Міністерство освіти і науки України
ДВНЗ "Прикарпатський національний університет імені Василя Стефаника"

Кафедра комп'ютерної інженерії та електроніки

Терлецький А. І., Фрик О. Б.

СПОСОБИ ПОДАННЯ ЧИСЕЛ У КОМП'ЮТЕРІ

методичні рекомендації до виконання лабораторних робіт
з дисципліни **"Архітектура комп'ютерів"**

для студентів напряму "Комп'ютерна інженерія"

Івано-Франківськ – 2012

УДК 004.222+511.1
ББК 22.13:32.973я73
Т35

Терлецький А. І., Фрик О. Б. Способи подання чисел у комп'ютері. Методичні рекомендації до виконання лабораторних робіт з дисципліни "Архітектура комп'ютерів" для студентів напряму "Комп'ютерна інженерія". – Івано-Франківськ, 2012. – 111 с.

У посібнику розглянуто основні способи подання і кодування чисел, які використовують сучасні обчислювальні машини. Детально проаналізовано міжнародні стандарти, які описують бінарні та десяткові числа з рухомою комою і правила виконання над ними арифметичних дій. Дано ряд методичних вказівок до виконання лабораторних робіт з дисципліни "Архітектура комп'ютерів", які передбачені для студентів напряму підготовки "Комп'ютерна інженерія".

Рецензенти:
професор кафедри радіофізики і електроніки
Прикарпатського національного університету імені Василя Стефаника,
доктор технічних наук **Когут І. Т.**

кандидат педагогічних наук, доцент кафедри статистики і
вищої математики Прикарпатського національного
університету імені Василя Стефаника **Кульчицька Н. В.**

Рекомендовано до друку Вченою радою фізико-технічного факультету Прикарпатського національного університету імені Василя Стефаника (протокол № 6 від 03.05.2012 р.)

ЗМІСТ

ПЕРЕДМОВА

Слово "комп'ютер" у перекладі означає обчислювач. Тобто основне і єдине призначення комп'ютера – обчислювати. Яка б інформація не опрацьовувалась у комп'ютері, вона завжди є не що інше як набір чисел. Зазвичай числа відображають за допомогою знаків або цифр. Загальноприйнятою є двійкова система числення, яка оперує цифрами "0" та "1". Така система числення вибрана з огляду на простоту своєї реалізації за допомогою простих фізичних пристроїв: контакти реле можуть бути замкнені чи розімкнені, електронна лампа, транзистор можуть перебувати у непровідному (закритому) або провідному (відкритому) станах, феритове кільце може мати намагніченість $+\boldsymbol{B}$ або $-\boldsymbol{B}$ тощо.

За допомогою наборів двійкових цифр можна подати не тільки числа, але і будь-якого роду нечислову інформацію, якщо задати спосіб її кодування. Наприклад, текстову, графічну, звукову та іншого роду інформацію в пам'яті комп'ютера подають у вигляді кодів двійкової системи числення.

Однак кодування суто числової інформації також має свої особливості. Числа бувають натуральними, цілими, дробовими (загалом раціональними), ірраціональними (загалом дійсними), скінченними, нескінченними, з різною кількістю значущих цифр, у широкому діапазоні значень, нескінченно великими чи нескінченно малими тощо. Подання (кодування) таких чисел в комп'ютері повинно задовольняти певним вимогам, таким як:

– ефективність використання пам'яті комп'ютера для їх зберігання;
– мінімальний час здійснення типових арифметичних дій над ними;
– забезпечення мінімальної похибки чи (та) необхідного діапазону подання,
– простоту реалізації арифметико-логічних пристроїв.

Існує ряд загальноприйнятих і перспективних способів кодування чисел. Кожен з таких способів кодування має як свої переваги, так і недоліки. Вивчення особливостей подання числової інформації у комп'ютері є основою розуміння принципів його роботи. Ці знання особливо необхідні для вивчення дисципліни "Архітектура комп'ютерів". У зв'язку з тим, що ця дисципліна є базовою для студентів напряму "Комп'ютерна інженерія", значна увага приділена детальному розгляду особливостей кодування чисел та практичному застосуванню їх у сучасних обчислювальних системах.

Розділ 1. ОСНОВНІ КОМП'ЮТЕРНІ СИСТЕМИ ЧИСЛЕННЯ

1.1. Загальні відомості про системи числення

Спосіб зображення (подання) чисел за допомогою умовних знаків називають **системою числення**. Очевидно, найпростіше зобразити число за допомогою паличок, рисок. Скільки рисок – таке ж і число. Вполював мамонта (антилопу, кенгуру) – поставив риску на стіні печери. Такий спосіб подання чисел зберігся і до сьогодні. Саме так інколи ведуть підрахунок певних подій. Як не дивно, але цю систему числення, вірніше її модифікацію, і тепер використовують у комп'ютері (унарна система числення). Однак навіть невеликі числа подавати в такий спосіб не надто зручно, тому для їх запису використовують ряд певних символів (ієрогліфів), які називають цифрами. **Сукупність всіх цифр називають алфавітом системи числення**. Існує два способи подання числа за допомогою цифр. У першому кожній цифрі незалежно від її положення в записі числа присвоюють певне значення, а значення числа є сумою значень всіх його цифр. Такі системи числення називають непозиційними. До них зокрема належать римська та старослов'янська системи числення [1, 2]. Наприклад, у римській системі числення маємо:

Таблиця 1.1.

Число	1	2	3	4	5	6	7	8	9
Цифра	I	II	III	IV	V	VI	VII	VIII	IX
Число	10	20	30	40	50	60	70	80	90
Цифра	X	XX	XXX	XL	L	LX	LXX	LXXX	XC
Число	100	200	300	400	500	600	700	800	900
Цифра	C	CC	CCC	CD	D	DC	DCC	DCCC	CM
Число	1000	2000	3000	3999					
Цифра	M	MM	MMM	MMMCMXCIX					

Тут число 3999 є максимальним. Існує ряд правил запису чисел в цій системі, зокрема насамперед записують тисячі, потім сотні, і т.д, а кожен одинарний символ не може повторюватися більше трьох разів. Крім того, якщо менша цифра стоїть перед більшою, то її віднімають від більшої, а якщо після – то додають. Наприклад, число 1988 матиме запис MCMLXXXVIII. Непозиційні системи числення незручні для математичних обчислень, тому їх не використовують в комп'ютерах.

Більш досконалими системами числення є позиційні. В таких системах значення цифри залежить від позиції її запису в числі. Кожну позицію з присвоєним їй порядковим номером називають **розрядом** числа. Якщо число має n розрядів цілої та m розрядів дробової частин, то старшому розряду цілої частини присвоюють номер n-1, молодшому розряду цілої частини – номер 0, старшому розряду дробової частини – номер -1, а молодшому розряду – номер $-m$. При цьому кожна i-та позиція має певний ваговий коефіцієнт p_i, а значення числа є сума добутків значень цифр на відповідний ваговий коефіцієнт

$$A \equiv a_{n-1}a_{n-2}...a_i...a_1a_0, a_{-1}a_{-2}...a_{-m} = a_{n-1} \cdot p_{n-1} + a_{n-2} \cdot p_{n-2} +$$
$$+ ... + a_i \cdot p_i + ... + a_0 \cdot p_0 + a_{-1} \cdot p_{-1} + ... + a_{-m} \cdot p_{-m} = \sum_{i=-m}^{n-1} a_i \cdot p_i. \quad (1.1)$$

У запису числа позиції з більшими ваговими коефіцієнтами розташовують лівіше, ніж з меншими, а самі вагові коефіцієнти можуть набувати довільного значення. Такі системи числення зазвичай є надлишковими, тобто одне і те ж число можна подати різними способами. Розглянемо, як приклад, систему числення з ваговими коефіцієнтами рівними числам Фібоначчі. Як відомо, кожне наступне число Фібоначчі є сумою двох попередніх, а загалом вони утворюють ряд 1, 1, 2, 3, 5, 8, 13, 21... Якщо обмежитися 8-бітним форматом та цифрами 0 і 1, тоді, наприклад, число 26 можна подати 6-ма способами:

Вагові коефіцієнти	21	13	8	5	3	2	1	1
$26 \equiv$	1	0	0	1	0	0	0	0
$26 \equiv$	1	0	0	0	1	1	0	0
$26 \equiv$	1	0	0	0	1	0	1	1
$26 \equiv$	0	1	1	1	0	0	0	0
$26 \equiv$	0	1	1	0	1	1	0	0
$26 \equiv$	0	1	1	0	1	0	1	1

Надлишковість систем числення часто використовують для передавання даних за умов високих завад, оскільки такі коди легше відновити навіть у випадку збою в одному чи двох розрядах. Якщо в позиційній системі числення всі вагові коефіцієнти взяти однаковими і рівними одиниці, то отримуємо все ту ж унарну систему.

Ще більш досконалими є степеневі позиційні системи числення. Тут вага кожного розряду є відповідним степенем певного числа $\boldsymbol{q}$, яке називають основою числення, а запис числа матиме вигляд

$$A \equiv a_{n-1}a_{n-2}...a_i...a_1a_0, a_{-1}a_{-2}...a_{-m} = a_{n-1}q^{n-1} + a_{n-2}q^{n-2} +$$
$$+ ... + a_iq^i + ... + a_0q^0 + a_{-1}q^{-1} + ... + a_{-m}q^{-m} = \sum_{i=-m}^{n-1} a_iq^i. \quad (1.2)$$

Числа a_i – це цифри, які належать алфавіту системи числення. Зазвичай цифрами є числа натурального ряду від 0 до $\boldsymbol{q}$-1. Основа числення в деяких системах може набувати нецілочисельних, від'ємних та навіть ірраціональних значень. Прикладом такої системи є код золотої p-пропорції, як частковий випадок узагальненої Фібоначчієвої системи числення. Однак надалі розглядатимемо тільки такі системи числення, основа яких – натуральне число $\boldsymbol{q}$, а алфавіт складається із цілих чисел від 0 до $\boldsymbol{q}$-1.

1.2. Унарна та інверсна унарна системи числення

Як вже було сказано вище, унарна система є найстарішою і найпростішою з систем числення. Її алфавіт складається тільки з одного символу "1", а щодо позиційності чи непозиційності цієї системи питання залишається відкритим. З

одного боку значення числа є сукупність значень всіх цифр, тобто система непозиційна, з іншого – це позиційна система з одиничними ваговими коефіцієнтами, або навіть степенева позиційна система з основою $q = 1$. Унарна система числення поєднує в собі такі різні властивості, а до того ж вона є частковим випадком кодів золотої p-пропорції, якщо $p = 0$ [3].

Оскільки інші цифри, крім "1", в запису числа відсутні, то незаповнені місця позначають пропусками. На практиці пропуски часто замінюють нулями, хоча нулі не належать до алфавіту цієї системи числення. В інверсній унарній системі всі одиниці інверсують до пропусків (нулів), а всі пропуски (нулі) – до одиниць. При цьому з усіх одиниць залишають тільки одну – крайню справа. Обидва випадки кодування від 0 до 9 демонструє табл 1.2.

Таблиця 1.2.

