп'ютерній техніці. Головна причина полягає в тому, що в самій машині та програмах для неї є багато такого роду об'єктів: перш за все, адреси комірок пам'яті, а також різноманітні лічильники. До цього списку варто додати числа,

що позначають дату та час, розміри графічних зображень в пікселях і т.п. Все перераховане вище завжди і у всіх програмах має тільки ціле значення.

2.1.1. Представлення цілих беззнакових чисел в цифрових системах

Беззнакові цілі числа представляються в машині найпростіше. Для цього достатньо перевести число в двійкову форму та доповнити отриманий результат зліва нулями до стандартної розрядності.

Приклад 2.1.

Записати як буде зберігатися в цифровій системі число 21_{10}, якщо для його зберігання відведено $k = 8$ бітів (розрядів).

Представимо число 21_{10} в двійковому вигляді:

$$21_{10} = 10101_2.$$

В умові задачі задано, що для зберігання цього числа в цифровій системі буде відведено $k = 8$ бітів, але після переходу в двійкову систему числення ми маємо тільки 5 розрядів, тому це число доповнюється 3 двійковими розрядами зліва:

$$21_{10} = 00010101_2.$$

Для зручності візуального сприйняття для запису двійкові числа часто розбиваються на групи при 4 розряди і записуються наступним чином:

$$21_{10} = 0001\ \ 0101_2.$$

Як згадувалося раніше, двійкова форма запису не є зручною через велику кількість розрядів, тому зручніше (компактніше) записувати числа в шістнадцятковому форматі. В попередній темі було наведено швидкий алгоритм переходу від двійкового представлення числа до шістнадцяткового, коли усі двійкові розряди числа, починаючи із крайньої правою позиції, розбиваються на групи по 4 розряди, після чого кожна така група заміняється одним шістнадцятковим розрядом відповідно до таблиці 1.4:

$$21_{10} = 0001\ \ 0101_2 = 15_{16}$$

В технічній документації (специфікації, стандарти, технічні завдання і тому подібне) використовується саме шістнадцятковий запис чисел, маючи на увазі використання двійкових чисел в цифрових системах.

Відповідь: $21_{10} = 0001\ \ 0101_2 = 15_{16}$.

Приклад 2.2.

Записати як буде зберігатися в цифровій системі число 21_{10}, якщо для його зберігання відведено два байти.

Один байт містить 8 бітів (двійкових розрядів), тому 2 байти, як задано в умові задачі, $k = 16$ бітів (розрядів). Оскільки в двійковому представленні число 21_{10} містить тільки 5 бітів ($21_{10} = 10101_2$), то до необхідної розрядності $k = 16$ потрібно додати 16 - 5 = 11 нулів ліворуч:

$$21_{10} = 0000\ \ 0000\ \ 0001\ \ 0101_2 = 0015_{16}$$

Відповідь: $21_{10} = 0000\ \ 0000\ \ 0001\ \ 0101_2 = 0015_{16}$.

Для беззнакових чисел досить легко визначити діапазон значень, які приймає число, яке складається із k -бітів.

Мінімальне значення –це коли в усіх розрядах міститься мінімальне цифра, притаманна двійковій системі числення -нулі, тобто для будь-якого k це значення завжди дорівнює нулю:

$$Min_k = 0.$$

Максимальне значення – це коли в усіх розрядах міститься максимальна цифра, яка використовується в двійковій системі числення- одиниця. Тобто, в загальному випадку, максимальне можливе значення, яке можна записати за допомогою k двійкових розрядів, буде дорівнювати:

$$Max_k = \sum_{i=0}^{k-1} 2^i = 2^k - 1.$$

Приклад 2.3.

Визначити діапазон чисел, які можна зберігати в цифровій системі, яка має $k = 3$ двійкові розряди в кожному машинному слові.

$$Min_{k=3} = 0;$$
$$Max_{k=3} = 111_2 = 7_{10};$$
$$Max_{k=3} = 2^3 - 1 = 8 - 1 = 7_{10}.$$

Відповідь: комірка із $k = 3$ дозволяє зберігати числа [0…7].

Приклад 2.4.

Визначити діапазон чисел, які можна зберігати в цифровій системі в якій довжина машинного слова дорівнює 1 байту.

Один байт означає, що $k = 8$, тобто:

$$Min_{k=8} = 0;$$

$$Max_{k=8} = 1111\ 1111_2 = 255_{10};$$

$$Max_{k=8} = 2^8 - 1 = 256 - 1 = 255_{10}.$$

Відповідь: 1 байт дозволяє зберігати беззнакові цілі числа у діапазоні [0…255].

Оскільки, зазвичай, в обчислювальній техніці використовують одно-, дво-, чотири- та восьмибайтові формати запису чисел, в табл. 2.1 показані максимальні значення чисел, які можна записати за використання відповідної довжини машинного слова.

Таблиця 2.1. Максимальне ціле значення, яке може бути збережено 1-, 2-, 4- та 8- байтовому машинному слові (представлення в різних системах числення).

k	1 байт ($k=8$)	2 байти ($k=16$)	4 байт ($k=32$)	8 байт ($k=64$)
Максимальне значення (двійковий формат)	1111 1111	1111 1111 1111 1111	-	-
Максимальне значення (шіснацятковий формат)	FF	FF FF	FF FF FF FF	FF FF FF FF FF FF FF FF

Максимальне значення (десяткове представлення)	255	65 535	4 294 967 295	18 446 744 073 709 551 615

Як ми бачимо, однобайтне представлення дозволяє записати числа від 0 до 255. Якщо розглядати 2 байти як єдину величину, то можна записати числа від 0 до 65 535. Чотири- та восьмибайтні формати чисел дають ще більший діапазон можливих значень.

2.1.2. Представлення цілих знакових чисел в цифрових системах

Для запису знакових цілих чисел використовують дещо інший підхід. Окрім власне модуля числа слід також передбачити можливість зберігати також і знак числа, щоб відрізняти додатні числа від від'ємних. Найпоширенішими є такі формати зберігання (див рис. 2.2):

- прямий код;
- обернений;
- додатковий код (інколи в літературі використовують термін «Доповняльний код»).

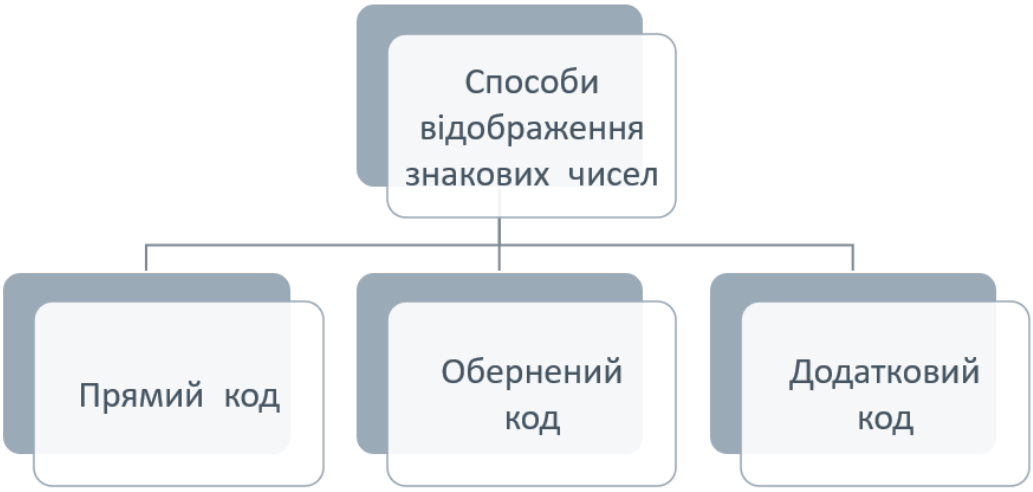


Рис. 2.2. Найпоширеніші формати зберігання знакових цілих чисел.

Формат зберігання знакового цілого числа в цифровій системі наведено на рис. 2.3. Всі три способи використовують самий лівий (старший) розряд бітового набору довжини k для кодування знаку числа, причому:

- знак "плюс" кодується 0,
- знак "мінус" кодується одиницею.

Решта $k-1$ розряд, які називаються мантиса або цифрова частина, використовуються для запису абсолютної величини числа.

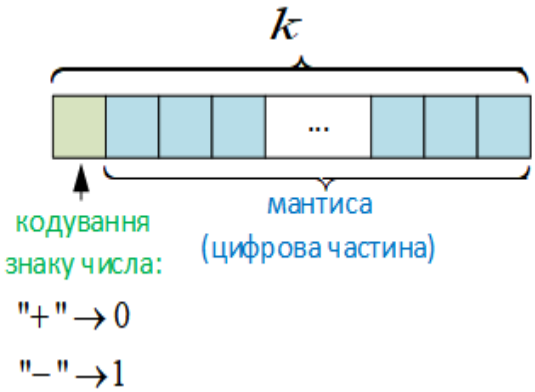


Рис. 2.3. Спосіб зберігання знакового цілого числа в цифровій системі.

Обернений та додатковий коди застосовуються особливо широко, позаяк дозволяють спростити конструкцію арифметико-логічного пристрою шляхом заміни арифметичної операції віднімання операцією додавання.

Зазвичай десяткові числа у процесі введення в машину перетворюються в двійковий код, якщо це ціле число без знаку, чи обернений або додатковий код, якщо це ціле число зі знаком і вже в такому вигляді зберігаються, переміщуються і беруть участь в операціях. При виведенні результатів з машини відбувається зворотне перетворення із двійкового, оберненого чи додаткового представлення в десяткове число.

2.1.3. Представлення цілих знакових чисел в цифрових системах: прямий, обернений та додатковий коди додатних чисел

Додатні числа (і нуль) в прямому, оберненому та додатковому кодах зображуються однаково – цифрова частина містить двійковий запис числа, а в знаковому розряді міститься 0. Якщо число, яке записується в мантису, містить меншин ніж $k-1$ двійковий розряд, то наявні розряди записуються в мантису починаючи із наймолодшого (крайнього лівого), а незаповнені розряди мантиси заповнюються нулями.

Мінімальне значення модуля числа за такої форми запису буде 0:

$$Min_k = 0.$$

Максимальне значення можна отримати у разі одиниць в усіх $k-1$ цифрових розрядах:

$$Max_k = 2^{k-1} - 1.$$

Приклад 2.5.

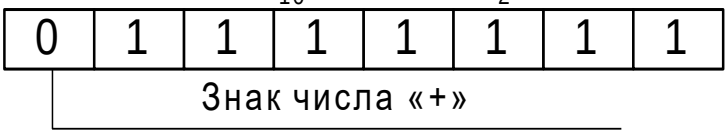
Визначити діапазон додатних значень, які можна записати за допомогою 1 байту, якщо числа зберігаються як так, що можуть мати знак.

$$Min = 0;$$

$$Max = 111\ 1111_2 = 127_{10};$$

$$Max = 2^{8-1} - 1 = 2^7 - 1 = 128 - 1 = 127_{10}.$$

$127_{10} = 1111111_2$



Відповідь: комірка із $k = 8$ (додатне число, знаковий формат) дозволяє зберігати числа [0…127].

Приклад 2.6

Записати число 45_{10} в комірку цифрової системи довжиною 1 байт, якщо вважається, що формат зберігання числа відповідає знаковому числу.

Число 45_{10} є додатнім, тому в найстарший (крайній лівий) розряд записується нуль. Переведемо число 45_{10} в двійкову систему числення:

$$45_{10} = 10\ \ 1101_2 .$$

В двійковому форматі число що розглядається містить тільки 6 розрядів, в той час як для зберігання мантиси відведено $k - 1 = 8 - 1 = 7$ розрядів, тому доповнюємо число одним нулем зліва:

$$45_{10} = 10\ \ 1101_2\ = 010\ \ 1101_2.$$

Відповідь: комірка із $k = 8$ (додатне число, знаковий формат) дозволяє зберігати число 45_{10} так : 00101101 .

2.1.4. Запис цілих знакових чисел в цифрових системах: прямий код від'ємних чисел

У разі запису від'ємних чисел в прямому коді у знаковий розряд поміщається цифра 1, яка означає знак «мінус», а в розряди цифрової частини числа – двійковий код абсолютної величини цього числа. Якщо модуль, який записується в мантису містить менше, ніж $k-1$ двійковий розряд, то наявні розряди записуються в мантису починаючи із наймолодшого (крайнього лівого), а незаповнені розряди мантиси заповнюються нулями.

Множина від'ємних чисел у такому випадку буде обмежена зверху нулем:

$$Max_k = 1000\ 0000_2 = 0.$$

Зверніть увагу, що через ознаку від'ємного числа (1 у старшому розряді), записане значення можна кваліфікувати як «-0».

Мінімальне значення можна отримати при одиницях в усіх $k-1$ цифрових розрядах мантиси і коли в знаковому розряді записано 1 (означає знак «мінус»):

$$Min_k = -\left(2^{k-1} - 1\right).$$

Приклад 2.7.

Визначити діапазон від'ємних значень, які можна записати за допомогою 1 байту, якщо числа зберігаються як такі, що можуть мати знак.

$$Max = 0$$

$$Min = \mathbf{1}\ 111\ 1111_2 = -127_{10}$$

Прямий код числа -127

Відповідь: комірка із $k = 8$ (від'ємне число, знаковий формат) дозволяє зберігати числа [-127…0].

Як можна було побачити із прикладів 2.5 та 2.7, в прямому коді маємо два представлення нуля, т.з. «плюс нуль» та «мінус нуль» (в згаданих прикладах це було 0000 0000 та 1000 0000). Така неоднозначність запису нуля може створювати певні труднощі під час виконання арифметичних операцій.

Отже, загальний діапазон знакових чисел, які можна записати за допомогою k двійкових розряди, становить:

$$\left[-\left(2^{k-1}-1\right) \ldots +\left(2^{k-1}-1\right)\right].$$

Приклад 2.8.