Число	Унарна система числення		Інверсна унарна система числення
0	␣␣␣␣␣␣␣␣␣␣	0000000000u	0000000001 $\bar{u}$
1	␣␣␣␣␣␣␣␣␣1	0000000001u	0000000010 $\bar{u}$
2	␣␣␣␣␣␣␣␣11	0000000011u	0000000100 $\bar{u}$
3	␣␣␣␣␣␣␣111	0000000111u	0000001000 $\bar{u}$
4	␣␣␣␣␣␣1111	0000001111u	0000010000 $\bar{u}$
5	␣␣␣␣␣11111	0000011111u	0000100000 $\bar{u}$
6	␣␣␣␣111111	0000111111u	0001000000 $\bar{u}$
7	␣␣␣1111111	0001111111u	0010000000 $\bar{u}$
8	␣␣11111111	0011111111u	0100000000 $\bar{u}$
9	␣111111111	0111111111u	1000000000 $\bar{u}$

Як можна помітити з табл. 1.2, в інверсній унарній системі числення одиницю завжди записують в тій позиції, що відповідає значенню числа. Інверсну унарну систему використовують для опису роботи шифраторів та дешифраторів, тоді як звичайну – для кодування вихідних сигналів деяких типів АЦП (аналогово-цифрових перетворювачів).

1.3. Двійкова, вісімкова та шістнадцяткова системи числення

Насамперед слід домовитися про позначення системи числення чисел, в якій вони записані. Оскільки всі системи числення мають ряд спільних цифр, то за записом числа інколи неможливо визначити систему числення. Існує три способи позначення. В першому основу системи числення записують в дужках у вигляді нижнього індексу, наприклад $258{,}34_{(10)}$ чи $100101_{(2)}$. Такий спосіб запису є історичним, але не надто зручним на практиці. Другий спосіб використовує великі латинські літери назви відповідної системи поруч з числом, наприклад 258,34D (D – decimal) чи 01110B (B – binary). Як буде показано далі, B і D є цифрами шістнадцяткової системи числення. Щоби не плутати будемо позначати систему числення малими латинськими літерами: b – binary (двійкова), q – octal (вісімкова), d – decimal (десяткова), h – hexadecimal (шістнадцяткова). Ряд перших натуральних чисел в цих системах числення матимуть вигляд, як показано в табл. 1.3.

Таблиця 1.3. Натуральні числа в різних основних системах числення.

десят-кова	двій-кова	вісім-кова	шістнад-цяткова	десят-кова	двій-кова	вісім-кова	шістнад-цяткова
0d	00000b	00q	00h	16d	10000b	20q	10h
1d	00001b	01q	01h	17d	10001b	21q	11h
2d	00010b	02q	02h	18d	10010b	22q	12h
3d	00011b	03q	03h	19d	10011b	23q	13h
4d	00100b	04q	04h	20d	10100b	24q	14h
5d	00101b	05q	05h	21d	10101b	25q	15h
6d	00110b	06q	06h	22d	10110b	26q	16h
7d	00111b	07q	07h	23d	10111b	27q	17h
8d	01000b	10q	08h	24d	11000b	30q	18h
9d	01001b	11q	09h	25d	11001b	31q	19h
10d	01010b	12q	0Ah	26d	11010b	32q	1Ah
11d	01011b	13q	0Bh	27d	11011b	33q	1Bh
12d	01100b	14q	0Ch	28d	11100b	34q	1Ch
13d	01101b	15q	0Dh	29d	11101b	35q	1Dh
14d	01110b	16q	0Eh	30d	11110b	36q	1Eh
15d	01111b	17q	0Fh	31d	11111b	37q	1Fh

Загальноприйнята десяткова система числення має основу числення $\boldsymbol{q} = 10$ і алфавіт (цифри) 0, 1, 2, ... 9. Згідно з (1.2) в десятковій системі числення, наприклад число 258,34d матиме запис

$$258{,}34\text{d} = 2\cdot 100 + 5\cdot 10 + 8\cdot 1 + 3\cdot 0{,}1 + 4\cdot 0{,}01 = 2\cdot 10^2 + 5\cdot 10^1 + 8\cdot 10^0 + 3\cdot 10^{-1} + 4\cdot 10^{-2}.$$

Десяткова система числення незручна для використання в ЕОМ, оскільки кожну з цифр необхідно кодувати відповідним рівнем напруги. Схеми з десятьма дискретними рівнями напруги є надто складними, ненадійними та нестійкими до завад. Найбільш завадостійкими є кодування:

1) низький рівень напруги – 0, високий – 1;

2) високий рівень від'ємної напруги – 0, високий рівень додатної – 1

3) високий рівень від'ємної напруги – 0, низький рівень – 1, високий рівень додатної напруги – 2.

Першим двом відповідає двійкова система числення з алфавітом 0 та 1, третьому – трійкова з алфавітом 0, 1 та 2. У зв'язку з тим, що схемна реалізація двох останніх варіантів є значно складніша, ніж у першому випадку, майже всі ЕОМ використовують двійкову систему числення (незважаючи на те, що трійкова система числення більш оптимальна, ніж двійкова).

Згідно з (1.2) число в двійковій системі числення записують як

$$A \equiv a_{n-1}a_{n-2}\ldots a_i\ldots a_1a_0, a_{-1}a_{-2}\ldots a_{-m} = a_{n-1}2^{n-1} + a_{n-2}2^{n-2} +$$

$$+\ldots + a_i 2^i + \ldots + a_0 2^0 + a_{-1}2^{-1} + \ldots + a_{-m}2^{-m} = \sum_{i=-m}^{n-1} a_i 2^i.$$

Однак алфавіт системи числення містить всього дві цифри "0" і "1", тому запис довільного числа, наприклад 1001101,1101b, матиме такий вигляд (тут всі числа подані в двійковій формі відповідно до табл. 1.3.

1001101,1101b=

$$=1b\cdot 10b^{110b}+0b\cdot 10b^{101b}+0b\cdot 10b^{100b}+1b\cdot 10b^{11b}+1b\cdot 10b^{10b}+0b\cdot 10b^{1b}+1b\cdot 10b^{0b}+$$

$$+1b\cdot 10b^{-1b}+1b\cdot 10b^{-10b}+0b\cdot 10b^{-11b}+1b\cdot 10b^{-100b}.$$

Крім двійкової системи числення, яка лежить в основі роботи пристроїв та вузлів ЕОМ, для **спрощення запису двійкового числа** широко застосовують шістнадцяткову та дещо рідше вісімкову системи числення. У вісімковій системі числення використовують вісім відомих нам цифр від 0 до 7, тоді як шістнадцяткова потребує додаткових цифр: A, B, C, D, E, F (див. табл. 1.3).

1.4. Правила виконання арифметичних дій в основних системах числення

Всі арифметичні дії в системі числення з основою q здійснюють за тим же алгоритмом, що і в загальноприйнятій десятковій системі числення, однак для цього використовують таблиці додавання та множення відповідних систем числення. Для складання таблиці додавання в будь-якій системі числення користуються таким правилом: якщо сума цифр $a_i + a_j < q$ то $a_i + a_j = a_k$, де a_k – цифра, що належить алфавіту цієї системи числення; якщо $a_i + a_j \geq q$, то цифра в заданому розряді рівна $a_k = a_i + a_j - q$, але крім того появляється одиниця перенесення в старший розряд тобто $a_i + a_j = 1a_k$.

Таблиці додавання для двійкової, вісімкової та шістнадцяткової систем показані в табл. 1.4-1.6. Додавання в різних системах числення демонструють такі приклади:

перенесення	1 1 1 1	1 1 1 1	1 1 1 1
доданок 1	1 0 0 1,0 1 b	3 5 7,3 4 q	7 A 4,A 9 h
доданок 2	1 1 0 1,1 1 b	4 2 6,2 4 q	F C D,0 7 h
сума	1 0 1 1 1,0 0 b	1 0 0 5,6 0 q	1 7 7 1,B 0 h

Здійснюючи додавання великої кількості доданків в двійковій (чи іншій) системі числення слід особливу увагу звертати на перенесення, яке може бути одно-, дво- та більше розрядне. Наприклад, в двійковій системі числення для отримання суми шести доданків слід ретельно виписувати всі перенесення, які додають до відповідних розрядів за правилами додавання двійкових чисел:

перенесення			100	11	10	11	100	100	10			
доданок 1				1	0	0	0	0,	1	1 b	=	16,75d
доданок 2				1	1	1	0	1,	1	1 b	=	29,75d
доданок 3				1	1	0	1	0,	1	0 b	=	26,50d
доданок 4				1	1	0	0	1,	1	0 b	=	25,50d
доданок 5				1	1	0	1	1,	1	1 b	=	27,75d
доданок 6				1	1	0	1	1,	1	1 b	=	27,75d
сума	1	0	0	1	1	0	1	0,	0	0 b	=	154,00d

Таблиці 1.4-1.6. Додавання в двійковій, вісімковій та шістнадцятковій системах числення

Двійкова

+	0b	1b
0b	0b	1b
1b	1b	10b

Вісімкова

+	0q	1q	2q	3q	4q	5q	6q	7q
0q	0q	1q	2q	3q	4q	5q	6q	7q
1q	1q	2q	3q	4q	5q	6q	7q	10q
2q	2q	3q	4q	5q	6q	7q	10q	11q
3q	3q	4q	5q	6q	7q	10q	11q	12q
4q	4q	5q	6q	7q	10q	11q	12q	13q
5q	5q	6q	7q	10q	11q	12q	13q	14q
6q	6q	7q	10q	11q	12q	13q	14q	15q
7q	7q	10q	11q	12q	13q	14q	15q	16q

Шістнадцяткова

+	0h	1h	2h	3h	4h	5h	6h	7h	8h	9h	Ah	Bh	Ch	Dh	Eh	Fh
0h	0h	1h	2h	3h	4h	5h	6h	7h	8h	9h	Ah	Bh	Ch	Dh	Eh	Fh
1h	1h	2h	3h	4h	5h	6h	7h	8h	9h	Ah	Bh	Ch	Dh	Eh	Fh	10h
2h	2h	3h	4h	5h	6h	7h	8h	9h	Ah	Bh	Ch	Dh	Eh	Fh	10h	11h
3h	3h	4h	5h	6h	7h	8h	9h	Ah	Bh	Ch	Dh	Eh	Fh	10h	11h	12h
4h	4h	5h	6h	7h	8h	9h	Ah	Bh	Ch	Dh	Eh	Fh	10h	11h	12h	13h
5h	5h	6h	7h	8h	9h	Ah	Bh	Ch	Dh	Eh	Fh	10h	11h	12h	13h	14h
6h	6h	7h	8h	9h	Ah	Bh	Ch	Dh	Eh	Fh	10h	11h	12h	13h	14h	15h
7h	7h	8h	9h	Ah	Bh	Ch	Dh	Eh	Fh	10h	11h	12h	13h	14h	15h	16h
8h	8h	9h	Ah	Bh	Ch	Dh	Eh	Fh	10h	11h	12h	13h	14h	15h	16h	17h
9h	9h	Ah	Bh	Ch	Dh	Eh	Fh	10h	11h	12h	13h	14h	15h	16h	17h	18h
Ah	Ah	Bh	Ch	Dh	Eh	Fh	10h	11h	12h	13h	14h	15h	16h	17h	18h	19h
Bh	Bh	Ch	Dh	Eh	Fh	10h	11h	12h	13h	14h	15h	16h	17h	18h	19h	1Ah
Ch	Ch	Dh	Eh	Fh	10h	11h	12h	13h	14h	15h	16h	17h	18h	19h	1Ah	1Bh
Dh	Dh	Eh	Fh	10h	11h	12h	13h	14h	15h	16h	17h	18h	19h	1Ah	1Bh	1Ch
Eh	Eh	Fh	10h	11h	12h	13h	14h	15h	16h	17h	18h	19h	1Ah	1Bh	1Ch	1Dh
Fh	Fh	10h	11h	12h	13h	14h	15h	16h	17h	18h	19h	1Ah	1Bh	1Ch	1Dh	1Eh