Записати число -9_{10} в комірку цифрової системи довжиною 1 байт, якщо вважається, що формат зберігання числа відповідає знаковому числу в прямому коді.

Число -9_{10} є від'ємним, тому в найстарший (крайній лівий) розряд записується одиницю. Переведемо модуль числа -9_{10} (тобто число 9_{10}) в двійкову систему числення:

$$9_{10} = 1001_2$$

В двійковому форматі число що розглядається містить тільки 4 розряди, в той час як для зберігання мантиси відведено $k-1=8-1=7$ розрядів, тому доповнюємо число трьома нулями зліва:

$$9_{10} = 1001_2 = 000\ \ 1001_2$$

Відповідь: комірка із $k = 8$ (від'ємне число, знаковий формат) дозволяє зберігати число -9_{10} наступним чином: 1000 1001 .

2.1.5. Представлення цілих знакових чисел в цифрових системах: обернений код від'ємних чисел

Обернений код (англ. ones' complement) від'ємних чисел можна отримати інвертуванням всіх цифр двійкового коду абсолютної величини (модуля) числа, включаючи розряд знаку: нулі замінюються одиницями, а одиниці – нулями. Пари протилежних за знаком чисел у оберненому коді доповнюють одне одного так, що у разі їх додавання у всіх розрядах утворяться одиниці. Така система отримала поширення завдяки простоті реалізації арифметико-логічного пристрою мікропроцесора, позаяк для віднімання числа від нуля слід було застосувати операцію логічного заперечення до усіх його розрядів, що є найпростішою логічною операцією. На ранніх етапах розвитку обчислювальної техніки існувало кілька машин, які виконували арифметичні операції саме у оберненому коді.

Діапазон можливих значень співпадає із діапазоном значень від'ємних чисел у прямому коді і становить:

$$\left[-\left(2^{k-1}-1\right) \ldots +\left(2^{k-1}-1\right)\right].$$

Так само, як і у випадку прямого коду, в оберненому коді маємо два представлення нуля, т.з. «плюс нуль» та «мінус нуль».

Приклад 2.9.

Записати число -127_{10} в комірку цифрової системи довжиною 1 байт, якщо вважається, що формат зберігання числа відповідає знаковому числу в оберненому коді.

Переведемо модуль числа -127_{10} (тобто число 127_{10}) в двійкову систему числення:

$$127_{10} = 111\ \ 1111_2.$$

В двійковому форматі число що розглядається містить 7 розрядів, тому мантису не потрібно доповнювати нулями, але потрібно записати нуль в розряд знаку:

$$127_{10} = 0111\ \ 1111_2 \text{ – це прямий код числа.}$$

Для переходу в обернений код нам потрібно виконати побітову інверсію кожного розряду, включаючи і знаковий розряд:

$127_{10} = 0111\ \ 1111_2$– це прямий код числа,

$-127_{10} = 1000\ \ 0000_2$– це обернений код числа.

Відповідь: комірка із $k = 8$ (від'ємне число, знаковий формат, обернений код) дозволяє зберігати число -127_{10} наступним чином: 1000 0000 .

Приклад 2.10.

Записати число -9_{10} в комірку цифрової системи довжиною 1 байт, якщо вважається, що формат зберігання числа відповідає знаковому числу в оберненому коді.

Переведемо модуль числа -9_{10} (тобто число 9_{10}) в двійкову систему числення:

$$9_{10} = 1001_2.$$

В двійковому форматі число, що розглядається, містить тільки 4 розряди, в той час як для зберігання мантиси відведено $k-1=8-1=7$ розрядів, тому доповнюємо число трьома нулями зліва та запишемо нуль в розряд знаку:

$9_{10} = 1001_2 = 0000\ \ 1001_2$ – це прямий код числа.

Для переходу в обернений код нам потрібно виконати побітову інверсію кожного розряду, включаючи і знаковий розряд:

$9_{10} = 0000\ \ 1001_2$ – це прямий код числа,

$-9_{10} = 1111\ \ 0110_2$ – це обернений код числа.

Відповідь: комірка із $k=8$ (від'ємне число, знаковий формат, обернений код) дозволяє зберігати число -9_{10} наступним чином: 1111 0110.

2.1.6. Представлення цілих знакових чисел в цифрових системах: додатковий код від'ємних чисел

Додатковий код від'ємних цілих чисел можна отримати додавши до оберненого коду числа одиницю, відкидаючи при цьому можливий розряд

переносу. У англомовній літературі такий код часто називається коротко two's complement, хоча насправді мається на увазі доповнення до 2^k, де k – кількість двійкових розрядів.

Максимальне значення, яке можна представити у такому форматі, буде дорівнювати –1.

Мінімальне значення яке можна записати, отже, дорівнює:

$$Min_k = -2^{k-1}.$$

Отже загальний діапазон знакових чисел, які можна записати за допомогою k двійкових розрядів, становить:

$$\left[-2^{k-1} \ \ldots \ +\left(2^{k-1}-1\right)\right].$$

Важливою особливістю представлення чисел в додатковому коді є те, що забезпечується єдине представлення нуля, що, в свою чергу, полегшує технічну реалізацію додавання чисел.

Приклад 2.11.

Записати число -127_{10} в комірку цифрової системи довжиною 1 байт, якщо вважається, що формат зберігання числа відповідає знаковому числу в додатковому коді.

Раніше, в прикладі 2.9 ми вже отримали обернений код числа -127_{10} :

$127_{10} = 1000\ \ 0000_2$– це обернений код числа.

Для переводу числа в додатковий код, потрібно додати одиницю до представлення в оберненому коді:

$127_{10} = 1000\ \ 0000_2$– це обернений код числа.

$127_{10} = 1000\ \ 0001_2$– це додатковий код числа.

Відповідь: комірка із $k = 8$ (від'ємне число, знаковий формат, додатковий код) дозволяє зберігати число -127_{10} наступним чином: 1000 0001 .

Приклад 2.12.

Записати число -9_{10} в комірку цифрової системи довжиною 1 байт, якщо вважається, що формат зберігання числа відповідає знаковому числу в додатковому коді.

В прикладі 2.10 було отримано обернений код заданого числа. Додаючи до нього одиницю отримаємо представлення в додатковому коді:

$-9_{10} = 1111\ \ 0110_2$ – це обернений код числа,

$-9_{10} = 1111\ \ 0111_2$ – це додатковий код числа.

Відповідь: комірка із $k = 8$ (від'ємне число, знаковий формат, додатковий код) дозволяє зберігати число -9_{10} наступним чином: 1111 0111 .

Один і той же набір бітів довжиною k можна інтерпретувати по-різному: або як ціле число без знаку, або як ціле число зі знаком (в прямому, оберненому або додатковому коді).

Комп'ютер не знає, як потрібно інтерпретувати той чи інший бітовий набір – для нього це просто набір бітів, а сенс цього набору відомий програмісту, який працює із цими даними.

В таблиці 2.2 можна побачити, що одні й ті ж самі біти можуть бути інтерпретовані зовсім по-різному.

Таблиця 2.2. Приклади інтерпретацій бітових наборів довжиною $k = 3$ двійковим розрядам.

Набір бітів	Як інтерпретувати			
	Число без знаку	Число зі знаком в прямому коді	Число зі знаком в оберненому коді	Число зі знаком в додатковому коді
000	0	+0	+0	0
001	1	+1	+1	+1
010	2	+2	+2	+2
011	3	+3	+3	+3
100	4	-0	-3	-4
101	**5**	**-1**	**-2**	**-3**
110	6	-2	-1	-2
111	7	-3	-0	-1

В таблиці 2.2 рядок, із зеленим шрифтом, демонструє як можна інтерпретувати набор бітів 101_2:

- ✓ Ці біти можна розглядати як число без знаку. Тоді ми просто перейдемо від двійкового числа 101 до десяткового представлення і отримаємо 5.
- ✓ Якщо розглядати 101 число зі знаком в прямому коді, то крайній лівий розряд буде розглядатися як знаковий, тобто це буде від'ємне число. Інші розряди, тобто 01 будуть розглядатися як модуль числа у двійковому представленні. Тобто отримаємо число -1.
- ✓ Нехай 101 – це знакове число в оберненому коді (четвертий стовбець). Крайній лівий розряд дорівнює одиниці, тобто це буде від'ємне число. Отримаємо модуль числа інвертуванням

усіх розрядів. 101 перетвориться на 010, що дорівнює 2 у десятковій системі числення. Тобто якщо розглядати число 101 як знакове число в оберненому коді, то значення цього числа буде -2.

- ✓ Якщо розглядати 101 як знакове число в додатковому коді, то отримаємо від'ємне число, модуль якого можна знайти спочатку віднявши одиницю, а потім інвертувавши усі біти результату: 101 – 1 = 100. А тепер інверсії отримаємо 011. Тобто в такій інтерпретації це буде число -3.

Нехай x – це число зі знаком. Нижче наведені формальні математичні формули визначення оберненого та додаткового коду числа.

$$\textit{обернений}(x) = \begin{cases} x, & \textit{якщо } x \geq 0; \\ 2^k - 1 - |x|, & \textit{якщо } x < 0; \end{cases}$$

$$\textit{додатковий}(x) = \begin{cases} x, & \textit{якщо } x \geq 0; \\ 2^k - |x|, & \textit{якщо } x < 0. \end{cases}$$

2.2. Завдання

Номер варіанта потрібно обирати за наступною формулою:

$$N = K + M ,$$

де N – номер варіанту,

$K = 0$ для груп ДМ-х1 та ДП-х1 та $K = 20$ для груп ДМ-х2 та ДП-х2,

M – номер із залікової книжки.

Навести формат зберігання наведених в завданні пар чисел в:

а. у прямому коді,

b. в оберненому коді,

c. у додатковому коді.

Примітка: використовувати розрядність k = 9 (8 розрядів для зберігання модуля числа та 1 розряд для зберігання знаку).

Номер варіанта	Завдання1	Завдання2	Завдання3	Завдання4
1	255 та -24	15 та -199	33 та -127	229 та -13
2	161 та -62	98 та -4	77 та -126	69 та -221
3	223 та -97	198 та -25	134 та -125	253 та -23
4	179 та -34	75 та -222	242 та -1	252 та -96
5	124 та -250	232 та -61	224 та -95	12 та -175
6	94 та -251	29 та -123	228 та -22	159 та -26
7	27 та -248	177 та -2	5 та -122	220 та -11
8	142 та -8	140 та -160	21 та -197	119 та -68
9	32 та -178	25 та -241	121 та -6	158 та -28
10	182 та -33	162 та -100	51 та -176	21 та -227
11	156 та -59	3 та -120	20 та -254	225 та -92
12	255 та -93	183 та -30	27 та -157	60 та -230
13	240 та -22	226 та -74	58 та -141	10 та -119
14	32 та -118	91 та -254	29 та -196	155 та -19
15	195 та -13	73 та -227	28 та -117	140 та -15
16	116 та -244	247 та -18	228 та -90	180 та -30
17	153 та -56	50 та -194	253 та -115	229 та -154
18	78 та -250	31 та -152	23 та -181	193 та -17
19	231 та -20	17 та -72	15 та -139	230 та -89
20	249 та -77	55 та -184	144 та -16	57 та -239
21	246 та -14	250 та -88	54 та -138	231 та -114
22	87 та -243	143 та -16	32 та -251	67 та -248
23	225 та -13	71 та -246	137 та -52	232 та -86

Номер варіанта	Завдання1	Завдання2	Завдання3	Завдання4
24	76 та -244	238 та -34	113 та -7	185 та -9
25	136 та -17	53 та -112	11 та -145	245 та -66
26	167 та -4	33 та -226	35 та -233	164 та -6
27	243 та -85	169 та -12	165 та -14	8 та -110
28	146 та -13	84 та -242	5 та -135	224 та -3
29	18 та -168	252 та -36	40 та -166	233 та -109
30	134 та -10	41 та -70	83 та -241	39 та -174
31	237 та -7	12 та -170	64 та -186	108 та -234
32	107 та -38	133 та -19	1 та -106	163 та -245
33	239 та -82	2 та -223	37 та -172	105 та -24
34	171 та -42	65 та -104	9 та -132	46 та -236
35	81 та -238	8 та -151	26 та -221	11 та -103
36	249 та -10	131 та -20	80 та -237	3 та -187
37	236 та -99	4 та -192	150 та -63	235 та -79
38	188 та -48	1 та -130	244 та -5	173 та -43
39	102 та -31	45 та -222	25 та -147	190 та -47
40	148 та -9	101 та -234	129 та -6	44 та -251
41	7 та -243	128 та -49	8 та -189	100 та -149

2.3. Контрольні питання

1. Чи відрізняється формат зберігання цілих додатних беззнакових та цілих додатних знакових чисел?
2. Скільки байтів потрібно для збереження цілого беззнакового числа 300 без втрати даних?
3. Якщо спробувати записати число, довжина якого 10 бітів, в комірку пам’яті довжиною 1 байт, чи відбудеться втрата даних?
4. В якому форматі чи форматах є два представлення нуля?
5. Яким значенням зазвичай кодується «+», а яким «–» в знакових форматах зберігання цілих чисел?

3. Арифметичні операції над цілими числами

Мета роботи: навчитися виконувати арифметичні дії над цілими двійковими числами, записаними у різних форматах.

3.1. Теоретичні відомості

3.1.1. Арифметичні операції над цілими числами без знаку

Додавання та віднімання беззнакових цілих чисел відбувається за звичайними для позиційних систем числення алгоритмами, якщо у процесі виконання не виникають від'ємні числа. Ці операції виконуються в двійковій системі числення.

Приклад 3.1.

Виконати додавання двох чисел 1 та 4 в цифровій системі, де зберігаються цілі числа без знаку та $k = 3$.