Віднімання здійснюють подібним чином. Якщо $a_i \geq a_j$, тоді $a_k = a_i - a_j$, а якщо $a_i < a_j$, то $a_k = \boldsymbol{q} + a_i - a_j$ і необхідно запозичити одиницю зі старшого розряду зменшуваного. Наприклад:

запозичення	1 1 1 1 1	1 1	1 1 1
зменшуване	1 1 0 0 1,0 1 b	5 5 2,3 4 q	F A 4,A 7 h
від'ємник	1 1 0 1,1 1 b	4 2 6,2 6 q	7 C D,0 9 h
різниця	1 0 1 1,1 0 b	1 2 4,0 6 q	7 D 7,9 E h

Множення чисел в позиційних системах здійснюють за алгоритмом, аналогічним алгоритму множення в десятковій системі числення, тобто необхідно помножити множене на кожен розряд множника та знайти суму часткових добутків, зміщених один відносно одного на один розряд. Оскільки

множення чисел в вісімковій та шістнадцятковій системах числення зазвичай здійснюють за допомогою множення в двійковій системі, то необхідно пам'ятати просте правило множення двійкових чисел:

×	0b	1b
0b	0b	0b
1b	0b	1b

Наприклад:

```
      1 0 1 1 1 0 1 1 0 1,0 1 1          (749,375d)
    ×                 1 1,0 1            (3,25d)
      1 0 1 1 1 0 1 1 0 1 0 1 1
+     0 0 0 0 0 0 0 0 0 0 0 0 0
+   1 0 1 1 1 0 1 1 0 1 0 1 1
+ 1 0 1 1 1 0 1 1 0 1 0 1 1
1 0 0 1 1 0 0 0 0 0 1 1,0 1 1 1 1      (2435,46875d)
```

Алгоритм ділення в двійковій системі числення також такий, як і в десятковій системі числення:

x = 430d = 110101110b
y = 10d = 1010b
x/y = 43d =101011b

```
110101110  |1010
1010       |101011
  1101
  1010
    1111
    1010
     1010
     1010
        0
```

x = 204d = 11001100b
y = 12d = 1100b
x/y = 17d = 10001b

```
11001100  |1100
1100      |10001
     1
     0
     11
      0
     110
       0
     1100
     1100
        0
```

1.5. Переведення чисел з однієї системи в іншу. Метод безпосереднього заміщення

Переведення чисел за допомогою цього методу здійснюють в таким спосіб:

1. Задане число в системі числення з основою q відповідно до формули (1.1) подають у вигляді суми з ваговими коефіцієнтами;
2. Усі цифри a_i і основу q в правій частині формули записують в системі числення з новою основою p. Якщо $p < q$, то необхідно знати подання чисел від p до q в системі числення з основою p. Якщо $p > q$, то зображення цифр у p-й системі збігається з їхнім зображенням у q-й системі;

3. Усі арифметичні дії виконують відповідно до правої частини рівняння (1.1) в системі з основою ***p***. Розглянемо, наприклад, переведення 35,25d в двійкову систему числення (***p*** < ***q***):

$$35{,}25\text{d} = 3\cdot 10^{1} + 5\cdot 10^{0} + 2\cdot 10^{-1} + 5\cdot 10^{-2} = 11\text{b}\cdot 1010\text{b} + 101\text{b} + \frac{10\text{b}}{1010\text{b}} + \frac{101\text{b}}{1010\text{b}\cdot 1010\text{b}}.$$

Виконавши необхідні множення, ділення та додавання, отримаємо результат 35,25d = 100011,01b.

Інший приклад (***p*** > ***q***). Перевести число 623,2q в десяткову систему числення:

$$623{,}2\text{q} = 6\cdot 8^{2} + 2\cdot 8^{1} + 3\cdot 8^{0} + 2\cdot 8^{-1} = 384\text{d} + |16\text{d} + 0{,}25\text{d} = 403{,}25\text{d}.$$

Метод безпосереднього заміщення зручний для переведення чисел з будь-якої в ту, в якій найпростіше виконуються арифметичні операції, а саме **в десяткову**. В ЕОМ цей метод використовують для переведення чисел в двійкову систему числення.

1.6. Метод послідовного ділення (множення) на основу

Метод послідовного ділення використовують для переведення із однієї системи в іншу **тільки цілих чисел**. Нехай число **A** записано в системі з основою ***q*** (1.1). Вважаємо, що подання числа **A** в системі з основою ***p*** має вигляд:

$$\text{A} \equiv a_{n-1}a_{n-2}\ldots a_1a_0 = a_{n-1}p^{n-1} + a_{n-2}p^{n-2} + \ldots + a_1p^{1} + \ldots + a_0 \qquad (1.3)$$

Поділимо число **A** на основу ***p***. Оскільки $a_0 < p$, то в результаті ділення отримаємо цілу частину $\text{A}_1 = a_{n-1}p^{n-2} + a_{n-2}p^{n-3} + \ldots + a_2p^{1} + a_1$ та залишок a_0. Отже, залишок від ділення числа **A** на основу ***p*** дорівнює значенню цифри молодшого розряду. Для того, щоб отримати наступний розряд, залишок слід відкинути, а число $\mathbf{A_1}$ знову поділити на ***p*** і визначити залишок a_1. Цей процес необхідно повторити ***n***-1 разів, а з отриманих залишків записати подання числа **A** в системі з основою ***p***. Розглянемо, наприклад, переведення числа 783d в двійкову систему числення.

```
783  |2
782  |391  |2
  1  390  |195  |2
        1  194  |97  |2
              1  96  |48  |2
                    1  48  |24  |2
                          0  24  |12  |2
                                0  12  |6  |2
                                      0  6  |3  |2
                                            0  3  |1
                                               1
1   1    1    1   0   0   0  0  1  1  – залишки
<-----------------------------------------
```

Результат записують у зворотному порядку, а саме:

783d = 1100001111b.

Інший приклад: перевести число 783d у вісімкову систему числення.

```
783  |8
776  |97  |8
  7   96  |12  |8
        1   8  |1
               4
  7    1    4   1   – залишки
  <--------------
```

Результат 783d = 1417q.

Для **переведення дробових чисел** з однієї системи в іншу використовують **метод послідовного множення** на основу ***p***. Нехай число **A** записане в системі з основою ***q***. Вважаємо, що подання числа **A** в системі з основою ***p*** має вигляд:

$$A \equiv 0, a_{-1}a_{-2}...a_{-m} = 0 + a_{-1}p^{-1} + a_{-2}p^{-2} ... + a_{-m}p^{-m} \qquad (1.4)$$

Помножимо число **A** на основу ***p***. У результаті множення отримаємо число

$$A_1 = a_{-1}, a_{-2}...a_{-m} = a_{-1} + a_{-2}p^{-1} ... + a_{-m}p^{-m+1}.$$

Ціла частина числа $\mathbf{A_1}$ рівна старшому дробовому розряду a_{-1}. Відкидаючи цілу частину і помноживши залишок на ***p*** отримаємо наступний розряд a_{-2} і т. д. Слід зауважити, що в загальному випадку дробові числа, записані в одній системі числення у вигляді скінченного дробового числа, в іншій системі можуть мати нескінчене подання. Наприклад, перевести число 0,703125d в двійкову та вісімкову системи числення:

```
0,<--    0,703125          0,<--    0,703125
  ×  _______2              ×  _______8
1 <--    1,406250          5 <--    5,625000
  ×  _______2              ×  _______8
0 <--    0,812500          5 <--    5,000000
  ×  _______2              ×  _______8
1 <--    1,625000          0 <--    0,000000
  ×  _______2
1 <--    1,250000
  ×  _______2
0 <--    0,500000
  ×  _______2
1 <--    1,000000
  ×  _______2
0 <--    0,000000
```

На відміну від методу послідовного ділення, в методі послідовного множення отримують розряди у тій самій послідовності, як вони розташовані у запису числа. Тому 0,703125d = 0,101101b = 0,55q, записується у вигляді скінченного дробу. Інший приклад, перевести число 0,3d в двійкову та вісімкову системи числення:

0,<--	0,3		0,<--	0,3	
	× 2			× 8	
0 <--	0,6		2 <--	2,4	
	× 2			× 8	
1 <--	1,2		3 <--	3,2	
	× 2			× 8	
0 <--	0,4		1 <--	1,6	
	× 2			× 8	
0 <--	0,8		4 <--	4,8	
	× 2			× 8	
1 <--	1,6		6 <--	6,4	
	× 2			× 8	
1 <--	1,2		3 <--	3,2	
	і т.д.			і т.д.	

Таким чином 0,3d = 0,0100110011...b = 0,2314631463...q. Враховуючи форму запису нескінчених дробів, маємо 0,3d = 0,0(1001)b = 0,2(3146)q. Оскільки для зберігання чисел в ЕОМ використовують комірки скінчених розмірів, то такі числа можуть бути подані в ЕОМ тільки з певною похибкою.

Для переведення довільного числа в іншу систему числення необхідно окремо перевести цілу і дробову частини числа, а результат об'єднати. Слід зауважити, що метод послідовного ділення (множення) застосовний для будь-яких q, $p \geq 2$, однак його використовують зазвичай у тих випадках, коли необхідно перевести число із системи з більшою основою до системи з меншою, тобто коли $q > p$.

1.7. Переведення чисел з двійкової в вісімкову або шістнадцяткову, та навпаки

Якщо основа однієї системи числення є цілим степенем основи іншої системи, то переведення чисел між цими системами значно спрощується. Розглянемо перехід між двійковою та вісімковою системами числення. Отримані результати можна поширювати і на інші подібні випадки.

Нехай задано невід'ємне ціле число **A** в вісімковій системі числення:

$$A \equiv a_{n-1}a_{n-2}...a_1a_0 = a_{n-1}8^{n-1} + a_{n-2}8^{n-2} + ... + a_1 8^1 + a_0. \qquad (1.5)$$

Будемо вважати, що подання даного числа **A** в двійковій системі числення має вигляд:

$$A \equiv b_{l-1}b_{l-2}...b_1b_0 = b_{l-1}2^{l-1} + b_{l-2}2^{l-2} + ... + b_1 2^1 + b_0. \qquad (1.6)$$

Оскільки рівності (1.5) та (1.6) подають однакове число, то

$$\begin{aligned} A &= a_{n-1}8^{n-1} + a_{n-2}8^{n-2} + ... + a_2 8^2 + a_1 8^1 + a_0 = \\ &= b_{l-1}2^{l-1} + b_{l-2}2^{l-2} + ... + b_3 2^3 + b_2 2^2 + b_1 2^1 + b_0 \end{aligned} \qquad (1.7)$$

Розділивши число **A** на 8 отримаємо одинакові цілі частини $\mathbf{A_1}$

$$\begin{aligned} A_1 &= a_{n-1}8^{n-2} + a_{n-2}8^{n-3} + ... + a_3 8^2 + a_2 8^1 + a_1 = \\ &= b_{l-1}2^{l-4} + b_{l-2}2^{l-5} + ... + b_6 2^3 + b_5 2^2 + b_4 2^1 + b_3 \end{aligned} \quad (1.8)$$

та одинакові залишки

$$a_0 = b_2 2^2 + b_1 2^1 + b_0 \quad (1.9)$$

Вираз (1.9) свідчить про те, що молодшу вісімкову цифру a_0 можна подати трирозрядним двійковим числом $b_2 b_1 b_0$. Якщо рівність (1.8) знову поділити на 8, то в аналогічний спосіб отримаємо подання наступної вісімкової цифри у вигляді трирозрядного числа $b_5 b_4 b_3$.