Спочатку обидва операнди потрібно перевести до формату аналізованої системи, тобто числа потрібно записати у двійковій системі числення у форматі цілих беззнакових із $k = 3$:

$$1_{10} = 001_2;$$
$$4_{10} = 100_2.$$

Далі потрібно додати усі розряди за звичайними правилами двійкової арифметики:

$$\begin{array}{r} 001 \\ +\ 100 \\ \hline 101 \end{array}.$$

Відповідь: 101, що, очевидно, відповідає $1_{10}+4_{10} = 5_{10}$.

Зазначимо, що у процесі виконання розглянутої операції не виникло нестандартних ситуацій, як от переповнення розрядної сітки. Однак реальні обчислювальні системи повинні мати засоби реагування на такі ситуації.

Приклад 3.2

Виконати наступну операцію: 5 – 3 в цифровій системі, де зберігаються цілі числа без знаку та $k = 3$.

Спочатку обидва операнди потрібно перевести в формат, притаманний системі що розглядається, тобто перевести числа у двійкову систему числення із $k = 3$ та записати у форматі цілих беззнакових:

$$5_{10} = 101_2;$$
$$3_{10} = 011_2.$$

Далі потрібно відняти усі розряди за звичайними правилам двійкової арифметики:

$$\begin{array}{r} 101 \\ -\ 011 \\ \hline 010 \end{array}.$$

Відповідь: 010, що, очевидно відповідає $5_{10} – 3_{10} = 2_{10}$.

Ситуації, коли зменшуване менше від’ємника або коли результат суми не вміщується в k розрядів, вважаються помилковими та повинні відслідковуватися цифровою системою. Реакція на такі помилки залежить від розробника цифрової системи.

У деяких обчислювальних системах множення проводиться як послідовність додавань та зсувів, проте існують і апаратні засоби для обчислення добутку.

Ділення для комп’ютера є більш складною операцією та реалізується за спеціальними алгоритмами програмно або за допомогою апаратного прискорювача.

3.1.2. Арифметичні операції над цілими числами в прямому коді із знаком

Додавання додатних цілих чисел відбувається за звичайними для позиційних систем числення алгоритмами. Віднімання цілих чисел із знаком в прямому коді або виконання додавання в ситуації коли операнди мають різні знаки, виконується за наступним алгоритмом:

1. Спочатку порівнюються модулі операндів і знак числа із більшим модулем становиться знаком результату.
2. Від числа із більшим модулем віднімається число із меншим модулем.

Всі операції виконуються в двійковій системі числення.

Приклад 3.3.

Виконати додавання двох чисел 25 та –4 в цифровій системі, де зберігаються цілі числа із знаком в прямому коді та $k = 8$.

Спочатку обидва операнди потрібно перевести в двійковий формат цілих чисел із знаком, коли $k = 8$:

$$25_{10} = 0001\ \ 1001_2;$$
$$-4_{10} = 1000\ \ 0100_2.$$

Порівняємо модулі операндів: $25 > 4$. Операнд 25 має знак «+», тому результат також буде додатним числом, тобто в крайньому лівому розряді результату потрібно записати 0.

Далі потрібно відняти $k-1=7$ розрядів мантиси операнда із меншим модулем від 7 розрядів операнда із більшим модулем за звичайними правилами двійкової арифметики:

$$\begin{array}{r} 001\ 1001 \\ -\ 000\ 0100 \\ \hline 001\ \ 0101 \end{array}.$$

Залишилося тільки додати крайній лівий розряд знаку: **0**001 0101

Відповідь: 0001 0101, що відповідає $25_{10} - 4_{10} = 21_{10}$.

Приклад 3.4.

Виконати додавання двох чисел –105 та 17 в цифровій системі, де зберігаються цілі числа із знаком в прямому коді та $k=8$.

Спочатку обидва операнди потрібно перевести в двійковий формат цілих чисел із знаком, коли $k=8$:

$$-105_{10} = 1110\ \ 1001_2;$$
$$17_{10} = 0001\ \ 0001_{2.}$$

Порівняємо модулі операндів: $105>17$. Операнд –105 має знак «–», тому результат також буде від'ємним числом, тобто в крайньому лівому розряді результату потрібно записати 1.

Далі потрібно відняти $k-1=7$ розрядів мантиси операнда із меншим модулем від 7 розрядів операнда із більшим модулем за звичайними правилами двійкової арифметики:

$$\begin{array}{r} 110\ 1001 \\ -\ 001\ 0001. \\ \hline 101\ 1000 \end{array}$$

Залишилося тільки додати крайній лівий розряд знаку: **1**101 1000

Відповідь: 1101 1000, що відповідає $-105_{10} + 17_{10} = 88_{10}$.

3.1.3. Арифметичні операції над цілими числами в оберненому коді

Додавання в оберненому коді відбувається наступним чином: за звичайним алгоритмом додаються всі розряди, включаючи знаковий. Результат додавання двох чисел довжиною k розрядів, в загальному випадку, може мати довжину $k+1$ (крайній лівий розряд буде дорівнювати одиниці, якщо був перенос при додаванні старших розрядів операндів, а якщо переносу не було, то $k+1$-й розряд буде дорівнювати нулю).

Після цього значення лівого $k+1$-го розряду додається до молодшого розряду результату. Отримуємо k-розрядне число, яке і буде сумою двох чисел в оберненому коді. Фактично, операція віднімання зведена до двох послідовних операцій додавання, спочатку пари операндів, потім – результату з розрядом переносу. Відтак віднімач можна реалізувати за допомогою пари суматорів. У результаті додавання пари додатних чисел за умови, що переповнення не відбувається, знак суми теж буде додатним, тому у можливому розряді переносу залишатиметься нуль і його формальне додавання на другому кроці не спотворюватиме результат.

Операція віднімання чисел у оберненому коді $x-y$ еквівалентна операції додавання $x+(-y)$.

Приклад 3.5.

Виконати додавання двох чисел –105 та 17 в цифровій системі, де зберігаються цілі числа із знаком в оберненому коді та $k=8$.

Спочатку модулі обох операндів потрібно перевести у двійковий код з $k=8$:

$$105_{10} = 0110\ \ 1001_2;$$
$$17_{10} = 0001\ \ 0001_2.$$

Знайдемо обернений код обох операндів. Оскільки число –105 – від'ємне, то в ньому інвертується кожен розряд:

$$105_{10} = 0110\ \ 1001_2 \ - \quad \text{прямий код;}$$
$$-105_{10} = 1001\ \ 0110_2 - \text{обернений код.}$$

Обернений код додатного число 17 співпадає із прямим і в крайній лівий знаковий розряд записується 0:

$$17_{10} = \ 001\ \ 0001_2 \ - \quad \text{прямий код;}$$
$$17_{10} = 0001\ \ 0001_2 - \text{обернений код.}$$

Далі потрібно додати усі $k=8$ розрядів обох операндів на пристрої розрядністю $k+1$:

$$\begin{array}{r} 0\,1001\ 0110 \\ +\ 0\,0001\ 0001 \\ \hline 0\,1010\ \ 0111 \end{array}.$$

Після цього $k+1$-й розряд переносу слід додати до результату і отримати результат шириною $k=8$:

$$\begin{array}{r} 1010\ 0111 \\ +\qquad\quad 0 \\ \hline 1010\ \ 0111 \end{array}.$$

У розглянутому прикладі $k+1$-й розряд переносу дорівнює нулю, його додавання не вплинуло на результат операції, проте апаратні засоби формально виконують і цей крок.

Переконаємося в правильності отриманого результату. Результат, так само як і операнди, записаний в оберненому коді. Крайній лівий розряд дорівнює одиниці, тобто це від'ємне число. Проінвертуємо усі розряди результату щоб дізнатися модуль результату:

$$10100111 \xrightarrow{\text{інверсія}} 01011000$$

Перейдемо в десяткову систему числення: $01011000_2 = 88_{10}$, тобто результат дорівнює –88.

Виконаємо розрахунок в десятковій системі числення: –105+17 = -88. Результати, отримані в двійковому форматі (обернений код) та в десятковому форматі – однакові.

Відповідь: 1010 0111.

Приклад 3.6.

Виконати додавання двох чисел 120 та -25 в цифровій системі, де зберігаються цілі числа із знаком в оберненому коді та $k=8$.

Спочатку модулі обох операндів потрібно перевести в двійковий формат, коли $k=8$:

$$120_{10} = 0111\ \ 1000_2$$
$$25_{10} = 0001\ \ 1001_2$$

Порахуємо обернений код обох операндів. Обернений код додатного числа 120 співпадає із прямим і в крайній лівий знаковий розряд записується 0:

$$120_{10} = \ 111\ \ 1000_2\ - \quad \text{прямий код}$$
$$120_{10} = 0111\ \ 1000_2\ - \text{обернений код}$$

Оскільки число -25 – від'ємне, то в ньому інвертується кожен розряд:

$$25_{10} = 0001\ \ 1001_2 - \quad \text{прямий код}$$
$$-25_{10} = 1110\ \ 0110_2 - \text{обернений код}$$

Далі потрібно додати усі $k = 8$ розрядів обох операндів:

$$\begin{array}{r} 0111\ \ 1000 \\ +\ 1110\ \ 0110 \\ \hline 1\ 0101\ \ 1110 \end{array}$$

Результат виконання операції додавання має $9 = k + 1$ розрядів, тому потрібно додати $k + 1$-й розряд переносу (має синій колір в останньому виразі) додати до $k = 8$ розрядів результату:

$$\begin{array}{r} 0101\ \ 1110 \\ + \qquad\qquad 1. \\ \hline 0101\ \ 1111 \end{array}$$

Це і є остаточний результат розрахунку.

Переконаємося в правильності отриманого результату. Результат, так само як і операнди, записаний в оберненому коді. Крайній лівий розряд дорівнює нулю, тобто це додатне число, тому просто переведемо це число із двійкового формату в десятковий:

$01011111_2 = 95_{10}$, тобто результат дорівнює +95.

Виконаємо розрахунок в десятковій системі числення: $120 - 25 = 95$. Результати, отримані в двійковому форматі (обернений код) та в десятковому форматі – однакові.

Відповідь: 0101 1111.

3.1.4. Арифметичні операції над цілими числами в додатковому коді

Додавання в додатковому коді виконується за наступним алгоритмом: за правилами двійкової системи числення додаються всі розряди, включаючи знаковий; а одиниця переносу в $k+1$-ому розряді відкидається. Такий спосіб ще називають додавання за модулем 2. Головними перевагами є можливість отримати результат операції за одну стадію там можливість використання апаратних суматорів розрядністю k.

Операція віднімання чисел у оберненому коді $x - y$ також зводиться до додавання $x + (-y)$.

Множення та ділення чисел в оберненому та додатковому кодах виконуються подібно множенню та діленню цілих чисел без знаку.

Приклад 3.7.

Виконати додавання двох чисел 120 та –25 в цифровій системі, де зберігаються цілі числа із знаком в додатковому коді та $k = 8$.

Спочатку модулі обох операндів потрібно перевести у двійкову систему числення із $k = 8$:

$$120_{10} = 0111\ \ 1000_2;$$
$$25_{10} = 0001\ \ 1001_2.$$

Знайдемо додатковий код обох операндів. Обернений та додатковий коди додатного числа 120 співпадають із прямим і в крайній лівий знаковий розряд записується 0:

$$120_{10} = \ 111\ \ 1000_2\ - \quad \text{прямий код;}$$
$$120_{10} = 0111\ \ 1000_2\ - \quad \text{обернений код;}$$
$$120_{10} = 0111\ \ 1000_2\ - \text{додатковий код.}$$

Оскільки число –25 – від'ємне, то для отримання оберненого коду в ньому інвертується кожен розряд, а потім до молодшого розряду додається 1 для отримання додаткового коду:

$$25_{10} = 0001\ \ 1001_2 - \quad \text{прямий код}$$
$$-25_{10} = 1110\ \ 0110_2 - \text{обернений код}$$
$$-25_{10} = 1110\ \ 0111_2 - \text{додатковий код}$$

Далі потрібно додати усі $k = 8$ розрядів обох операндів:

$$\begin{array}{r} 0111\ \ 1000 \\ +\ 1110\ \ 0111 \\ \hline 1\ 0101\ \ 1111 \end{array}$$

Результат виконання операції додавання має $9 = k + 1$ розрядів, крайній лівий $k+1$-й розряд переносу відкидається, тоді результат обчислення буде мати вигляд: $\cancel{1}\ 0101\ \ 1111 \longrightarrow 0101\ \ 1111$. Це і є остаточний результат розрахунку.

Переконаємося в правильності отриманого результату. Результат, так само як і операнди, записаний в оберненому коді. Крайній лівий розряд дорівнює нулю, тобто це додатне число, тому просто переведемо це число із двійкової системи числення у десяткову:

$01011111_2 = 95_{10}$, тобто результат дорівнює +95.

Виконаємо розрахунок в десятковій системі числення: 120-25 = 95. Результати, отримані в двійковому форматі (обернений код) та в десятковому форматі – однакові.

Відповідь: 0101 1111.

3.1.5. Помилки у процесі виконання арифметичних операцій на комп'ютері

Під час виконання арифметичних операцій можливе виникнення ситуацій, які відхиляються від усталених правил. Наприклад, старші розряди результату операції можуть не поміщатися до відведеної для цього області пам'яті. Розглянемо цю ситуацію на прикладах.

Приклад 3.8.

Виконати додавання двох цілих беззнакових чисел 3 та 5, коли $k = 3$.

В двійковій системі числення із $k = 3$ маємо:

$$3_{10} = 011_2;$$
$$5_{10} = 101_2.$$

Додамо операнди:

$$\begin{array}{r} 011 \\ +\ 101 \\ \hline 1\ 000 \end{array}$$

Результат обчислень містить 4 розряди, але, за умовою задачі, для зберігання чисел в цифровій системі, яку ми розглядаємо, відводиться тільки $k=3$ розряди. Тому крайній лівий розряд буде втрачено і результат обчислення буде мати вигляд: $\cancel{1}\ 000 \longrightarrow 000 = 0$.

Відповідь: 0.

Така ситуація називається переповненням цифрової частини (мантиси) і виникає коли старші розряди результату операції не поміщаються у відведену для нього області пам'яті. Обчислювальні машини можуть сповіщати користувача про виникнення подібних помилок або просто встановлювати ознаку переповнення в спеціальному регістрі стану процесора (англ. FLAGS).

Приклад 3.9.

Виконати додавання двох цілих знакових чисел –3 та –2 в додатковому коді, коли $k=3$.

В двійковому коді з $k=3$ операнди в додатковому коді мають вигляд:

$$3_{10} = 011_2 \xrightarrow{\text{обернений код}} 100 \xrightarrow{\text{додатковий код}} 101$$

$$2_{10} = 010_2 \xrightarrow{\text{обернений код}} 101 \xrightarrow{\text{додатковий код}} 110$$

Виконаємо додавання:

$$\begin{array}{r} 101 \\ +\ 110 \\ \hline 1\ 011 \end{array}$$

Результат обчислення містить 4 розряди, але, за умовою задачі, для зберігання чисел в цифровій системі, яку ми розглядаємо, відводиться тільки $k=3$ розряди. Тому крайній лівий розряд буде втрачено і результат розрахунку буде мати наступний вигляд: $\cancel{1}\ 011 \longrightarrow 011 = +3_{10}$.

Такий результат є помилкою і пов'язаний із недостатньою кількістю двійкових розрядів, які відведені для зберігання інформації в цифровій системі.

3.2. Завдання

Номер варіанта потрібно обирати за формулою:

$$N = K + M ,$$

де N – номер варіанту,

$K = 0$ для груп ДМ-х1 та ДП-х1 та $K = 20$ для груп ДМ-х2 та ДП-х2,

M – номер із залікової книжки.

Виконати додавання наведених в завданні пар чисел:

a. у прямому коді,
b. в оберненому коді,
c. у додатковому коді.

Примітка: використовувати розрядність k = 9 (8 розрядів для зберігання модуля числа та 1 розряд для зберігання знаку).

Номер варіанта	Завдання1	Завдання2	Завдання3	Завдання4
1	255 та -24	15 та -199	33 та -127	229 та -13
2	161 та -62	98 та -4	77 та -126	69 та -221
3	223 та -97	198 та -25	134 та -125	253 та -23
4	179 та -34	75 та -222	242 та -1	252 та -96
5	124 та -250	232 та -61	224 та -95	12 та -175
6	94 та -251	29 та -123	228 та -22	159 та -26
7	27 та -248	177 та -2	5 та -122	220 та -11

Номер варіанта	Завдання1	Завдання2	Завдання3	Завдання4
8	142 та -8	140 та -160	21 та -197	119 та -68
9	32 та -178	25 та -241	121 та -6	158 та -28
10	182 та -33	162 та -100	51 та -176	21 та -227
11	156 та -59	3 та -120	20 та -254	225 та -92
12	255 та -93	183 та -30	27 та -157	60 та -230
13	240 та -22	226 та -74	58 та -141	10 та -119
14	32 та -118	91 та -254	29 та -196	155 та -19
15	195 та -13	73 та -227	28 та -117	140 та -15
16	116 та -244	247 та -18	228 та -90	180 та -30
17	153 та -56	50 та -194	253 та -115	229 та -154
18	78 та -250	31 та -152	23 та -181	193 та -17
19	231 та -20	17 та -72	15 та -139	230 та -89
20	249 та -77	55 та -184	144 та -16	57 та -239
21	246 та -14	250 та -88	54 та -138	231 та -114
22	87 та -243	143 та -16	32 та -251	67 та -248
23	225 та -13	71 та -246	137 та -52	232 та -86
24	76 та -244	238 та -34	113 та -7	185 та -9
25	136 та -17	53 та -112	11 та -145	245 та -66
26	167 та -4	33 та -226	35 та -233	164 та -6
27	243 та -85	169 та -12	165 та -14	8 та -110
28	146 та -13	84 та -242	5 та -135	224 та -3
29	18 та -168	252 та -36	40 та -166	233 та -109
30	134 та -10	41 та -70	83 та -241	39 та -174
31	237 та -7	12 та -170	64 та -186	108 та -234
32	107 та -38	133 та -19	1 та -106	163 та -245
33	239 та -82	2 та -223	37 та -172	105 та -24
34	171 та -42	65 та -104	9 та -132	46 та -236

Номер варіанта	Завдання1	Завдання2	Завдання3	Завдання4
35	81 та -238	8 та -151	26 та -221	11 та -103
36	249 та -10	131 та -20	80 та -237	3 та -187
37	236 та -99	4 та -192	150 та -63	235 та -79
38	188 та -48	1 та -130	244 та -5	173 та -43
39	102 та -31	45 та -222	25 та -147	190 та -47
40	148 та -9	101 та -234	129 та -6	44 та -251
41	7 та -243	128 та -49	8 та -189	100 та -149

3.3. Контрольні питання

1. Які найпоширеніші коди використовуються для роботи з від’ємними числами?
2. У чому полягають переваги та недоліки оберненого коду від’ємних чисел?
3. Які переваги додаткового коду від’ємних чисел?
4. Скільки бітових комбінацій можуть означати нуль у прямому, оберненому та додатковому коді?
5. Які аварійні ситуації можуть траплятися при виконанні арифметичних операцій із цілими знаковими числами?

4. Зберігання дійсних чисел в цифрових системах

Мета роботи: навчитися виконувати обчислення із дробовими числами в двійковому форматі

4.1. Теоретичні відомості

Найпоширенішими способами зберігання дійсних чисел в цифрових системах є формат з фіксованою крапкою (комою) та формат із плаваючою точкою.

Формат із фіксованою комою передбачає, що для запису цілої та дробової частин числа відводиться фіксована кількість розрядів. У разі використання формату із фіксованою комою ще на етапі конструювання цифрової системи визначається, скільки і які розряди машинного слова відведені під збереження цілої та дробової частин числа. Кома в розрядній сітці, теоретично, може бути зафіксована після будь-якого розряду. Загальний вигляд формату зберігання дійсних чисел із фіксованою комою в цифрових системах наведено на рис. 4.1.

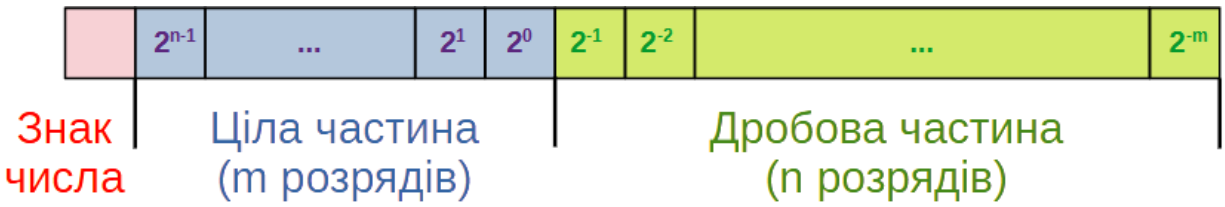


Рис. 4.1. Формат зберігання дійсних чисел із фіксованою комою в обчислювальній техніці

В цьому випадку запис цілих чисел можна вважати окремим випадком запису дійсних чисел із фіксованою комою, коли усі розряди окрім знакового використовуються для запису тільки цілої частини числа.

До переваг використання формату зберігання чисел з фіксованою комою відноситься простота виконання арифметичних операцій та висока точність відображення чисел, але основний недолік цього формату – невеликий діапазон чисел, які можна зберігати в такому форматі.

Довідка.

Навіть з використанням апаратних математичних сопроцесорів, що характерно для сучасних процесорів загального призначення, операції множення та ділення дійсних чисел у форматі з плаваючою комою вимагають кількох тактів і навіть операції додавання та віднімання потребують додаткового кроку вирівнювання мантис. Натомість операції множення цілих чисел з використанням апаратних перемножувачів можуть виконуватися за один такт. Тому у спеціалізованих мікропроцесорах, зокрема цифрових сигнальних процесорах (англ. DSP – Digital Signal Processor), для збереження та роботи з дійсними числами використовують так звані Q-формати [1]. Так, формат Qm.n означає, що для запису цілої частини відводиться m двійкових розрядів, для дробової – n розрядів. Один розряд потрібен для збереження знаку, тому для зберігання дійсного числа у Qm.n форматі потрібно $m+n+1$ розрядів. Діапазон чисел, які можна записати у такому форматі, становить $\left(-2^m;2^m\right)$, а ціна молодшого розряду дробової частини становить 2^{-n}. Виконання арифметичних операцій у Q-форматі, звісно, потребує врахування можливого зростання мантиси результату арифметичної операції. Так, добуток пари чисел у форматі Q3.12 може давати результат із 6 розрядами цілої частини. Якщо відведених розрядів недостатньо, може відбуватися переповнення.

Формат із плаваючою точкою передбачає , що для запису чисел, які мають дробову частину, використовують форму запису з порядком основи системи числення, тобто будь-яке число N в системі числення з основою q можна записати так:

$$N = M \cdot q^p,$$

де M – це мантиса,

q – це основа системи числення ,

p – порядок.

Такий спосіб запису називають записом із плаваючою комою.

Приклад 4.1.

Записати декілька варіантів запису числа $1{,}25_{10}$ використавши форму запису з порядком основи системи числення.

1) $1{,}25_{10} = 1{,}25 \cdot 10^0$
2) $1{,}25_{10} = 0{,}125 \cdot 10^1$
3) $1{,}25_{10} = 125 \cdot 10^{-2}$

За зображенням чисел в деяких мовах програмування можна зрозуміти чи це ціле число, чи це дробове. Наприклад якщо написано просто 5 – це ціле, а запис подібний до 5.0 – підказує компілятору, що це число потенційно може мати дробову частину.

Якщо кома у мантисі розташована після першої цифри, яка більша від нуля ($M \in [1, 9)$ у випадку десяткового представлення та $M \in [1, 2)$ у випадку двійкового запису), то при такій формі запису у відведені розряди поміститься максимальна кількість цифр числа, тобто число буде відображено максимально точно. Таке відображення називається ***нормалізованим***.

Зазвичай, в обчислювальній техніці, числа, які мають дробову частину, зберігаються в *двійковому нормалізованому вигляді*.

Приклад 4.2.

Представити в нормалізованому вигляді наступні десяткові числа: 152,15 та -0,039.

Для нормалізації числа потрібно зробити мантису такою, щоб вона мала цілу частину менше 10:

1) $152,15 = 1,5215 \cdot 10^{2}$;
2) $-0,039 = -3,9 \cdot 10^{-2}$.

Приклад 4.3.

Представити в нормалізованому вигляді двійкові числа: –101,01 та 0,000011.

Зауважимо, що в цих прикладах для запису порядку також використовується двійкова система числення.

Число –101,01 в нормалізованому вигляді буде мати мантису -1,0101, тобто для нормалізації потрібно посунути кому на 2 двійкові розряди ліворуч. Переведемо число 2_{10} в двійкову систему числення, оскільки порядок також потрібно записувати в двійковому форматі: $2_{10} = 10_2$. Отже, нормалізований вигляд числа –101,01 буде

$$–101,01 = –1,0101 \cdot 2^{-10}.$$

Для нормалізації мантиси числа 0,000011 потрібно посунути кому на 5 розрядів праворуч: $5_{10} = 101_2$. Отже, нормалізований вигляд числа 0,000011 буде

$$0,000011 = 1,1 \cdot 2^{-101}.$$

Як правило, числа із дробовою частиною зберігаються в цифрових системах у вигляді бітових наборів, в яких відводяться розряди для мантиси, порядку та знаку числа, як це показано на рис. 4.2.

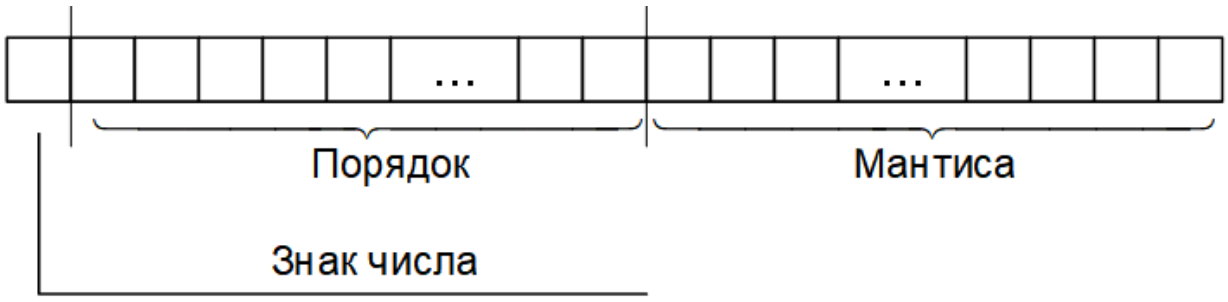


Рис. 4.2. Формат зберігання дійсних чисел із плаваючою точкою в обчислювальній техніці.

Що більше розрядів відводиться під запис мантиси, тим вищою буде точність представлення числа. Що більше розрядів займає порядок, тим ширший діапазон від найменшого до найбільшого числа можна записати.

Під зберігання знаку відводиться 1 біт, а кількість бітів під мантису та порядок – залежить від обраного типу даних та реалізації обчислювальної машини. Так само як і у випадку цілих чисел, для додатних чисел біт знаку дорівнює нулю, а для від'ємних – одиниці. Серед бітів, відведених під зберігання порядку також є біт, який несе інформацію про знак порядку.

Сучасні цифрові пристрої та програмне забезпечення, які працюють із дійсними числами, використовують уніфікований формат зберігання таких чисел, який описаний в спеціальному стандарті – *IEEE Standard for Floating-Point Arithmetic (IEEE 754)* [2].