Таким чином, **для переведення вісімкового числа в двійкову систему числення достатньо кожну вісімкову цифру замінити відповідним їй трирозрядним двійковим числом (двійковою тріадою)**. Наприклад, перевести число 564q в двійкову систему числення:

```
 5     6     4     q = 101110100b.
101   110   100
```

Переведення дробового числа з двійкової у вісімкову систему числення здійснюється аналогічною заміною двійкової тріади вісімковою цифрою. Отже, для числа 111010011,000101110b маємо:

```
111   010   011 , 000   101   110 b = 723,056q.
 7     2     3   ,  0     5     6
```

Переведення з двійкової в вісімкову систему числення полягає в заміні кожної двійкової тріади відповідною їй вісімковою цифрою. Необхідно однак пам'ятати два правила:

1. Двійковий запис розділяють на тріади починаючи від дробової коми вліво та вправо.
2. Якщо у результаті розділення крайні тріади виявляться неповними, їх слід доповнити нулями.

Наприклад, перевести в вісімкову систему числення двійкове число: 1101010010010101000,1000100011b. Розбиваємо число на тріади:

```
001 101 010 010 010 101 000,100 010 001 100b = 1522250,4214q.
 1   5   2   2   2   5   0 , 4   2   1   4
```

(Тут було доповнено нулями обидві крайні тріади).

Переведення з шістнадцяткової системи числення в двійкову та навпаки здійснюється аналогічно, як і у випадку вісімкової системи числення, з тією тільки відмінністю, що кожне шістнадцяткове число замінюють відповідним йому (табл. 1.3) чотирирозрядним двійковим числом (тетрадою) та навпаки. Наприклад: Перевести шістнадцяткове число 1EB6,C3h у двійкову систему числення:

```
 1      E      B      6    ,  C      3    h  = 1111010110110,11000011b.
0001   1110   1011   0110 , 1100   0011
```

або 100100100100000100,101010000100b в шістнадцяткову:

0010 0100 1001 0000 0100 , 1010 1000 0100 b = 24904,A84h.
2 4 9 0 4 , A 8 4

Якщо необхідно перейти від вісімкової системи до шістнадцяткової або навпаки, то слід спочатку перевести число в двійкову систему, а далі в ту, яка необхідна:

2 C 7 , A 4 h = 1011000111,101001b = 1307,51q.
0010 1100 0111 , 1010 0100
перегруповуємо тетради в тріади:
001 011 000 111 , 101 001
1 3 0 7 , 5 1

Враховуючи сказане вище, можна вказати простий спосіб переходу з десяткової системи числення в шістнадцяткову:

десяткова → двійкова → шістнадцяткова.

Розділ 2. СПОСОБИ ПОДАННЯ ВІД'ЄМНОГО ЧИСЛА

2.1. Знакові та числові розряди

Над числами, записаними в двійковій системі, можна здійснювати арифметичні дії додавання, віднімання, множення тощо. З шкільного курсу математики відомо, що віднімання від числа **A** числа **B** можна подати у вигляді додавання до числа **A** від'ємного числа **-B**: **A - B** = **A** + (**-B**). З іншого боку, більшість задач математики неможливо розв'язати без використання від'ємних чисел. Тому постає запитання: яким чином у комп'ютері записати від'ємне число?

Оскільки комп'ютер оперує тільки з двійковими розрядами, то для запису знаку можна використати додатковий, так званий "знаковий" розряд. Цей розряд є найстаршим в записі числа. **Знак "+" позначають нулем, а знак "-" – одиницею (у зміщеному коді навпаки).** Записуючи двійкові числа на папері, будемо відділяти знаковий розряд від цифрових розрядів крапкою. Однак у пам'яті ЕОМ число записується без виділення додаткових розрядів на крапку (чи кому). Наприклад:

$0.a_{n-1}a_{n-2}...a_i...a_1a_0, a_{-1}a_{-2}...a_{-m}$ - додатне число,

$1.b_{n-1}b_{n-2}...b_i...b_1b_0, b_{-1}b_{-2}...b_{-m}$ - від'ємне число.

Слід враховувати також, що для зберігання будь-якого числа в пам'яті комп'ютера завжди відводиться скінченна кількість розрядів. Зазвичай ця величина кратна 8-ми. Знак займає один старший розряд, тому на число залишається $8\times\boldsymbol{k}$ - 1 розрядів. Ці розряди можуть бути розділені на цілу і дробову частини довільним чином (за бажанням програміста). На практиці дробові числа, як правило, записують в форматі чисел з рухомою комою стандарту IEEE-754, а звичайну форму запису використовують для цілих чисел. Однак ми будемо розглядати узагальнений варіант, тобто коли число має один знаковий, ***n*** цілих та ***m*** дробових розрядів. Висновки, отримані для такого формату числа, можна поширювати на інші, спрощені форми подання чисел. Існують чотири основні способи запису цілого числа: прямий, інверсний, доповняльний та зміщений коди.

2.2. Прямий код

Прямий код відповідає звичному запису числа зі своїм знаком.
Числа в прямому коді можна подати у формі:

$$\mathbf{A} = (0.a_{n-1}a_{n-2}...a_i...a_1a_0, a_{-1}a_{-2}...a_{-m})_{np} = +\sum_{i=-m}^{n-1} a_i 2^i, \qquad (2.1)$$

$$\mathbf{B} = (1.b_{n-1}b_{n-2}...b_i...b_1b_0, b_{-1}b_{-2}...b_{-m})_{np} = -\sum_{i=-m}^{n-1} b_i 2^i. \qquad (2.2)$$

Наприклад, число 13,75d записується як $0.1101{,}11b_{пр}$, а число -13,75d – як $1.1101{,}11b_{пр}$. Нуль в прямому коді має два подання:

$$0.00...00_{пр} = +0 \quad \text{та} \quad 1.00...00_{пр} = -0.$$

Арифметичні операції в прямому коді виконують за звичайними правилами двійкової арифметики, однак їх не завжди можна застосовувати до знакового розряду числа:

Приклади 2.1 та 2.2.

+14d = 0.001110b$_{пр}$	-14d = 1.001110b$_{пр}$
+ <u>+45d = 0.101101b$_{пр}$</u>	+ <u>-45d = 1.101101b$_{пр}$</u>
+59d = 0.111011b$_{пр}$	+59d = 0.111011b$_{пр}$

У другому випадку в результаті додавання двох від'ємних чисел отримали додатне число, тому результат некоректний (одиниця перенесення в наступний старший розряд вже не буде "знакова"). Отже, **у випадку додавання чисел в прямому коді не можна розглядати знаковий розряд як звичайний цифровий.**

Прямий код застосовують в ЕОМ для введення чи виведення інформації, а також для зберігання чисел в запам'ятовуючих пристроях. Однак прямий код незручний для реалізації арифметичних дій в ЕОМ, оскільки вимагає від арифметичного пристрою виконання як операції додавання, так і операції віднімання. **Застосування інверсного або доповняльного кодів** позволяє суттєво спростити арифметичний пристрій за рахунок **заміни операції віднімання операцією додавання**.

2.3. Інверсний (зворотний) код.

Загальна ідея подання від'ємних чисел в інверсному та доповняльному кодах полягає у введенні спеціального розряду з від'ємним ваговим коефіцієнтом. Число **A** записують у вигляді

$$\mathbf{A} = a_n.a_{n-1}a_{n-2}...a_1a_0, a_{-1}...a_{-m} = a_n(-C) + \sum_{i=-m}^{n-1} a_i 2^i . \qquad (2.3)$$

Коефіцієнт $\boldsymbol{C} > 0$ визначає діапазон зображуваних чисел. Введення розряду з від'ємною вагою дозволяє замінити операцію віднімання операцією алгебричного додавання чисел з різними знаками. В цьому випадку арифметичні дії здійснюють над усіма розрядами, в тому числі знаковому.

Для утворення інверсного коду коефіцієнт *C* у формулі (2.3) вибирають рівним максимальному додатному числу, яке може бути записано в комірку з *n* цілими та *m* дробовими розрядами, тобто

$$\boldsymbol{C} = 2^n - 2^{-m}.$$

Таким чином, двійкове число в інверсному коді записують у вигляді

$$\mathbf{A} = (a_n.a_{n-1}a_{n-2}...a_1a_0, a_{-1}...a_{-m})_{\text{інв}} = a_n(-2^n + 2^{-m}) + \sum_{i=-m}^{n-1} a_i 2^i . \qquad (2.4)$$

Використовуючи (2.4) можна знайти значення чисел, записаних в інверсному коді:

$$\text{A} = 0.0110{,}01_{\text{інв}} = 0\cdot(-2^4 + 2^{-2}) + 0\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0 + 0\cdot 2^{-1} + 1\cdot 2^{-2} = 6{,}25\text{d},$$

$$\text{B} = 1.0110{,}01_{\text{інв}} = 1\cdot(-2^4 + 2^{-2}) + 0\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0 + 0\cdot 2^{-1} + 1\cdot 2^{-2} = -9{,}50\text{d}.$$

Додатні числа в прямому та інверсному кодах мають однакову форму запису, оскільки коефіцієнт при ***C***: $a_n = 0$. Для того, щоби знайти правило переходу від прямого коду до інверсного прирівняємо праві частини співвідношень (2.2) та (2.4).

$$-\sum_{i=-m}^{n-1} b_i 2^i = a_n(-2^n + 2^{-m}) + \sum_{i=-m}^{n-1} a_i 2^i . \qquad (2.5)$$

Враховуючи, що для від'ємних чисел $a_n = 1$, а

$$-2^n + 2^{-m} = -\sum_{i=-m}^{n-1} 2^i \qquad (2.6)$$

як сума членів геометричної прогресії, можна записати рівняння (2.5) у вигляді

$$-\sum_{i=-m}^{n-1} b_i \cdot 2^i = -\sum_{i=-m}^{n-1} (1 - a_i) \cdot 2^i . \qquad (2.7)$$

З (2.7) випливає, що для будь-яких ***i*** повинна виконуватись рівність $b_i = 1 - a_i$, або $a_i = 1 - b_i$ тобто якщо $b_i = 1$, то $a_i = 0$ і навпаки.