Відповідно до цього стандарту у поле порядку інформація записується в дещо зміненому вигляді – до реального значення додається певна константа (яка залежить від конкретного формату), отже всі порядки зберігаються у вигляді додатних чисел, так званий offset .

В полі мантиси інформація також зберігається в зміненому вигляді – у відведених розрядах зберігається «хвіст» реальної мантиси. Тобто в полі мантиси записуються всі дробові цифри мантиси (окрім найстаршої двійкової одиниці, яка, фактично, є цілою частиною). Ціла частина мантиси у нормалізованому двійковому представлення завжди дорівнює 1, тому немає сенсу її зберігати. Це так звана «прихована одиниця».

Більш детально про способи зберігання дійсних чисел у цьому форматі можна подивитися у Додаток А. Основні положення стандарту IEEE 754.

Приклад 4.4.

Записати вигляд машинного слова довжиною 32 біта (*Single precision* – одинарна точність), якщо в ньому записати десяткове число 0.3125_{10}.

Інформація стосовно формату зберігання даних із одинарною точністю наведена в Додаток А. Основні положення стандарту IEEE 754.

Перейдемо від десяткового представлення наведеного числа до двійкового:

$$0.3125_{10} = 0,25 + 0,0625 = \frac{1}{4} + \frac{1}{16} = \frac{1}{2^2} + \frac{1}{2^4} = 2^{-2} + 2^{-4} = 0.0101_2$$

Отримане двійкове представлення потребує нормалізації:

$$0.0101_2 = 1.01 \cdot 2^{-2}.$$

Як описано в Додатку А (параграф 8.1.2) зміщення порядку дорівнює 127, тому до реального значення порядку (–2) додаємо зміщення порядку і отримаємо:

$$-2+127=125_{10}.$$

Оскільки це значення також зберігається в двійковому вигляді, перейдемо двійкової системи:

$$125_{10}=1111101_2.$$

Саме ця величина запишеться в поле порядку. В форматі *Single precision* для зберігання порядку відводиться 8 двійкових розрядів, а число 125_{10} в двійковому представленні має тільки 7 значущих розрядів, тому двійкове представлення числа 125_{10} доповнюється одним нулем зліва:

$$E=01111101.$$

Сформуємо «хвіст» мантиси, який використовується для зберігання, прибравши старшу одиницю із нормалізованого двійкового представлення реальної мантиси, тобто з 1.0.1 буде збережено тільки 01_2. В форматі Single precision для зберігання хвоста мантиси відводиться 23 двійкових розряди, тому реальний вигляд цього поля

$$T=01000000000000000000000.$$

Число 0.3125_{10} є додатним, тому в знак порядку запишеться нуль:

$$S=0.$$

Отже, остаточний вигляд машинного слова, в якому зберігається число 0.3125_{10} у форматі *Single precision*, наведено на рис. 4.3.

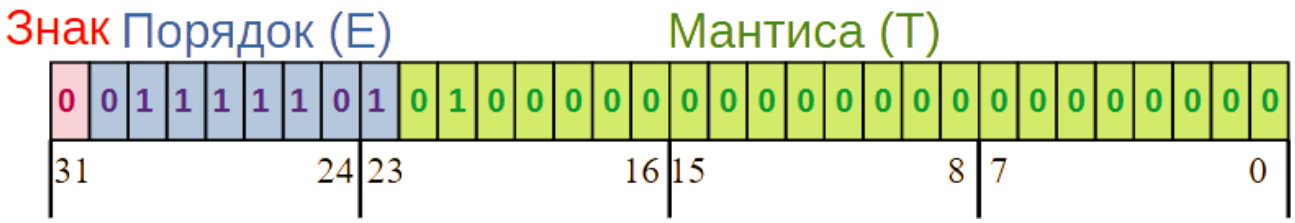


Рис. 4.3. Представлення десяткового числа 0.3125_{10} згідно стандарту *IEEE 754* у форматі *Single precision*.

Розглянемо як виконується додавання та віднімання дійсних чисел в нормалізованому форматі.

Перед початком виконання арифметичних операцій потрібно виконати додатковий крок, який називається ***вирівнюванням порядків***. Для цього мантиса числа з меншим порядком зсувається в своєму регістрі праворуч. Причому кількість зсувів буде дорівнювати різниці порядків операндів. Після кожного зсуву порядок операнду, для якого було виконано зсув, збільшується на одиницю.

В результаті вирівнювання порядків однойменні розряди мантис чисел виявляються розташованими у відповідних розрядах обох регістрів, після чого мантиси додаються або віднімаються, а порядок результату не змінюється. За необхідності отриманий результат нормалізується шляхом зсуву мантиси результату із корекцією порядку.

Приклад 4.5.

Виконаємо додавання двох десяткових чисел $20_{10} = 2{,}0 \cdot 10^{1}$ та $0{,}003_{10} = 3{,}0 \cdot 10^{-3}$ за описаним алгоритмом.

Оскільки порядки операндів – не однакові (1 та –3), потрібно виконати операцію вирівнювання порядків. Оскільки другий операнд має менший порядок, то порядок вирівнюватися буде в ньому. Різниця порядків дорівнює 4, тому кома у другому операнді буде перенесена на 4 розряди ліворуч:

$$3,0\cdot10^{-3}=0,3\cdot10^{-2}=0,03\cdot10^{-1}=0,003\cdot10^{0}=0,0003\cdot10^{1}.$$

Після вирівнювання порядків додамо мантиси і залишимо порядок незмінним:

$$\begin{array}{r} 2,0 \\ +\ 0,0003 \\ \hline 2,0003 \end{array}$$

Оскільки результат вже має нормалізований вигляд, то додаткова нормалізація результату не потрібна.

Відповідь: $2,0003\cdot10^{1}$

Приклад 4.6.

Виконаємо віднімання двох двійкових чисел $1.0101\cdot2^{11}$ – $1.1101\cdot2^{10}$ за описаним алгоритмом.

Порядки операндів – різні , тому спочатку вирівняємо порядки. Оскільки другий операнд має менший порядок, то порядок вирівнювати будемо в ньому. Різниця порядків дорівнює 1 ($11_2 = 3_{10}$ мінус $10_2 = 2_{10}$ дорівнює 1), тому кому у другому операнді буде перенесено на 1 розряд ліворуч:

$$1.1101\cdot2^{10}=0.11101\cdot2^{11}.$$

Тепер виконаємо операцію віднімання мантис із збереженням порядку:

$$\begin{array}{r} 1.01010\cdot2^{11} \\ -\ 0.11101\cdot2^{11} \\ \hline 0.01101\cdot2^{11} \end{array}$$

Мантиса результату має ненормалізований вигляд. Це значить, що потрібно ще виконати нормалізацію результату. В цьому прикладі це означає, що потрібно перенести кому на 2 розряди праворуч та зменшити порядок результату на 2:

$$0.01101 \cdot 2^{11} = 0.1101 \cdot 2^{10} = 1.101 \cdot 2^{1}.$$

Можна виконати ці розрахунки в десятковій системі числення та переконатися, що обчислення виконано правильно.

4.2. Завдання

Номер варіанта потрібно обирати за наступною формулою:

$$N = K + M,$$

де N – номер варіанту,

$K = 0$ для груп ДМ-х1 та ДП-х1 та $K = 20$ для груп ДМ-х2 та ДП-х2,

M – номер із залікової книжки.

Виконати додавання (завдання 1 та 2) та віднімання (завдання 3 та 4) пар чисел, наведених в таблиці нижче за алгоритмом додавання та віднімання дійсних чисел в нормалізованому форматі.

Номер варіанта	Завдання 1	Завдання 2	Завдання 3	Завдання 4
1	0,75 та 5,375	3,25 та 1,5	2,25 та 1,375	4,5 та 3
2	1 та 2,75	2,5 та 10	6,625 та 3,75	7,5 та 4,125
3	3,125 та 7	0,25 та 8,625	7,25 та 4,5	5,25 та 0,375

Номер варіанта	Завдання 1	Завдання 2	Завдання 3	Завдання 4
4	5,125 та 1,375	4,875 та 7,5	10 та 0,875	6,25 та 3,625
5	5,75 та 7,625	1,625 та 9	5 та 0,5	1,75 та 1,625
6	0,125 та 7,875	5 та 7,25	7,5 та 3,25	9,625 та 3,75
7	4,375 та 6	1,875 та 8,375	6,5 та 4,125	5,625 та 0,75
8	2,125 та 8	6,5 та 0,375	9,625 та 3	10 та 5,125
9	0,625 та 7,5	8 та 2,75	7 та 3,625	9,5 та 0,625
10	7,25 та 2,375	5,125 та 3	6,25 та 1	9,75 та 4,25
11	7,75 та 1,625	3,625 та 9,375	8 та 0,75	6,625 та 3,25
12	3,25 та 5,875	6,25 та 1,25	7,625 та 5,5	8,625 та 2,25
13	8,125 та 0, 25	4,5 та 6,625	9,375 та 1,75	8,875 та 4,75
14	2,5 та 4	3,75 та 6,125	9,5 та 5,125	6,5 та 0,5
15	3,625 та 7,125	5,875 та 0,125	9,75 та 2,75	9,375 та 5
16	1,875 та 6,5	4,375 та 7	5,625 та 1,25	8,25 та 4,625
17	3,375 та 7,375	1,5 та 6	5,75 та 3,875	9 та 0,375
18	2 та 4,875	1 та 5,375	8,75 та 4,25	9,125 та 3,125
19	4,75 та 6,875	7,625 та 2,375	8,875 та 5,375	6 та 1,375
20	6,125 та 3	1,375 та 8,875	6,125 та 2,5	8,5 та 4,875
21	1,25 та 6,375	7,875 та 3,375	7 та 1,625	9,25 та 4,375
22	3,5 та 6,625	8,125 та 2,125	8 та 5,25	5,5 та 0,875
23	2,25 та 8,375	5,25 та 7,125	8,25 та 3,375	7,25 та 1,25
24	3,75 та 9	9,625 та 0,375	5,875 та 4,625	8,375 та 2,75
25	2,625 та 6,25	4 та 6,875	7,875 та 1,875	8,125 та 0,625
26	4,5 та 8,875	3 та 7,375	6 та 0,125	6,125 та 4
27	4,25 та 5	7,75 та 1,125	9,125 та 3,125	7,375та 1,5
28	6,75 та 5,25	0,5 та 5,625	8,375 та 5	8 та 3,875
29	0,875 та 8,625	2,625 та 1	8,125 та 2,375	7,875 та 3,375
30	6,375 та 8,5	4,625 та 2	9,25 та 4,875	5,75 та 1

Номер варіанта	Завдання 1	Завдання 2	Завдання 3	Завдання 4
31	5,625 та 9,75	8,75 та 0,75	6,875 та 3	7 та 2,125
32	1,125 та 9,625	2,875 та 9,125	7,125 та 4,75	7,625 та 2,5
33	0,375 та 10	3,125 та 8,5	9 та 1,5	6,375 та 2,875
34	1,75 та 8,75	3,5 та 8,25	8,625 та 1,125	6,75 та 1,875
35	5,5 та 9,375	6,75 та 2,25	8,5 та 0, 25	5,875 та 3,5
36	4,625 та 9,25	0,625 та 9,75	7,375 та 2,875	7,75 та 3
37	2,875 та 9,5	0,125 та 4,25	8,75 та 2,625	2 та 0,125
38	3,875 та 9,125	3,25 та 0,875	6,375 та 3,5	7,125 та 2,375
39	7,75та 0,5	4,75 та 9,5	7,75 та 2,125	6 та 2,625
40	0,25 та 3,125	1,75 та 9,25	6,75 та 4	6,875 та 1,125
41	1,5 та 8,25	3,875 та 0,125	4,375 та 2	5,375 та 0,25

4.3. Контрольні питання

1. Що таке нормалізоване представлення чисел?
2. Що потрібно збільшити – мантису чи порядок для того, щоб збільшити точність представлення числа?
3. Що потрібно збільшити – мантису чи порядок для того, щоб можна було записати ширший діапазон від найменшого до найбільшого числа?
4. Що таке вирівнювання порядків та коли виконується ця операція?
5. Скільки бітів відводиться для зберігання знаку числа?

5. Машинна логіка

Мета роботи: навчитися виконувати логічні операції над булевими величинами.

5.1. Теоретичні відомості

Алгебра – це наука, яка вивчає множини деяких елементів та дії над ними.

Алгебра логіки – це розділ математичної логіки, в якому вивчаються логічні операції над висловлюваннями, тобто методи встановлення істинності чи хибності складних логічних висловлювань за допомогою алгебраїчних методів.

В алгебрі логіки складне логічне висловлювання описується функцією, результатом обчислення якої може бути значення або 1 (*true*), або 0 (*false*). При цьому аргументи функції також можуть мати тільки два значення: 0, або 1.

Операції в алгебрі логіки продумані так , щоб її можна було використовувати в логічних міркуваннях.