Таким чином, **для отримання інверсного коду від'ємного числа необхідно в розряд з від'ємною вагою записати одиницю, а у всіх інших розрядах прямого коду інвертувати значення розрядів.** Аналогічно отримують прямий код з інверсного.

$$\text{A} = -10110{,}1011\text{b} = 1.10110{,}1011\text{b}_{\text{пр}} = 1.01001{,}0100\text{b}_{\text{інв}} .$$

Увага! В інверсному коді від'ємного числа нулі, записані в кінці числа і на початку (крім знакового розряду), відкидати не можна, оскільки вони значущі. Якщо для додатних чисел, записаних в будь-якому коді можна дописувати чи відкидати нулі справа та зліва, то для від'ємних чисел в інверсному коді так поступати можна тільки з одиницями. Це означає, що числа $1.11...11010{,}1011...11\text{b}_{\text{об}}$ та $1.010{,}10\text{b}_{\text{об}}$ є тотожні.

Нуль в інверсному коді також має два значення:

$$(+0)_{\text{об}} = 0.00...00{,}00...00\text{b}_{\text{інв}} = 0\cdot(-2^{n}+2^{-m})+0\cdot 2^{n-1}+...+0\cdot 2^{0}+0\cdot 2^{-1}+...+0\cdot 2^{-m} = 0{,}00,$$
$$(-0)_{\text{об}} = 1.11...11{,}11...11\text{b}_{\text{інв}} = 1\cdot(-2^{n}+2^{-m})+1\cdot 2^{n-1}+...+1\cdot 2^{0}+1\cdot 2^{-1}+...+1\cdot 2^{-m} = 0{,}00.$$

Від'ємні і додатні числа, записані в інверсному **коді, симетрично розміщені відносно нуля** (див. табл. 2.1 в кінці розділу).

Визначимо **діапазон чисел**, які можна подати в інверсному коді, використовуючи ***k = n+m+1*** розрядів, де ***n*** – кількість цілих, а ***m*** – кількість дробових розрядів плюс один знаковий. Максимальне додатне число $0.11...11{,}11...11_{\text{інв}}$ згідно з (2.4):

$$0_n.1_{n-1}1_{n-2}...1_1 1_0 , 1_{-1}...1_{-m} = 0\cdot(-2^n - 2^{-m}) + \sum_{i=-m}^{n-1} 1\cdot 2^i = +(2^n - 2^{-m}),$$

аналогічно мінімальне від'ємне число (максимальне за модулем) $1.00...00{,}00...00_{\text{інв}}$:

$$1_n.0_{n-1}0_{n-2}...0_1 0_0, 0_{-1}...0_{-m} = 1\cdot(-2^n + 2^{-m}) + \sum_{i=-m}^{n-1} 0\cdot 2^i = -(2^n - 2^{-m}).$$

Якщо ***m*** = 0, тобто дробові розряди відсутні, то діапазон чисел, які можна подати за допомогою ***k*** розрядів, становить від $-(2^{k-1}-1)$ до $+(2^{k-1}-1)$ (всього 2^k чисел, **враховуючи два нулі**).

2.4. Додавання чисел, записаних в інверсному коді.

Оскільки додатні числа в прямому та інверсному кодах мають однакову форму запису, то їх додають аналогічно, як і в прямому коді. Розглянемо додавання двох чисел з різними знаками. Нехай число **A** додатне, а число **B** – від'ємне і їх можна записати як:

$$\mathbf{A} = (0.a_{n-1}a_{n-2}...a_1a_0, a_{-1}a_{-2}...a_{-m})_{\text{інв}} = + \sum_{i=-m}^{n-1} a_i 2^i,$$

$$\mathbf{B} = (1.b_{n-1}b_{n-2}...b_1b_0, b_{-1}b_{-2}...b_{-m})_{\text{інв}} = -2^n + 2^{-m} + \sum_{i=-m}^{n-1} b_i 2^i.$$

Тоді їхню суму можна записати як:

$$\mathbf{C} = (c_n.c_{n-1}c_{n-2}...c_1c_0, c_{-1}c_{-2}...c_{-m})_{\text{інв}} = c_n(-2^n + 2^{-m}) + \sum_{i=-m}^{n-1} c_i 2^i$$

або

$$\mathbf{C} = c_n(-2^n + 2^{-m}) + \sum_{i=-m}^{n-1} c_i 2^i = -2^n + 2^{-m} + \sum_{i=-m}^{n-1} (a_i + b_i)\cdot 2^i. \qquad (2.8)$$

Значення суми $\sum_{i=-m}^{n-1}(a_i + b_i)\cdot 2^i$ можна знайти за допомогою звичайних правил додавання. Можливі два випадки:

1. $$\sum_{i=-m}^{n-1}(a_i + b_i)\cdot 2^i < 2^n.$$

Такій нерівності відповідає |**A**| < |**B**|, отже сума **C** – від'ємна ($c_n = 1$), а коефіцієнти c_i можна визначити з рівняння

$$\sum_{i=-m}^{n-1} c_i \cdot 2^i = \sum_{i=-m}^{n-1}(a_i + b_i)\cdot 2^i. \qquad (2.9)$$

Отже, в цьому випадку також справедливе звичайне правило додавання чисел:

Приклад 2.3.

$$\begin{array}{rl} & +6{,}50\text{d} = 0.0110{,}10\text{b}_{\text{інв}} \\ + & \underline{-10{,}00\text{d} = 1.0101{,}11\text{b}_{\text{інв}}} \\ & -3{,}50\text{d} = 1.1100{,}01\text{b}_{\text{інв}} \end{array}$$

2. $$\sum_{i=-m}^{n-1}(a_i + b_i)\cdot 2^i \geq 2^n. \qquad (2.10)$$

Нерівність означає, що $|\mathbf{A}| \geq |\mathbf{B}|$, тобто $\mathbf{C} \equiv \mathbf{A} - \mathbf{B} \geq 0$. В цьому випадку появляється одиниця перенесення в **n**-ний розряд числа, який відповідає за його знак, а отже можна записати

$$\sum_{i=-m}^{n-1}(a_i + b_i)\cdot 2^i = 2^n + \sum_{i=-m}^{n-1} c_i \cdot 2^i .$$

Тоді співвідношення (2.8) набуває вигляду

$$\mathbf{C}_{\text{інв}} = -2^n + 2^{-m} + 2^n + \sum_{i=-m}^{n-1} c_i 2^i = 2^{-m} + \sum_{i=-m}^{n-1} c_i \cdot 2^i . \qquad (2.11)$$

Це означає, що **для отримання інверсного коду суми в розряд з від'ємною вагою слід записати нуль** (оскільки **A - B** є число додатне), **а для того, щоб отримати інші розряди числа, необхідно до результату формального додавання додати одиницю молодшого розряду** (2^{-m}). Одночасно появляється одиниця перенесення із розряду з від'ємною вагою. Саме цю одиницю необхідно добавити до молодшого розряду. **Така операція, коли одиницю перенесення в n+1 розряд насправді додають до m-го називається циклічним перенесенням.**

Вищесказане ілюструє такий **приклад 2.4**:

```
    +13,50d =   0.1101,10b_інв
+    -5,00d =   1.1010,11b_інв
          10.1000,01
            |———→1
-------------------------------
    +8,50d =   0.1000,10b_інв
```

Аналогічно можна показати, що **у випадку додавання двох від'ємних чисел також слід виконувати операцію циклічного перенесення:**

Приклад 2.5.

```
    -5,00d =   1.1010,11b_інв
+   -3,75d =   1.1100,00b_інв
          11.0110,11
            |———→1
-------------------------------
    -8,75d =   1.0111,00b_інв
```

Таким чином, в випадку додавання чисел, записаних в інверсному коді, **завжди** слід виконувати операцію циклічного перенесення, однак, якщо перенесення в ***n*+1** розряд рівне нулю, то таке циклічне перенесення фактично аналогічне його відсутності. У випадку реалізації операції додавання чисел в ЕОМ, **операція циклічного перенесення призводить до ускладнення схеми суматора та збільшенню часу виконання операції додавання**. Причиною циклічного перенесення є наявність двох нулів (+0 та -0) у інверсному коді.

2.5. Доповняльний (доповнюючий) код

Ідея доповняльного коду виникає в зв'язку з бажанням позбутися операції циклічного перенесення. Для цього необхідно встановити коефіцієнт С в формулі (2.3) рівним -2^n. Тоді числа в доповняльному коді можна записати як:

$$\mathbf{A} = (a_n.a_{n-1}a_{n-2}...a_1a_0, a_{-1}...a_{-m})_{\text{доп}} = a_n(-2^n) + \sum_{i=-m}^{n-1} a_i 2^i . \qquad (2.12)$$

Порівнюючи (2.4) та (2.12) можна зробити наступні висновки:

1. **Додатні числа в прямому, інверсному та доповняльному кодах мають однакову форму запису.**
2. Від'ємні числа в інверсному та доповняльному кодах відрізняються на одиницю молодшого розряду. **Для утворення доповняльного коду від'ємного числа необхідно записати це число в інверсному коді та додати до нього одиницю молодшого розряду:**

Приклад 2.6.

$\mathbf{A}$ = +9,625d → $0.1001,101b_{\text{пр}}$ → $0.1001,101b_{\text{інв}}$ → $0.1001,101b_{\text{доп}}$

$\mathbf{B}$ = -9,625d → $1.1001,101b_{\text{пр}}$ → $1.0110,010b_{\text{інв}}$

$$+ \quad \underline{\qquad\qquad 1}$$

$1.0110,011b_{\text{доп}}$

Перевіримо $\mathbf{A} + \mathbf{B} \equiv 0$:

$$+9,625d \rightarrow 0.1001,101b_{\text{доп}}$$

$$+ \quad \underline{-9,625d \rightarrow 1.0110,011b_{\text{доп}}}$$

$$0,0d \quad \rightarrow 0.0000,000b_{\text{доп}}$$

Для утворення прямого коду від'ємного числа з доповняльного коду навпаки слід відняти одиницю молодшого розряду, а отриманий результат (інверсний код) інвертувати (замінити нулі одиницями, а одиниці нулями). Однак цей метод переведення незручний для використання в ЕОМ, оскільки містить операцію віднімання. **Тому для знаходження прямого коду з доповняльного достатньо утворити доповняльний до нього код:**

Приклад 2.7.

-9,625d = $1.0110,011b_{\text{доп}}$ → $(1.1001,100b_{\text{доп}})_{\text{інв}}$

$$+ \underline{\qquad\qquad 1}$$

$(1.1001,101b_{\text{доп}})_{\text{доп}} = 1.1001,101b_{\text{пр}}$

Нуль в доповняльному коді має єдине подання, оскільки:

Приклад 2.8.

$$(+0)_{\text{доп}} = 0.00...00,00...00b$$

$$(-0)_{\text{доп}} = 1.11...11,11...11b_{\text{інв}}$$

$$+ \underline{\qquad\qquad\qquad 1}$$

$0.00...00,00...00b_{\text{доп}} \equiv (+0)_{\text{доп}}$

Діапазон чисел, які можна подати в доповняльному коді, дещо інший, ніж для інверсного коду. Максимальне додатне число в цих кодах однакове. Згідно з (2.12) $0.11...11,11...11_{\text{доп}}$:

$$(0_n.1_{n-1}1_{n-2}...1_11_0, 1_{-1}...1_{-m})_{\text{доп}} = 0\cdot(-2^n) + \sum_{i=-m}^{n-1} 1\cdot 2^i = +(2^n - 2^{-m}).$$

Мінімальне від'ємне число (максимальне за модулем) $1.00...00,00...00_{\text{доп}}$:

$$(1_n.0_{n-1}0_{n-2}...0_1 0_0, 0_{-1}...0_{-m})_{\text{доп}} = 1\cdot(-2^n) + \sum_{i=-m}^{n-1} 0\cdot 2^i = -2^n.$$

Це означає, що **від'ємних чисел на одне більше ніж додатних**. Причиною цього є єдине подання нуля в доповняльному коді на відміну від двох нулів в інверсному коді. Насправді ж нуль також додатний – $(+0)_{\text{доп}}$ = 0.00...00,00...00b, тому фактично додатних і від'ємних чисел порівну.