Цей розділ математики виник в XIX столітті завдяки зусиллям англійського математика Джорджа Буля. Діапазон наукових інтересів Буля був дуже широкий: рівною мірою його цікавили і математика, і логіка – наука про закони і форми мислення. В ті часи логіка вважалася гуманітарною наукою, але вченому захотілося зробити науку про закони і форми мислення такою ж строгою, як і математика та фізика. Для цього Буль став позначати буквами не числа, як це робиться в звичайній алгебрі, а висловлювання і показав, що такими рівняннями, дуже схожими з алгебраїчними, можна розв’язувати питання про істинність та хибність висловлювань, зроблених людиною. Так виникла алгебра логіки (яку ще називають булева алгебра на честь самого Джоржа Буля). У двох роботах, "*The mathematical analysis of logic*" ("Математичний аналіз логіки") 1847 року та "*An Investigation of the Laws of Thought, on Which Are Founded the Mathematical Theories of Logic and Probabilities*" ("Дослідження законів

мислення, на яких засновані математичні теорії логіки і ймовірностей") 1854 року Джордж Буль заклав основи цієї алгебри. Спочатку булева алгебра не мала ніякого практичного застосування. Однак уже в 20 столітті положення булевої алгебри знайшли застосування в описі функціонування та розробленні різних електронних схем. В 30-х роках минулого сторіччя, працюючи над дисертацією, Клод Шеннон дійшов до висновку, що булеву алгебру можна з успіхом використовувати для аналізу та синтезу електричних схем на реле та перемикачах. Він також показав, що можна зібрати релейні схеми, що виконують логічні операції І, АБО, НІ та схему, яка може реалізувати порівняння. У 1937 році Шенноном написана дисертація під назвою "Символічний аналіз релейних схем та схем на перемикачах". Робота Шеннона мала дуже важливе значення: закони та апарат алгебри логіки стали використовуватися у процесі проектування різних частин обчислювальних машин.

Логічні операції зручно описувати за допомогою ***таблиць істинності***, в яких відображають результати обчислень складних висловлювань за усіх можливих комбінацій значень вхідних величин.

В таблиці 5.1 наведено приклад таблиці істинності. Останній стовпчик містить результат обчислення логічної функції ~ (A & B), а усі інші стовпчики (в нашому прикладі 2) – містять значення вхідних величин (A та B), які приймають участь у формування логічної функції. Причому кількість рядків повинна бути такою, щоби вмістити усі можливі комбінації вхідних величин.

Таблиця 5.1. Приклад таблиці істинності.

A	B	~ (A & B)
false (0)	false (0)	true (1)
false (0)	true (1)	true (1)
true (1)	false (0)	true (1)
true (1)	true (1)	false (0)

Якщо у склад логічної функції входить N змінних, то таблиця істинності буде мати 2^N рядків, оскільки кожна із вхідних величин може приймати два значення – 0 чи 1, тому кількість можливих комбінацій буде дорівнювати 2^N.

За N вхідних змінних кількість стовпчиків в таблиці істинності буде дорівнювати $N+1$ – N стовпчиків із значеннями вхідних величин і останній стовпчик із значеннями логічної функції.

В нашому прикладі, наведеному в таблиці 5.1, було 2 вхідні змінні, тому в таблиці істинності було 2+1=3 стовпчики, та 2^2=4 рядки.

У комп'ютерах булеві змінні представляються (тобто кодуються) бітами, де 1 зазвичай означає істину, а 0 – хибну величину.

Основою цифрової техніки служать кілька ***основних логічних операцій***:

1. Інверсія (логічне заперечення, операція НЕ);
2. Диз'юнкція (логічне додавання, операція АБО);
3. Кон'юнкція (логічне множення, операція І).

Операції І, АБО, НЕ (точніше, пари функцій І та НЕ, АБО та НЕ) утворюють ***повну систему логічних операцій – базис***, з якого можна побудувати скільки завгодно складний логічний вираз.

В обчислювальній техніці також часто використовується ще одна логічна операція – це виключне АБО (XOR) або функція нерівнозначності.

5.1.1. Інверсія

Операцією заперечення (інверсії) логічної величини А називається висловлювання, яке істинне тоді, коли А – хибне, і хибне тоді, коли А - істинне.

Заперечення – це унарна операція, тобто операція, яка виконується тільки для одного операнда. Позначення в електроніці елементів, які виконують логічну функцію інверсії електронних схемах та графічна інтерпретація логічної функції інверсії наведені на рис. 5.1. Часто операція інверсії комбінуються із іншим

логічним виразом, на умовних знаках та електронних схемах це позначають кружечком біля відповідного провідника.

а б

Рис. 5.1. Позначення в електроніці елементів, які виконують логічну функцію інверсії сигналів електронних схемах (а) та графічна інтерпретація логічної функції інверсії (б).

В літературі логічну операцію інверсії зазвичай позначають наступним чином:

$$\neg A \quad \overline{A} \quad \widetilde{A} \quad !A \quad \sim A \quad |NOT\ A$$

.

Оскільки використовується один операнд, то таблиця істинності (табл. 5.2) буде містити 1+1=2 стовпчики, та 2+1 = 3 рядки.

Таблиця 5.2. Таблиця істинності логічної функції інверсії.

A	~*A*
0	1
1	0

В цифровій техніці інформація, зазвичай, зберігається в двійковому форматі. Кожне число займає один чи декілька байтів. До таких чисел, які

складаються із певної кількості бітів, можна застосувати побітову операцію інверсії.

Приклад 5.1.

Виконати побітове заперечення (побітову операцію інверсії) до 8-розрядної двійкової величини $x = 00110110_2$.

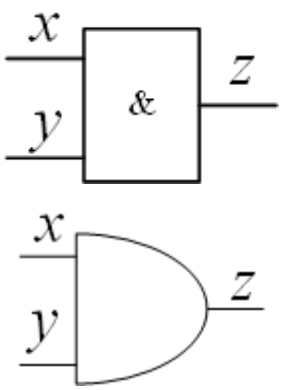
У разі виконання побітової операції, кожен біт числа замінюється на протилежний, тобто 0 стає 1 і 1 стає 0:

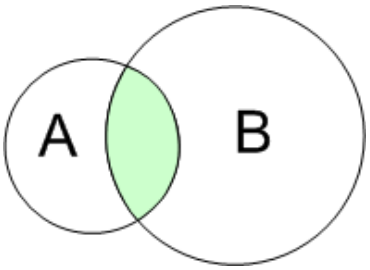
$x =$	0	0	1	1	0	1	1	0
$\sim x =$ ($\overline{x} =$)	1	1	0	0	1	0	0	1

Відповідь: $\sim x = 11001001_2$

5.1.2. Кон'юнкція

Кон'юнкція (її ще називають логічне множення, перетин, логічне І, AND) двох логічних висловлювань А та В є висловлювання, яке істинне тільки тоді, коли істинні обидва вхідних висловлювання. В протилежному випадку – результат буде хибний.





а б

Рис. 5.2. Позначення в електроніці елементів, які виконують логічну функцію кон'юнкції в електронних схемах (а) та графічна інтерпретація цієї логічної функції (б).

В літературі логічну операцію кон'юнкції зазвичай позначають так:

$$A\&B \quad A\wedge B \quad A\cdot B \quad AB \quad A\cap B .$$

У якості операндів дана логічна функція приймає щонайменше дві величини. Тому таблиця істинності містить 2+1=3 стовпчики та 2^2=4 рядки.

В таблиці істинності А та В істинні в останньому рядку, тому тільки в цьому рядку результат буде істина. Усі інші комбінації А та В дають хибний результат.

Таблиця 5.3. Таблиця істинності логічної функції кон'юнкції.

A	B	A & B
0	0	0
0	1	0
1	0	0
1	1	1

Так само операцію кон’юнкції можна поширити на більше, ніж два аргументи. Також для багаторозрядних операндів можна увести операцію побітової (порозрядної) кон’юнкції.

Приклад 5.2.

Виконати побітову операцію логічного І (AND) для двох 8-розрядних двійкових величин: $x = 00110110_2$ та $y = 01010101_2$.

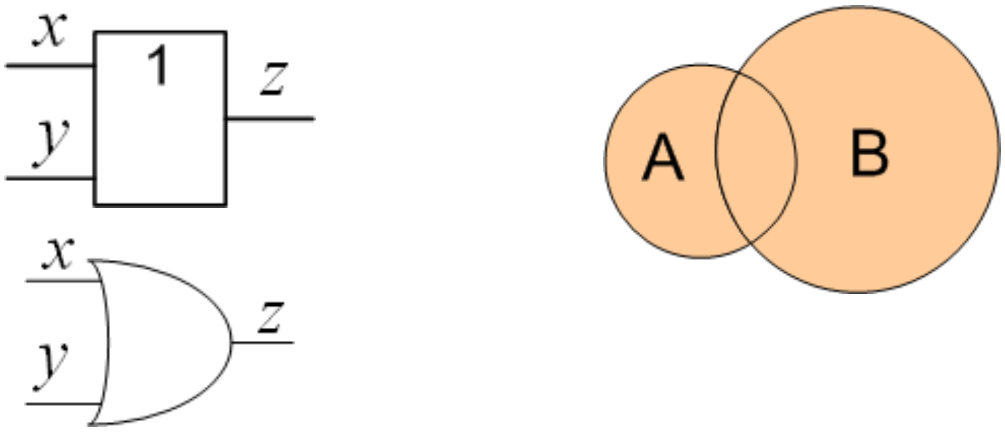
Для виконання побітової операції логічного І розглядаються відповідні біти кожного із операндів і дія над ними виконується попарно. Результуючий біт відповідає значенню із таблиці істинності цієї логічної операції (див. табл. 5.3).

$x =$	0	0	1	1	0	1	1	0
$y =$	0	1	0	1	0	1	0	1
$x \& y =$	0	0	0	1	0	1	0	0

Відповідь: $x \& y = 00010100_2$.

5.1.3. Диз'юнкція

Диз'юнкція (логічне додавання, об’єднання, логічне АБО) двох логічних висловлювань А та В є нове висловлювання, яке істинне, якщо істинне хоча б одне з вхідних висловлювань.



а б

Рис. 5.3. Позначення в електроніці елементів, які виконують логічну функцію диз'юнкції в електронних схемах (а) та графічна інтерпретація цієї логічної функції (б).

В літературі логічну операцію диз'юнкції зазвичай позначають наступним чином:

$$A \mid B \qquad A \vee B \qquad A \cup B$$

Як і для попередньої логічної функції, диз'юнкцією оперує щонайменше із двома вхідними величинами, тобто таблиця істинності містить 3 стовпчики та 4 рядки.

Результат хибний тільки в першому рядку, тому що обидві вхідні величини– хибні. Усі інші комбінації вхідних величин дають істину.

Таблиця 5.4. Таблиця істинності логічної функції диз'юнкції.

A	B	A | B
0	0	0
0	1	1

1	0	1
1	1	1

Можна також увести поняття операції диз'юнкції для трьох і більше аргументів, а для багаторозрядних аргументів – операцію порозрядної диз'юнкції.

Приклад 5.3.

Виконати побітову операцію логічного АБО (OR) для двох 8-розрядних двійкових величин: $x = 00110110_2$ та $y = 01010101_2$.

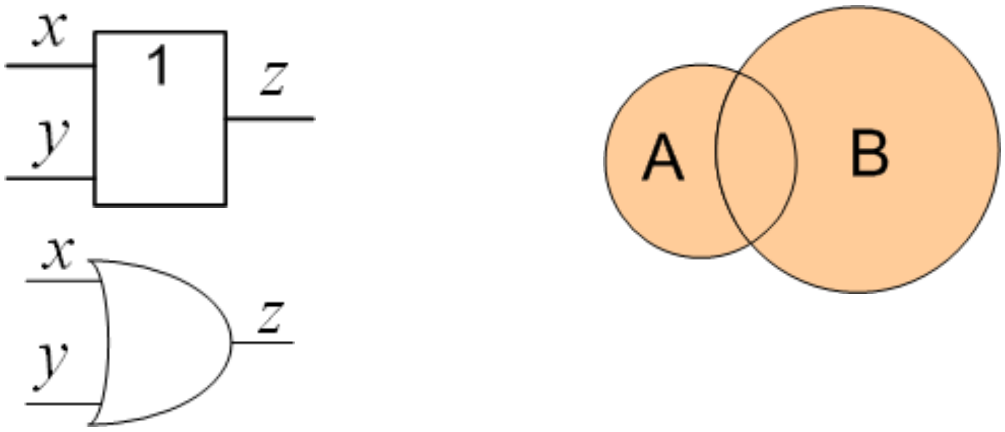
У разі виконання побітової операції логічного АБО розглядається кожна пара відповідних бітів операндів і на потрібну позицію результату записуємо значення із таблиці істинності (див. табл. 5.4).

$x =$	0	0	1	1	0	1	1	0
$y =$	0	1	0	1	0	1	0	1
$x \mid y =$	0	1	1	1	0	1	1	1

Відповідь: $x \mid y = 01110111_2$.

5.1.4. Виключне АБО

Результатом операції виключного АБО двох логічних висловлювань A та B є нове висловлювання, яке істинне, тільки якщо операнди мають різну істинність, тому ця функція також має назву функції нерівнозначності. Інша назва – додавання за модулем 2.



а б

Рис. 5.4. Позначення в електроніці елементів, які виконують логічну функцію виключного АБО в електронних схемах (а) та графічна інтерпретація цієї логічної функції (б).

В літературі логічну операцію виключного АБО зазвичай позначають так :

$$A \oplus B \qquad A + B \qquad A +_2 B$$

Виключне АБО оперує із щонайменше двома вхідними величинами, тобто таблиця істинності містить 3 стовпчики та 4 рядки.

В таблиці істинності результат буде істинним у другому та третьому рядках , оскільки ми маємо комбінації вхідних величин 0-1 та 1-0. У інших рядках результат буде хибний.

Таблиця 5.5. Таблиця істинності логічної функції виключного АБО.

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Приклад 5.4.

Виконати побітову операцію виключного АБО (XOR) для двох 8-розрядних двійкових величин: $x = 00110110_2$ та $y = 01010101_2$.

У разі виконання побітової операції виключного АБО розглядається кожна пара відповідних бітів операндів і на потрібну позицію результату записуємо значення із таблиці істинності (див. табл. 5.5).

$x =$	0	0	1	1	0	1	1	0
$y =$	0	1	0	1	0	1	0	1
$x \mid y =$	0	1	1	0	0	0	1	1

Відповідь: $x \mid y = 01100011_2$.

5.1.5. Пріоритет виконання логічних функцій

За допомогою логічних операцій та простих логічних висловлювань (операндів) можна побудувати логічні вирази, які називають булевими функціями. Оскільки в складних виразах може бути більше одної логічної операції, потрібно знати які логічні операції потрібно виконувати в першу чергу. Ця черговість операцій визначається пріоритетом операцій. Згідно пріоритету

1. Першими виконуються операції в дужках;
2. Заперечення (NOT);
3. Кон'юнкція (AND) та диз'юнкція (OR).

У будь-якому фрагменті формули зміна порядку двох обчислень які йдуть підряд і мають однаковий пріоритет не впливає на результат.

Приклад 5.5.

Визначити пріоритет виконання операцій в наступному логічному виразі:

$$C = \left(\left(\overline{A} \,\&\, B \right) \mid B \right) \oplus A .$$

Згідно раніш зазначених правил, найвищій пріоритет мають операції в дужках. Оскільки ми маємо дві пари дужок, то першими виконуються ті що знаходяться всередині. Розглянемо вираз в цих дужках: $\bar{A}\,\&\,B$. Цей вираз також містить дві операції – інверсію (заперечення) та логічне І (кон'юнкцію). Серед цих логічних операцій вищий пріоритет має саме заперечення, тому першою потрібно виконувати операцію інверсії:

1) $\bar{A}$;

2) $\bar{A}\,\&\,B$.

Далі виконується операція, яка знаходиться в зовнішніх дужках – логічна операція АБО (диз'юнкція):

3) $\left(\bar{A}\,\&\,B\right)|\,B$.

І останньою виконується операція виключного АБО:

4) $\left(\left(\bar{A}\,\&\,B\right)|\,B\right)\oplus A$.

Після виконання усіх логічних операцій виконується присвоювання змінній С результату виконання серії раніш зазначених логічних операцій:

5) $C=\left(\left(\bar{A}\,\&\,B\right)|\,B\right)\oplus A$.

5.1.6. Основні закони алгебри логіки

Інколи складні логічні висловлювання можна суттєво спростити. Для цього можна скористатися основними законами алгебри логіки. Розглянемо основні із них.

1. Закон подвійного заперечення: $\bar{\bar{A}}=A$,

   Закон подвійного заперечення означає, що якщо двічі виконати заперечення із логічною величиною, то ми отримаємо таке саме значення. Дійсно якщо А дорівнює 1 , то НЕ А буде 0, а НЕ-НЕ-А знову 1, тобто А. При А рівному 0, НЕ-А буде 1 і НЕ-НЕ-А знову ж таки 0.

2. Комутативний закон:
   i. для кон'юнкції: $A \& B = B \& A.$
   ii. для диз'юнкції: $A | B = B | A$
   iii. для виключного АБО: $A \oplus B = B \oplus A$

   Комутативний закон, так само як і у звичайній математиці, дозволяє міняти місцями висловлювання, зв'язані логічними операціями І (кон'юнкція) та АБО (диз'юнкція).

3. Асоціативний закон:
   i. для кон'юнкції: $(A \& B) \& C = A \& (B \& C)$
   ii. для диз'юнкції: $(A | B) | C = A | (B | C)$

   Закон асоціативності, який дозволяє по-різному поєднувати висловлювання, з'єднані з допомогою логічних операцій І та АБО.

4. Закон ідемпотентності:
   i. для кон'юнкції: $A \& A = A$
   ii. для диз'юнкції: $A | A = A$
   iii. для виключного АБО: $A \oplus A = 0$

   Ці закони стверджують, що в алгебрі логіки немає показників степенів і коефіцієнтів. Кон'юнкція однакових операндів рівнозначна одному з них; диз'юнкція однакових операндів так само рівнозначна одному із них:

5. Закон одиниці для кон'юнкції: $A \& 1 = A.$
6. Закон нуля для кон'юнкції: $A \& 0 = 0$
7. Закон одиниці для диз'юнкції: $A | 1 = 1$
8. Закон нуля для диз'юнкції: $A | 0 = A$

Останні чотири правила (в цьому списку вони записані під номерами 5, 6, 7 та 8) інколи ще називають законом поглинання констант. Ці закони стверджують, що хибне твердження не впливає на значення логічного виразу при диз'юнкції, а істинне – при кон'юнкції:

9. Закон виключеного третього: $A | \overline{A} = 1$.

   Закон виключеного третього стверджує, що для кожного висловлювання є лише дві можливості: це висловлювання або істинне, або хибне; третього не дано: Наприклад, «Сьогодні я або з'їм апельсин, або не з'їм його». В будь якому випадку, це твердження буде істинним. Тобто істиною завжди буде будь-яке твердження АБО його заперечення.

10. Закон суперечності: $A \& \overline{A} = 0$.

    Закон суперечності стверджує, що ніяке твердження і його заперечення не можуть бути одночасно істинними ніколи. Наприклад, твердження «Йде дощ» І «Не йде дощ» не може бути істинним.

Приклад 5.6.

Виконати над парою чисел (у двійковій системі, восьмибітний формат) такі побітові логічні операції:

1. операцію побітового логічного І (AND),
2. операцію побітового логічного АБО (OR),
3. операцію додавання виключного АБО (XOR)

Числа для обчислень: 43_{10} та 112_{10}.

В якості операндів задано два десяткових числа. Але, як вже зрозуміло, у десятковому вигляді ми не можемо виконати задані логічні операції. Слід

перевести ці десяткові числа до двійкової системи, причому для зберігання кожного із них буде використано 8 двійкових розрядів – бітів. Якщо після переходу в двійкову систему числення число має менше розрядів, то старші розряди доповнюється зліва нулями до потрібної розрядності.

В нашому прикладі в якості аргументів використовуються два десяткових числа: 43 та 112.

Число 43 в двійковій системі числення має 6 розрядів:

$$43_{10} = 101011_2,$$

тобто до потрібної розрядності потрібно додати 2 нулі зліва:

$$43_{10} = 101011_2 = 00101011_2.$$

Число 112 в двійковій системі числення має 7 розрядів:

$$112_{10} = 1110000_2,$$

тобто до потрібної розрядності потрібно додати 1 нуль зліва:

$$112_{10} = 1110000_2 = 01110000_2.$$

Для зручності виконання логічних операцій краще записати розряди операндів із однаковими номерами один під одним, і далі, виконувати відповідні логічні операції із цими парами бітів.

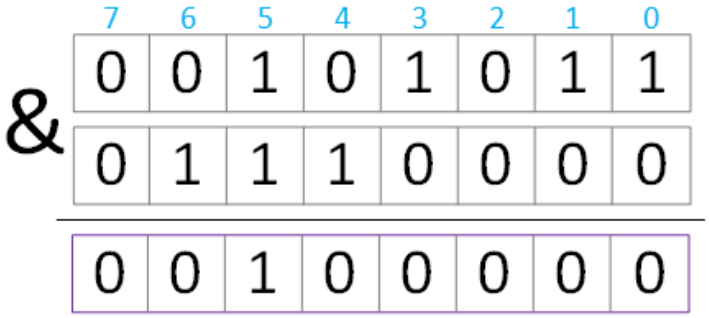


Рис. 5.5. Операція логічного І із числами 43 та 112.

На рис. 5.5 зображено виконання логічної операції І. Спочатку розглядаються біти із номером нуль обох операндів та виконуємо із ними операцію логічного І. Оскільки їх значення дорівнює 1 та 0 відповідно, то згідно із таблицею істинності біт результату дорівнюватиме 0. Пара бітів із номером розряду 1 будуть 1 та 0 і знову отримаємо 0. І так далі із усіма бітами до сьомого включно. Оскільки комбінація відповідних бітів операндів 1 та 1 зустрічається тільки на позиції 5, то в результаті одиниця зустрічається тільки на цій позиції, а усі інші біти результату дорівнюють нулю.

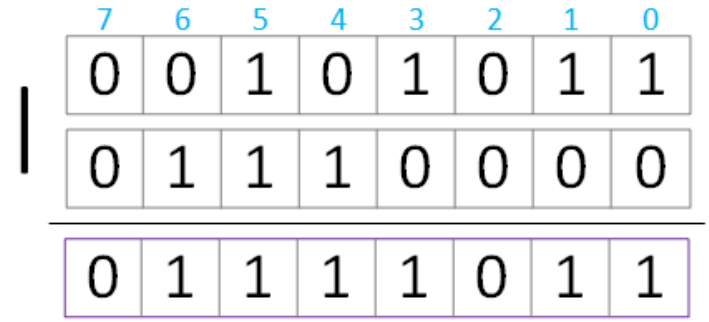


Рис. 5.6. Операція логічного АБО із числами 43 та 112.

Аналогічно виконується операція логічного АБО – це рис. 5.6. Біти номер 0 – це 1 та нуль, згідно із таблицею істинності дають 1. Розряд номер 1 – це так само 1 та 0 дають 1 і так далі до останнього розряду номер 7.

Оскільки комбінація двох нулів між пар відповідних бітів операндів зустрічаються тільки в розрядах номер 2 та 7, то і в результаті нулі зустрічаються тільки в розрядах 2 та 7, а в усіх інших розрядах має бути 1.

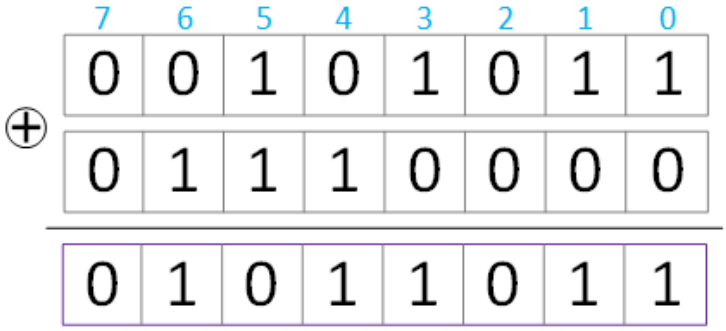


Рис. 5.7. Операція виключного АБО із числами 43 та 112.

На нижньому рис. 5.7 зображено виконання логічної операції виключного АБО. Оскільки біти операндів із однаковими значеннями зустрічаються тільки на позиціях 2 (це нулі), 5 (це одиниці) та 7 (знову нулі), то на цих позиціях результату потрібно записати нулі, а на усіх інших позиціях – одиниці.

5.2. Завдання

Номер варіанта потрібно обирати за наступною формулою:

$$N = K + M ,$$

де N – номер варіанту,

$K = 0$ для груп ДМ-х1 та $K = 20$ для груп ДМ-х2,

M – номер із залікової книжки.

Виконати з парами чисел (у двійковій системі, восьмибітний формат) такі побітові логічні операції:

1. операцію побітового логічного І (AND),
2. операцію побітового логічного АБО (OR),
3. операцію додавання за виключного АБО (XOR).

Номер варіанта	Завдання 1	Завдання 2	Завдання 3	Завдання 4
1	255 та 24	15 та 199	33 та 127	229 та 13
2	161 та 62	98 та 4	77 та 126	69 та 221
3	223 та 97	198 та 25	134 та 125	253 та 23
4	179 та 34	75 та 222	242 та 1	252 та 96
5	124 та 250	232 та 61	224 та 95	12 та 175
6	94 та 251	29 та 123	228 та 22	159 та 26
7	27 та 248	177 та 2	5 та 122	220 та 11
8	142 та 8	140 та 160	21 та 197	119 та 68
9	32 та 178	25 та 241	121 та 6	158 та 28
10	182 та 33	162 та 100	51 та 176	21 та 227

Номер варіанта	Завдання 1	Завдання 2	Завдання 3	Завдання 4
11	156 та 59	3 та 120	20 та 254	225 та 92
12	255 та 93	183 та 30	27 та 157	60 та 230
13	240 та 22	226 та 74	58 та 141	10 та 119
14	32 та 118	91 та 254	29 та 196	155 та 19
15	195 та 13	73 та 227	28 та 117	140 та 15
16	116 та 244	247 та 18	228 та 90	180 та 30
17	153 та 56	50 та 194	253 та 115	229 та 154
18	78 та 250	31 та 152	23 та 181	193 та 17
19	231 та 20	17 та 72	15 та 139	230 та 89
20	249 та 77	55 та 184	144 та 16	57 та 239
21	246 та 14	250 та 88	54 та 138	231 та 114
22	87 та 243	143 та 16	32 та 251	67 та 248
23	225 та 13	71 та 246	137 та 52	232 та 86
24	76 та 244	238 та 34	113 та 7	185 та 9
25	136 та 17	53 та 112	11 та 145	245 та 66
26	167 та 4	33 та 226	35 та 233	164 та 6
27	243 та 85	169 та 12	165 та 14	8 та 110
28	146 та 13	84 та 242	5 та 135	224 та 3
29	18 та 168	252 та 36	40 та 166	233 та 109
30	134 та 10	41 та 70	83 та 241	39 та 174
31	237 та 7	12 та 170	64 та 186	108 та 234
32	107 та 38	133 та 19	1 та 106	163 та 245
33	239 та 82	2 та 223	37 та 172	105 та 24
34	171 та 42	65 та 104	9 та 132	46 та 236
35	81 та 238	8 та 151	26 та 221	11 та 103
36	249 та 10	131 та 20	80 та 237	3 та 187
37	236 та 99	4 та 192	150 та 63	235 та 79

Номер варіанта	Завдання 1	Завдання 2	Завдання 3	Завдання 4
38	188 та 48	1 та 130	244 та 5	173 та 43
39	102 та 31	45 та 222	25 та 147	190 та 47
40	148 та 9	101 та 234	129 та 6	44 та 251
41	7 та 243	128 та 49	8 та 189	100 та 149

5.3. Контрольні питання

1. Хто вважається засновником алгебри логіки?
2. Назвіть унарну логічну операцію. Назвіть які ви знаєте бінарні логічні операції. Чим відрізняються унарні та бінарні операції?
3. Що таке таблиця істинності і для чого вона використовується?
4. Скільки рядків буде мати таблиця істинності, якщо в якості вхідних даних використовуються 3 аргументи?
5. Які логічні функції утворюють повну систему логічних операцій, з якої можна побудувати скільки завгодно складний логічний вираз?
6. Який пріоритет виконання логічних функцій?
7. Сформулюйте закон подвійного заперечення.