Якщо $\boldsymbol{m} = 0$, тобто дробові розряди відсутні, то діапазон чисел, які можна подати за допомогою $\boldsymbol{k}$ розрядів, становить від -2^{k-1} до $+(2^{k-1}-1)$ (всього 2^k чисел, **враховуючи один нуль**). Подання 8-розрядних цілих чисел, прямому, інверсному та доповняльному кодах демонструє табл. 2.1.

Додавання в доповняльному коді здійснюють відповідно до правил звичайного додавання двійкових чисел, розглядаючи знакові розряди як звичайні цифрові. У випадку появи одиниці перенесення зі знакового розряду її слід відкинути, тобто циклічне перенесення відсутнє.

Приклад 2.9.

+13,50d = 0.1101,10b$_{\text{пр}}$ =	0.1101,10b$_{\text{інв}}$ =	0.1101,10b$_{\text{доп}}$
+ -5,00d = 1.0101,00b$_{\text{пр}}$ =	1.1010,11b$_{\text{інв}}$ =	1.1011,00b$_{\text{доп}}$
	10.1000,01	10.1000,10b$_{\text{доп}}$
	↳——→1	↓відкинути
+8,50d = 0.1000,10b$_{\text{пр}}$ =	0.1000,10b$_{\text{інв}}$ =	0.1000,10b$_{\text{доп}}$

Доповняльний код використовується в більшості обчислювальних пристроїв, наприклад, всі операції віднімання здійснюються як операції додавання числа з протилежним знаком у доповняльному коді.

2.6. Основні принципи побудови зміщеного коду [5, 6]

Цей код дуже подібний до доповняльного і відрізняється від нього тільки позначенням знакового розряду. Якщо в доповняльному коді додатні числа мають у знаковому розряді 0, а від'ємні – 1, то у зміщеного коду картина протилежна: додатні числа мають знаковий розряд рівний 1, а від'ємні – 0. Нуль, залежно від величини зміщення, може мати знаковий розряд рівний 1 або 0. Тому розрізняють "додатний нуль" та "від'ємний нуль", відповідно.

$(1.a_{n-1}a_{n-2}...a_i...a_1a_0, a_{-1}a_{-2}...a_{-m})_{\text{зм}}$ – додатне число,

$(0.b_{n-1}b_{n-2}...b_i...b_1b_0, b_{-1}b_{-2}...b_{-m})_{\text{зм}}$ – від'ємне число.

Свою назву "зміщений" код отримав тому, що всю шкалу кодування від 0.00...00,00..00$_{\text{зм}}$ до 1.11...11,11...11$_{\text{зм}}$ можна отримати шляхом послідовного зсуву (зміщення) шкали беззнакових двійкових чисел від 00...00,00..00 до 11...11,11...11 на деяку величину, яка визначає вид зміщеного коду. Це добре видно з табл. 2.1, де подано всі способи кодування цілих двійкових чисел розміром 8 біт.

Зміщений код, як правило, використовують для подання цілих чисел, які задають порядки чисел з рухомою комою (див. п. 3.3.2), а також для кодування

діапазону вхідних сигналів аналого-цифрових перетворювачів. Число **A** в зміщеному коді записують як:

$$\mathbf{A} = (a_n.a_{n-1}a_{n-2}...a_1a_0, a_{-1}...a_{-m})_{\text{зм}} =$$

$$= a_n \cdot 2^n + \sum_{i=-m}^{n-1} a_i 2^i - C = \sum_{i=-m}^{n} a_i 2^i - C = A_{\text{зм}} - C, \qquad (2.13)$$

де величина ***C*** називається зміщенням і може набувати деяких певних значень. Величина $A_{\text{зм}} = \sum_{i=-m}^{n} a_i 2^i$ може набувати тільки додатних значень і фактично дорівнює значенню двійкового беззнакового числа, яке може бути записане в 1+***n***+***m*** розряди (включно зі знаковим). З рівняння (2.13) слідує, що $\mathbf{A}_{\text{зм}} = \mathbf{A} + \boldsymbol{C}$, **тобто зміщений код числа A можна отримати, якщо до цього числа додати зміщення, а результат подати у вигляді беззнакового двійкового числа.**

2.7. Зміщений код з додатнім нулем

Зазвичай **цей код скорочено називають просто "зміщений код".** Тут зміщення ***C*** рівне вазі старшого розряду подання, тобто $\boldsymbol{C} = 2^n$. Тоді рівняння (2.13) можна записати у вигляді:

$$\mathbf{A} = (a_n.a_{n-1}a_{n-2}...a_1a_0, a_{-1}...a_{-m})_{\text{зм}} =$$

$$= a_n \cdot 2^n + \sum_{i=-m}^{n-1} a_i 2^i - 2^n = (a_n - 1) \cdot 2^n + \sum_{i=-m}^{n-1} a_i 2^i . \qquad (2.14)$$

У зміщеному коді число нуль має єдине подання: (+0) = 1.00...00,00...00$b_{\text{зм}}$. **Оскільки знаковий розряд встановлено в 1, що відповідає знаку "плюс", то такий нуль називають "додатнім нулем"**. Числа в зміщеному коді розміщені в "звичному" для нас порядку від -(2^n) (код 0.00...00,00...00$b_{\text{зм}}$) до +(2^n - 2^{-m}) (код 1.11...11,11...11$b_{\text{зм}}$). Як і в доповняльному коді від'ємних чисел на одне більше, ніж додатних.

Існує просте **правило переходу від доповняльного до зміщеного (і навпаки): для переходу необхідно інвертувати знаковий розряд коду.** Це правило можна отримати, якщо записати число **B** в доповняльному коді згідно з (2.12):

$$\mathbf{B} = (b_n.b_{n-1}b_{n-2}...b_1b_0, b_{-1}...b_{-m})_{\text{доп}} = b_n(-2^n) + \sum_{i=-m}^{n-1} b_i 2^i$$

Вважаючи, що **A** = **B** отримуємо, що

$$(a_n - 1) \cdot 2^n + \sum_{i=-m}^{n-1} a_i 2^i = b_n(-2^n) + \sum_{i=-m}^{n-1} b_i 2^i . \qquad (2.15)$$

Ліва частина рівняння буде дорівнювати правій, якщо задовольняються такі умови:

$$\begin{cases} a_i = b_i, \text{ для всіх } i \in (-m...n-1); \\ 1 - a_n = b_n. \end{cases}$$

Перша умова означає, що запис числових розрядів числа в зміщеному коді тотожний запису в доповняльному коді, а друга – те, що якщо $a_n = 0$ то $b_n = 1$ і

навпаки, якщо $a_n = 1$ то $b_n = 0$, тобто знакові розряди в зміщеному і доповняльному кодах є взаємно інверсними.

Приклад 2.10. Подати числа +12,5d та -6,25d в зміщеному коді, якщо задано 1 знаковий, 4 цілих та 2 дробових розряди.

Розв'язування: Оскільки n = 4, то зміщення $\boldsymbol{C} = 2^4 = 16\text{d} = 1.0000{,}00\text{b}$.

1-й спосіб

$+12{,}5\text{d} + 16\text{d} = 28{,}5\text{d} = 1.1100{,}10\text{b}_{\text{зм}}$,

$- 6{,}25\text{d} + 16\text{d} = 9{,}75\text{d} = 0.1001{,}11\text{b}_{\text{зм}}$.

2-й спосіб

$+12{,}50\text{d} = 0.1100{,}10\text{b}_{\text{пр}} = 0.1100{,}10\text{b}_{\text{інв}} = 0.1100{,}10\text{b}_{\text{доп}} = 1.1100{,}10\text{b}_{\text{зм}}$,

інвертуємо розряд ↑

$-6{,}25\text{d} = 1.0110{,}01\text{b}_{\text{пр}} = 1.1001{,}10\text{b}_{\text{інв}} = 1.1001{,}11\text{b}_{\text{доп}} = 0.1001{,}11\text{b}_{\text{зм}}$.

інвертуємо розряд ↑

Важливою особливістю зміщеного коду є те, що для будь-яких чисел **A** та **B** таких, що **A** > **B** слідує, що $\mathbf{A}_{\text{зм}} > \mathbf{B}_{\text{зм}}$. Використання зміщених кодів спрощує порівняння чисел зі знаком, оскільки воно зводиться до простого порівняння кодів подання цих чисел.

Додавання чисел у зміщеному коді здійснюють згідно з правилами додавання двійкових чисел у доповняльному коді, однак знаковий розряд отриманої суми необхідно інвертувати. Ця вимога пов'язана з тим, що оскільки величина зміщення кожного із доданків становить 2^n, то величина зміщення суми (різниці) буде рівна $2^n + 2^n = 2^{n+1}$. Це число перевищує можливість подання числа за допомогою вибраної кількості розрядів, а отже фактично тотожне нулю. З іншого боку, величина зміщення суми також повинна становити 2^n, а додавання такого числа еквівалентне інверсії знакового розряду.

Віднімання в зміщеному коді реалізують шляхом додавання до зменшуваного від'ємника з протилежним знаком. Знак суми також слід інвертувати. Правила додавання та віднімання демонструє такий приклад:

Приклад 2.11. Над числами 30d, -30d, 70d та -70d поданими в 8-ми розрядному зміщеному коді виконати арифметичні дії додавання (віднімання).

Подамо числа в зміщеному коді:

$+30\text{d} = 0.0011110\text{b}_{\text{пр}} = 0.0011110\text{b}_{\text{інв}} = 0.0011110\text{b}_{\text{доп}} = 1.0011110\text{b}_{\text{зм}}$

$-30\text{d} = 1.0011110\text{b}_{\text{пр}} = 1.1100001\text{b}_{\text{інв}} = 1.1100010\text{b}_{\text{доп}} = 0.1100010\text{b}_{\text{зм}}$

$+70\text{d} = 0.1000110\text{b}_{\text{пр}} = 0.1000110\text{b}_{\text{інв}} = 0.1000110\text{b}_{\text{доп}} = 1.1000110\text{b}_{\text{зм}}$

$-70\text{d} = 1.1000110\text{b}_{\text{пр}} = 1.0111001\text{b}_{\text{інв}} = 1.0111010\text{b}_{\text{доп}} = 0.0111010\text{b}_{\text{зм}}$

інвертуємо розряди ↑

$+30\text{d} = 1.0011110\text{b}_{\text{зм}}$

$+ \quad +70\text{d} = \underline{1.1000110\text{b}_{\text{зм}}}$

10.1100100

відкинути ↓↑ інвертувати розряд суми

$1.1100100\text{b}_{\text{зм}} = 0.1100100\text{b}_{\text{доп}} = 0.1100100\text{b}_{\text{пр}} = +100\text{d}.$

↑ інвертувати для отримання доповняльного коду

$$
\begin{array}{rl}
 & -30d = 0.1100010b_{зм} \\
+ & -70d = \underline{0.0111010b_{зм}} \\
 & \quad 01.0011100
\end{array}
$$

відкинути ↓↑ інвертувати розряд суми

$0.0011100b_{зм} = 1.0011100b_{доп} = 1.1100100b_{пр} = -100d.$

↑ інвертувати для отримання доповняльного коду

$$
\begin{array}{rl}
 & +30d = 1.0011110b_{зм} \\
+ & -70d = \underline{0.0111010b_{зм}} \\
 & \quad 01.1011000
\end{array}
$$

відкинути ↓↑ інвертувати розряд суми

$0.1011000b_{зм} = 1.1011000b_{доп} = 1.0101000b_{пр} = -40d.$

↑ інвертувати для отримання доповняльного коду

$$
\begin{array}{rl}
 & -30d = 0.1100010b_{зм} \\
+ & +70d = \underline{1.1000110b_{зм}} \\
 & \quad 10.0101000
\end{array}
$$

відкинути ↓↑ інвертувати розряд суми

$1.0101000b_{зм} = 0.0101000b_{доп} = 0.0101000b_{пр} = +40d.$

↑ інвертувати для отримання доповняльного коду

Зміщений код з додатнім нулем використовують також для зберігання цілих чисел в пам'яті ЕОМ (див. табл. 3.2).

2.8. Зміщений код з від'ємним нулем

В цьому коді зміщення ***C*** становить $2^n - 2^{-m}$, тобто менше від зміщення для коду з додатнім нулем на одиницю молодшого розряду. Саме завдяки цьому значення чисел, які відповідають кодовим комбінаціям, зміщуються на одну позицію (див. табл. 2.1) відносно зміщеного коду з додатнім нулем, а **нуль переходить із додатного подання в від'ємне. Тому такий код називають зміщеним кодом з від'ємним нулем.** Більшість властивостей зміщеного коду залишаються незмінними, за винятком правил переведення чисел з інших кодів та виконання над ними арифметичних дій.

Число нуль також має єдине подання: $(-0) = 0.11...11,11...11b_{зм(-0)}$. Діапазон чисел дещо змінюється і становить від $-(2^n - 2^{-m})$ (код $0.00...00,00...00b_{зм(-0)}$) до $+(2^n)$ (код $1.11...11,11...11b_{зм(-0)}$). Від'ємних чисел стає на одне менше, ніж додатних. Числа в зміщеному коді з від'ємним нулем відрізняються від їх подання в зміщеному коді з додатнім нулем на одиницю молодшого розряду, тому **для переходу від кодування з додатнім нулем у кодування з від'ємним нулем слід зменшити подання числа на одиницю молодшого розряду, або скористатися 1-м способом переведення в зміщений код, враховуючи, що зміщення *C* становить $2^n - 2^{-m}$.**

Приклад 2.12. Подати числа +10,25d та -9,5d в зміщеному коді з від'ємним нулем, якщо задано 1 знаковий, 4 цілих та 2 дробових розряди.

Розв'язування: Оскільки n = 4, а m = 2, то зміщення $\boldsymbol{C} = 2^4 - 2^{-2} = 15{,}75\text{d} =$ $= 0.1111{,}11\text{b}$.

1-й спосіб

$+10{,}25\text{d} + 15{,}75\text{d} = 26{,}0\text{d} = 1.1010{,}00\text{b}_{\text{зм(-0)}}$

$-\ 9{,}50\text{d} + 15{,}75\text{d} = 6{,}25\text{d} = 0.0110{,}01\text{b}_{\text{зм(-0)}}$

2-й спосіб

$+10{,}25\text{d} = 0.1010{,}01\text{b}_{\text{пр}} = 0.1010{,}01\text{b}_{\text{інв}} = 0.1010{,}01\text{b}_{\text{доп}} = 1.1010{,}01\text{b}_{\text{зм}}$

інвертуємо розряд ↑ $\quad -\ \underline{\qquad\quad 1}$

$1.1010{,}00\text{b}_{\text{зм(-0)}}.$

$-\ 9{,}50\text{d} = 1.1001{,}10\text{b}_{\text{пр}} = 1.0110{,}01\text{b}_{\text{об}} = 1.0110{,}10\text{b}_{\text{доп}} = 0.0110{,}10\text{b}_{\text{зм}}$

інвертуємо розряд ↑ $\quad -\ \underline{\qquad\quad 1}$

$0.0110{,}01\text{b}_{\text{зм(-0)}}.$

Додавання чисел у зміщеному коді з від'ємним нулем здійснюють як і для зміщеного коду з додатнім нулем, тобто згідно з правилами додавання двійкових чисел в доповняльному коді з наступною інверсією знаку суми, однак **отриманий результат слід збільшити на одиницю молодшого розряду.**

Приклад 2.13. Здійснити арифметичні дії: 45d + 30d та 75d + (- 100d) поданими в 8-розрядному зміщеному коді з від'ємним нулем.

Подамо числа в зміщеному коді з від'ємним нулем:

$+30\text{d} = 0.0011110\text{b}_{\text{пр}} = 0.0011110\text{b}_{\text{об}} = 0.0011110\text{b}_{\text{доп}} = 1.0011110\text{b}_{\text{зм}} =$

інвертуємо розряд ↑ $\quad -\underline{\qquad 1}$

$1.0011101\text{b}_{\text{зм(-0)}}.$

$+45\text{d} = 0.0101101\text{b}_{\text{пр}} = 0.0101101\text{b}_{\text{об}} = 0.0101101\text{b}_{\text{доп}} = 1.0101101\text{b}_{\text{зм}} =$

інвертуємо розряд ↑ $\quad -\underline{\qquad 1}$

$1.0101100\text{b}_{\text{зм(-0)}}.$

$+75\text{d} = 0.1001011\text{b}_{\text{пр}} = 0.1001011\text{b}_{\text{об}} = 0.1001011\text{b}_{\text{доп}} = 1.1001011\text{b}_{\text{зм}} =$

інвертуємо розряд ↑ $\quad -\underline{\qquad 1}$

$1.1001010\text{b}_{\text{зм(-0)}}.$

$-100\text{d} = 1.1100100\text{b}_{\text{пр}} = 1.0011011\text{b}_{\text{об}} = 1.0011100\text{b}_{\text{доп}} = 0.0011100\text{b}_{\text{зм}} =$

інвертуємо розряд ↑ $\quad -\underline{\qquad 1}$

$0.0011011\text{b}_{\text{зм(-0)}}.$

$+30\text{d} = 1.0011101\text{b}_{\text{зм(-0)}}$

$+\quad +45\text{d} = \underline{1.0101100\text{b}_{\text{зм(-0)}}}$

10.1001001

відкинути ↓↑ інвертувати розряд суми

1.1001001

$+\underline{\qquad\quad 1}$

$1.1001010\text{b}_{\text{зм(-0)}} = 75\text{d}$ (див. вище).

$$\begin{array}{r} +75d = 1.1001010b_{зм(-0)} \\ + \quad -100d = \underline{0.0011011}b_{зм(-0)} \\ 01.1100101 \end{array}$$

відкинути ↓↑ інвертувати розряд суми

$$\begin{array}{r} 0.1100101 \\ + \underline{\qquad\quad 1} \\ 0.1100110b_{зм(-0)}. \end{array}$$

Перевіримо:

$$0.1100110b_{зм(-0)} = 0\cdot2^7 + 1\cdot2^6 + 1\cdot2^5 + 0\cdot2^4 + 0\cdot2^3 + 1\cdot2^2 + 1\cdot2^1 + 0\cdot2^0 - (2^7 - 2^0) =$$
$$= 64 + 32 + 4 + 2 - 127 = -25d.$$

(У цьому прикладі зміщення $\boldsymbol{C} = 2^7 - 2^0$, оскільки в поданні 7 числових та один знаковий розряд.)

Зміщений код з від'ємним нулем використовують для подання експоненти в двійкових числах з рухомою комою (див. п. 3.3.2).

2.9. Переповнення розрядної сітки

Кількість розрядів ЕОМ, які використовують для зберігання числа, є обмеженою. Тому сума двох чисел з однаковими знаками може виявитися більшою за модулем, ніж максимальне число, яке можна записати за допомогою заданої кількості двійкових розрядів (формату подання) і результат виявиться некоректним. Таке явище називають **переповненням розрядної сітки або виходом за межі діапазону подання**.

Розглянемо такий приклад:

Приклад 2.14.

A = +27d = +11011b → $0.11011b_{інв}$ → $0.11011b_{доп}$

B = +19d = +10011b → $\underline{0.10011}b_{інв}$ → $\underline{0.10011}b_{доп}$

A + **B** = $01.01110b_{інв}$ = $1.01110b_{доп} = 1.10010b_{пр} = -18d$

└——→0

$1.01110b_{інв} = -17d$

У цьому прикладі одиниця перенесення зі старшого цифрового розряду попадає в знаковий розряд, а отже результатом додавання двох додатних чисел буде зафіксоване від'ємне число. Причина полягає в тому, що 5-ти цифрових розрядів не вистачає для запису числа +46d (**A** + **B**), оскільки діапазон чисел, які можна записати в ці розряди для доповняльного коду становить від -32d до +31d.

У випадку додавання від'ємних чисел переповнення наступає тоді, коли відсутня одиниця перенесення зі старшого цифрового розряду в знаковий розряд:

Приклад 2.15.

A = -22d = -10110b → $1.01001b_{інв}$ → $1.01010b_{доп}$

B = -25d = -11001b → $\underline{1.00110}b_{інв}$ → $\underline{1.00111}b_{доп}$

A + **B** = $10.01111b_{інв}$ $10.10001b_{доп} = 0.10001b_{пр} = +17d$

└——→1

$0.10000b_{об} = +16d$

Зауважте, що у разі переповнення розрядної сітки результат додавання в інверсному коді інший, ніж в доповняльному коді, однак якщо додавання коректне, то він не залежить від вибору коду. У випадку додавання чисел з різними знакам результат завжди за модулем менший від найбільшого з доданків, а отже переповнення розрядної сітки відсутнє.

Існує два основних способи виявлення переповнення розрядної сітки. **Перший з них полягає в порівнянні знаків доданків та їхньої суми**. Якщо знаки доданків однакові, а знак їхньої суми інший, то це є ознакою переповнення (див. попередні два приклади). У випадку різних знаків доданків переповнення принципово бути не може. **Такий спосіб виявлення переповнення можна здійснювати програмними засобами.**

Другий спосіб використовують на апаратному рівні, тобто безпосередньо на рівні електронних схем. Він базується на використанні так званих модифікованих кодів. **Модифікація кодів полягає у введенні додаткового розряду, який розміщують перед знаковим**. Цей розряд називають розрядом переповнення. Інколи кажуть, що модифікований код містить два знакових розряди. Додатні числа мають у знакових розрядах два нулі, а від'ємні – дві одиниці. **Ознакою переповнення розрядної сітки є наявність у знакових розрядах суми різних цифр**.