6. ВИСНОВКИ

У посібнику розглянуто кілька основоположних питань інформатики, а саме розмаїття систем числення та алгоритми переходу від однієї системи числення, до іншої, поширені способи запису чисел у обчислювальних машинах та способи виконання арифметичних операцій над ними. Також розглянуто першооснови алгебри логіки. Детально розв’язано виклику кількість прикладів з тим, щоб продемонструвати застосування наведених правил, алгоритмів як у типових ситуаціях, так і у граничних.

Викладені у навчальному посібнику відомості стануть у нагоді для виконання розрахунків вручну, зокрема при виконанні контрольних робіт.

7. ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Appendix A.2. TMS320C64x DSP Library Programmer's Reference (PDF). Dallas, Texas, USA: Texas Instruments Incorporated. October 2003. SPRU565. https://www.ti.com/lit/ug/spru565b/spru565b.pdf
2. IEEE Standard for Floating-Point Arithmetic, in IEEE Std 754-2019 (Revision of IEEE 754-2008), vol., no., pp.1-84, 22 July 2019, https://doi.org/10.1109/IEEESTD.2019.8766229.

8. Додаток А.

Основні положення стандарту IEEE 754

Багато цифрових пристроїв, які працюють із числами, що мають дробову частину, використовують уніфікований формат зберігання таких чисел, який описаний в спеціальному стандарті – *IEEE Standard for Floating-Point Arithmetic (IEEE 754)*. Першу версію цього стандарту було випущено в 1985 році, вона мала назву *IEEE 754-1985*. Цей стандарт розв'язав проблеми, які існували при поточних реалізаціях представлення чисел із дробовою частиною.

В цьому стандарті було описано:

- формат зберігання чисел, які мають дробову частину, який містив такі складові частини як: мантиса, порядок та знак числа;
- подання нуля (додатне та від'ємне), подання нескінченності (додатне та від'ємне), а також не-числа (англ. *Not-a-Number, NaN*);
- шляхи перетворення чисел під час виконання над ними арифметичних операцій;
- поведінка при нестандартних (виняткових) ситуаціях, таких як переповнення, ділення на нуль та інші;
- алгоритм виконання арифметичних операцій.

В 2008 році було видано нову версію стандарту (*IEEE 754-2008*), яка доповнювала та заміняла попередню версію стандарту. У новий стандарт включені двійкові формати з попереднього стандарту та три нові формати. Відповідно до чинного стандарту, реалізація повинна підтримувати принаймні один із основних форматів. В таблиці 8.1 наведені приклади деяких форматів, , описаних в стандарті IEEE 754 2008.

Таблиця 8.1. Інформація про деякі формати, описані в стандарті IEEE 754-2008.

Формат	***Назва***	***Основа***	***Цифр мантиси***	***Бітів порядку***	***Максимальне значення***	***Примітка***
binary16	Half precision	2	11	5	65504	Не основний
binary32	Single precision	2	24	8	$3.40 \cdot 10^{38}$	Основний
binary64	Double precision	2	53	11	$1.80 \cdot 10^{308}$	Основний
binary128	Quadruple precision	2	113	15	$1.19 \cdot 10^{4932}$	Основний
binary256	Octuple precision	2	237	19	$1.61 \cdot 10^{78913}$	Не основний

8.1.1. Формат половинної точності: Half-precision floating-point format

Розташування полів та їх довжина в форматі із половинною точністю, відповідно до стандарту IEEE 754 2008 наведено на рис. 8.1.

Числа *half-precision binary floating-point* кодують поле порядку з використанням зсуву (*bias*) на 15.

Іншими словами, для отримання справжнього порядку треба від закодованого поля відняти 15 (тобто 01111_2).

За допомогою значень 00000_2 та 11111_2 поля *Порядок* кодують спеціальні випадки: 00000_2 – нулі та 11111_2 – нескінченності.

Мінімальне точне додатне значення дорівнює $2^{-24} \approx 5.96 \cdot 10^{-8}$.

Мінімальне додатне значення = $2^{-14} \approx 6.10 \cdot 10^{-5}$.

Максимальне значення = $(2-2^{-10})\ 2^{15} = 65504$.

Приклади кодування деяких чисел у форматі із половинною точністю, відповідно до стандарту IEEE 754 2008 наведено в табл. 8.2.

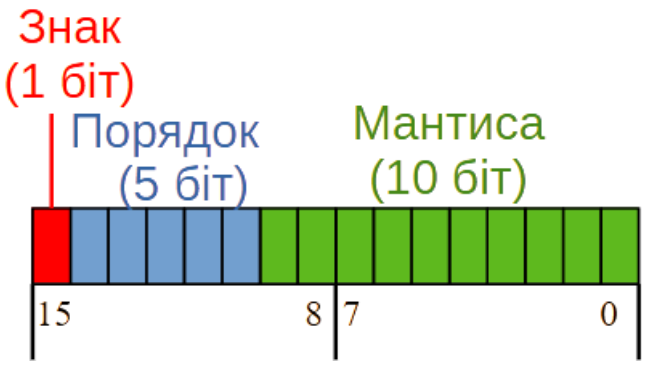


Рис. 8.1. Розташування полів та їх довжина в форматі із половинною точністю, відповідно до стандарту IEEE 754-2008.

Таблиця 8.2. Приклади представлення чисел із половинною точністю (Half-precision) в стандарті IEEE 754-2008

Значення	Формула	Двійкове представлення	Шістнадцяткове представлення
0		0000 0000 0000 0000	0x0000
1	$1\cdot2^0$	0011 1100 0000 0000	0x3C00
–2		1100 0000 0000 0000	0xC000
65504	$1\cdot2^{15}$	0111 1011 1111 1111	0x7BFF
∞		0111 1100 0000 0000	0x7C00
−∞		1111 1100 0000 0000	0xFC00

8.1.2. Формат одинарної точності: Single-precision floating-point format

Розташування полів та їх довжина в форматі із одинарною точністю, відповідно до стандарту IEEE 754 2008 наведена на рис. 8.2.

Для обчислення порядку (показника степеня з восьмирозрядного поля) віднімається *зміщення порядку*, що дорівнює $127_{10} = 7F_{16} = 011111111_2$. Оскільки у нормалізованої двійкової мантиси ціла частина завжди дорівнює одиниці, то у полі мантиси записується лише її дробова частина, тобто фактичний розмір мантиси числа з одинарною точністю становить 24 біти. Для обчислення мантиси до одиниці додається дробова частина мантиси з 23-розрядного поля дробової частини мантиси.

Мінімальне точне додатне значення = $2^{-149} \approx 1.40129846432 \cdot 10^{-45}$.

Мінімальне нормалізоване додатне значення = $2^{-126} \approx 1.17549435082 \cdot 10^{-38}$.

Максимальне значення = $3.40282346639 \cdot 10^{38}$.

Приклади кодування деяких чисел у форматі із одинарною точністю, відповідно до стандарту IEEE 754 2008 наведена в табл. 8.3.

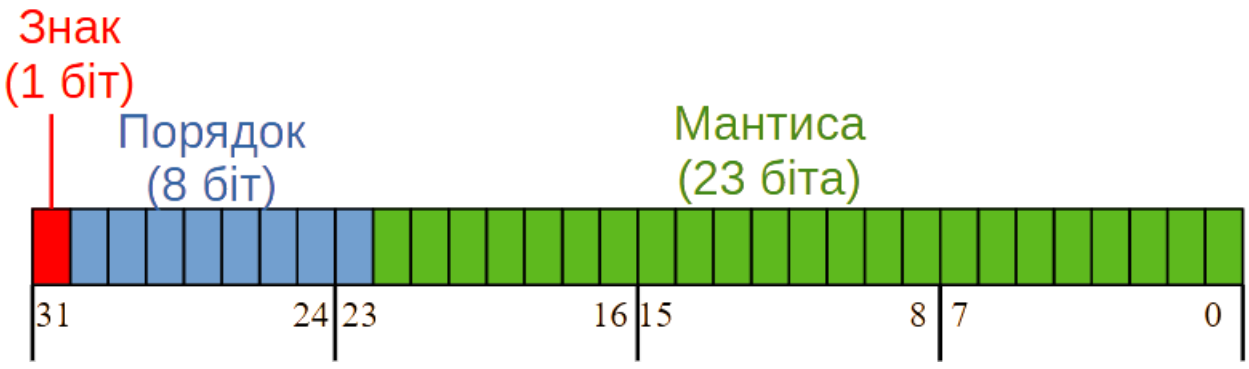


Рис. 8.2. Розташування полів та їх довжина в форматі із одинарною точністю, відповідно до стандарту IEEE 754-2008.

Таблиця 8.3. Приклади представлення чисел із одиничною точністю (Single precision) в стандарті IEEE 754-2008.

Значення	Формула	Двійкове представлення	Шістнадцяткове представлення
2	$1 \cdot 2^{1}$	0100 0000 0000 0000 0000 0000 0000 0000	0x40000000
–2	$-1 \cdot 2^{1}$	1100 0000 0000 0000 0000 0000 0000 0000	0xC0000000
4	$1 \cdot 2^{2}$	0100 0000 1000 0000 0000 0000 0000 0000	0x40800000
6	$1.5 \cdot 2^{2}$	0100 0000 1100 0000 0000 0000 0000 0000	0x40C00000
1	$1 \cdot 2^{0}$	0011 1111 1000 0000 0000 0000 0000 0000	0x3F800000
0.75	$1.5 \cdot 2^{-1}$	0011 1111 0100 0000 0000 0000 0000 0000	0x3F400000
2.5	$1.25 \cdot 2^{1}$	0100 0000 0010 0000 0000 0000 0000 0000	0x40200000
0.1	$1.6 \cdot 2^{-4}$	0011 1101 1100 1100 1100 1100 1100 1101	0x3DCCCCCD
0	$1 \cdot 2^{-128}$	0000 0000 0000 0000 0000 0000 0000 0000	0x00000000

8.1.3. Формат подвійної точності: Double-precision floating-point format

Розташування полів та їх довжина в форматі із подвійною точністю, відповідно до стандарту IEEE 754 2008 наведено на рис. 8.3.

Формат із подвійною точністю широко використовується в комп’ютерній техніці через його ширший діапазон порівняно з представленням із одинарною точністю. Такий формат представлення чисел підтримуються співпроцесором. У комп'ютерах, які мають 64-розрядні числа з плаваючою комою, більшість чисел зберігаються у подвійній точності, оскільки використання чисел одинарної точності забезпечує майже таку ж продуктивність, але всі обчислення у

процесорі із плаваючою точкою здійснюються у 80-бітному (розширеному, extended) поданні.

Приклади кодування деяких чисел у форматі із подвійною точністю, відповідно до стандарту IEEE 754 2008 наведена в табл. 8.4.

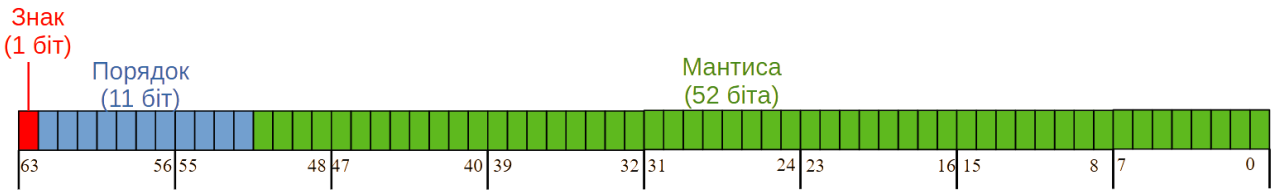


Рис. 8.3. Розташування полів та їх довжина в форматі із подвійною точністю, відповідно до стандарту IEEE 754-2008.

Таблиця 8.4. Приклади представлення чисел із подвійною точністю (Double precision) в стандарті IEEE 754-2008.

Значення	Формула	Шістнадцяткове представлення
1	$1 \cdot 2^1$	0x3FF0 0000 0000 0000
2	$1 \cdot 2^1$	0x4000 0000 0000 0000
-2	$-1 \cdot 2^1$	0xC000 0000 0000 0000
0		0x0000 0000 0000 0000
-0		0x8000 0000 0000 0000
∞		0x7FF0 0000 0000 0000
−∞		0xFFF0 0000 0000 0000
NaN		0x7FFF FFFF FFFF FFFF
1/3		0x3FD5 5555 5555 5555

8.1.4. Формат чотирикратної точності: Octuple-precision floating-point format

Формат із чотирикратною точністю має наступні довжини полів:

- Знак: 1 біт,
- Порядок: 15 біт,
- Мантиса: 112 біт.

Приклади кодування деяких чисел у форматі із чотирикратною точністю, відповідно до стандарту IEEE 754 2008 наведено в табл. 8.5.

Таблиця 8.5. Приклади представлення чисел із чотирикратною точністю (Octuple precision) в стандарті IEEE 754-2008.

Значення	Формула	Шістнадцяткове представлення
1	$1 \cdot 2^1$	0x 3FFF 0000 0000 0000 0000 0000 0000 0000
–2	$-1 \cdot 2^1$	0x C000 0000 0000 0000 0000 0000 0000 0000
0		0x0000 0000 0000 0000 0000 0000 0000 0000
-0		0x8000 0000 0000 0000 0000 0000 0000 0000
∞		0x7FF0 0000 0000 0000 0000 0000 0000 0000
−∞		0xFFF0 0000 0000 0000 0000 0000 0000 0000
1/3		0x3FFD 5555 5555 5555 5555 5555 5555 5555