Приклад 2.16:

A = +27d = +11011b → $00.11011b_{інв.мод}$ → $00.11011b_{доп.мод}$

B = -19d = -10011b → $\underline{11.01100b_{інв.мод}}$ → $\underline{11.01101b_{доп.мод}}$

A + **B** = $100.00111b_{інв.мод}$ $100.01000b_{доп.мод} = 0.01000b_{пр}$ = =+8d

$\underline{\text{└———→1}}$

$00.01000b_{інв.мод} = +8d$

– в інверсному та доповняльному кодах результат коректний.

Приклад 2.17.

A = -20d = -10100b → $11.01011b_{інв.мод}$ → $11.01100b_{доп.мод}$

B = -18d = -10010b → $\underline{11.01101b_{інв.мод}}$ → $\underline{11.01110b_{доп.мод}}$

A + **B** = $110.11000b_{інв.мод}$ $110.11010b_{доп.мод}$ = некоректний результат

$\underline{\text{└———→1}}$

$10.11001b_{інв}$ = некоректний результат

Недоліком модифікованих кодів є вдвічі менший діапазон чисел, які можна подати за допомогою фіксованого числа розрядів.

2.10. Висновки до розділів 1, 2

Порівнявши способи подання чисел у різних кодах, можна зробити ряд висновків:

1. У прямому коді множення та ділення виконуються простіше, ніж в інших кодах, однак він незручний для виконання операції алгебричного додавання, оскільки обчислювальний пристрій ЕОМ повинен вміти як додавати, так і віднімати.

2. Переведення чисел з прямого в інверсний легко реалізувати, оскільки числа зберігаються в лінійці тригерів (так званий регістр), кожен з яких має як звичайний, так і інверсний вихід. Недоліком інверсного коду є необхідність виконання додаткової операції циклічного перенесення.

3. У доповняльному коді простіше, ніж в інших виконується операція алгебричного додавання, оскільки відсутня операція циклічного перенесення. У доповняльному коді нуль має єдине подання.

4. Зміщений код має ту перевагу, що значення числа лінійно зростає зі збільшенням коду подання цього числа. Це дозволяє порівнювати числа шляхом порівняння кодів подання цих чисел на відміну від інверсного чи доповняльного кодів, де порівняння чисел здійснюють шляхом визначення знаку різниці між цими числами. Арифметичні операції в зміщеному коді потребують додаткової операції інверсії знакового розряду суми, що є його недоліком.

Подання 8-розрядних цілих чисел, у прямому, інверсному, доповняльному та зміщеному кодах демонструє табл. 2.1. Для чисел, що мають іншу розрядність (в т.ч. цілої та дробової частин), відповідну таблицю кодів цих чисел можна отримати за аналогією.

Таблиця 2.1. Значення цілих чисел в прямому, інверсному, доповняльному та зміщеному кодах.

Двійкові розряди	Значення числа:					
	беззнакове двійкове число	прямий код	інверсний код	доповняль-ний код	зміщений код	
					з додатним нулем	з від'ємним нулем
11111111	+255	-127	-0	-1	+127	+128
11111110	+254	-126	-1	-2	+126	+127
11111101	+253	-125	-2	-3	+125	+126
- - -	- - -	- - -	- - -	- - -	- - -	- - -
10000001	+130	-2	-125	-126	+2	+3
10000001	+129	-1	-126	-127	+1	+2
10000000	+128	-0	-127	-128	+0	+1
01111111	+127	+127	+127	+127	-1	-0
01111110	+126	+126	+126	+126	-2	-1
01111101	+125	+125	+125	+125	-3	-2
- - -	- - -	- - -	- - -	- - -	- - -	- - -
00000010	+2	+2	+2	+2	-126	-125
00000001	+1	+1	+1	+1	-127	-126
00000000	+0	+0	+0	+0	-128	-127

Розділ 3. ДВІЙКОВІ ЧИСЛА З РУХОМОЮ КОМОЮ

3.1. Подання цілих чисел в комп'ютері

Величини і числа, з якими ми стикаємося як в повсякденній практиці, так і в наукових, теоретичних чи інженерних обчисленнях, належать до дуже широкого числового діапазону. Зазвичай, як приклад, приводять величину маси електрона ($9\cdot10^{-28}$ г) та Сонця ($2\cdot10^{33}$ г). Діапазон цих величин перевищує 10^{60}. Якщо перше з цих чисел містить 28 знаків після коми, то друге навпаки – 33 знаки до коми. З іншого боку ці числа є досить наближеними, однак точне їхнє значення нам невідоме, і принципово не може бути відоме. Такі числа записують з певною точністю, такою, яка необхідна для тих чи інших обчислень. Однак навіть такі числа, які нам відомі з величезною точністю, як наприклад, число π чи e – основа натурального логарифму, ми зазвичай використовуємо в наближеному, округленому варіанті.

Деякі числа ми можемо обчислити точно, наприклад цілі числа. До них зокрема належить факторіал натурального числа $n!=n\cdot(n-1)\cdot(n-2)\cdot\ldots\cdot2\cdot1$, або числа Фібоначчі $\varphi(n)=\varphi(n-1)+\varphi(n-2)$, де $\varphi(1)=\varphi(2)=1$, величина яких дуже швидко зростає зі збільшенням n.

Цифрова обчислювальна машина може оперувати тільки зі скінченними числами, оскільки використовує скінченну кількість двійкових розрядів для їх подання. Це означає, що існує максимальне та мінімальне значення цілого числа, яке може бути подане в комп'ютері за допомогою $\boldsymbol{n}$ розрядів. Діапазон залежить від способу кодування числа та типу змінних в тій, чи іншій мовах програмування.

Таблиця 3.1. Діапазони цілих чисел для різних способів кодування.

Код	Діапазон
прямий	$0\ldots+(2^{\boldsymbol{n}}-1)$
інверсний	$-(2^{\boldsymbol{n}-1}-1)\ldots-0,+0\ldots+(2^{\boldsymbol{n}-1}-1)$
доповняльний	$-(2^{\boldsymbol{n}-1})\ldots0\ldots+(2^{\boldsymbol{n}-1}-1)$
зміщений (+0)	$-(2^{\boldsymbol{n}-1})\ldots0\ldots+(2^{\boldsymbol{n}-1}-1)$

Як правило, для подання цілих додатних чисел використовують прямий код, а для подання як додатних, так і від'ємних – доповняльний або зміщений коди. Будь-яка мова програмування має свої зарезервовані типи для подання цілих чисел. Для мов програмування Pascal та C++ вони показані в табл. 3.2.

3.2. Способи подання дійсних чисел та їхні похибки

Між двома несуміжними цілими числами завжди міститься скінченна кількість цілих чисел та нескінченна кількість дійсних чисел. Це означає, що дійсні числа можна подати в комп'ютері тільки в наближеному вигляді, з певною похибкою, яка не перевищує ваги наймолодшого розряду у поданні числа. Розрізняють абсолютну та відносну похибки подання.

Таблиця 3.2. Діапазони подання цілих чисел для зарезервованих типів в мовах Pascal та C++

Кількість розрядів числа ***n***	Кількість байт пам'яті для збе-рігання змінної	Тип змінної	Мова програ-мування	Діапазон чисел
8	1	Shortint	Pascal	-128...+127
		Byte	Pascal	0...+255
16	2	Integer	Pascal	-32768...+32767
		Word	Pascal	0...+65535
		unsigned short int	C++	0...+65535
		short int	C++	-32768...+32767
32	4	Longint	Pascal	-2147483648...+2147483647
		unsigned long int	C++	0...+4294967295
		long int	C++	-2147483648...+2147483647

Абсолютною похибкою подання називають модуль різниці між значенням числа **A** та значенням його комп'ютерного подання $\mathbf{A_{comp}}$:

$$\Delta = \left| A - A_{comp} \right|. \tag{3.1}$$

Відносною похибкою подання числа **A** називають відношення абсолютної похибки подання до величини цього числа:

$$\delta = \frac{\Delta}{A} \approx \frac{\Delta}{A_{comp}}. \tag{3.2}$$

Існує ряд способів подання дійсних чисел: з фіксованою комою, за допомогою масштабованих коефіцієнтів та за допомогою чисел з рухомою комою.

3.2.1. Подання дійсних чисел у форматі з фіксованою комою

Такий спосіб передбачає **фіксоване положення коми** серед виділених розрядів. Це означає, що для подання цілої частини числа виділяють фіксовану кількість розрядів ***n***, а для подання дробової частини числа – також фіксовану кількість розрядів ***m***.

$$\mathbf{A} = a_{n-1}a_{n-2}...a_1a_0, a_{-1}...a_{-m}. \tag{3.3}$$

Такі числа можна подати в діапазоні $0...+(2^{n}-2^{-m})$ для прямого коду та в діапазоні $-2^{n-1}...+(2^{n-1}-2^{-m})$ для доповняльного. Абсолютна похибка подання не перевищує величини 2^{-m} і не залежить від величини числа. Відносна ж похибка залежить від величини числа. Для максимального за модулем числа 2^{n-1} відносна похибка становить $2^{-m}/2^{n-1} = 2^{-(n+m-1)}$, а для найменшого за модулем, відмінного від нуля числа 2^{-m}, ця похибка рівна $2^{-m}/2^{-m} = 1$ або 100%. Слід зауважити, що числа, які за модулем менше 2^{-m} подають у вигляді нуля, який називають **машинним нулем**. Отже, **машинний нуль – це числа, які за абсолютною величиною не перевищують абсолютної похибки подання.**

Перевагами такого способу подання є:

а) простота подання коду;
б) простота виконання арифметичних операцій над числами;
в) можливість використання прямого, інверсного чи доповняльного кодів;
г) рівномірність подання, яка полягає в тому, що різниця між двома сусідніми числами є величина постійна для всіх чисел у межах діапазону і рівна 2^{-m}.

Недоліками фіксованої коми є те, що:

а) відносна похибка подання чисел неоднакова для різних чисел: для максимальних за модулем чисел вона мінімальна, а для чисел, близьких до нуля – максимальна і сягає 100%;
б) для подання дуже великих чисел необхідно нарощувати кількість цілих розрядів тоді, як значення дробових розрядів стають несуттєвими порівняно з величиною цих чисел. Наприклад, масу автомобіля для більшості випадків недоцільно записувати з точністю до грамів чи більш того до міліграмів;
в) у випадку виконання операції множення, коли множники набагато більші від одиниці, чи ділення, коли дільник дуже близький до нуля, можливе переповнення розрядної сітки (вихід за межі діапазону подання).

Формат фіксованої коми використовувався в ЕОМ перших поколінь. Сучасні обчислювальні пристрої його застосовують тільки тоді, коли числа, якими оперують, належать до вузького динамічного діапазону, наприклад у деяких спеціалізованих вимірювальних пристроях, або тоді, коли необхідно забезпечити однакову похибку в межах заданого діапазону чисел та рівномірний розподіл чисел у ньому.

3.2.2. Метод масштабованих коефіцієнтів

Цей метод також застосовувався в перших поколіннях ЕОМ. Зміст його полягає в тому, що число подають у вигляді добутку масштабного коефіцієнту на число, записане в нормальній формі:

$$\mathbf{A} = \mathbf{К}_{масшт} \cdot \mathbf{A}_{норм}, \qquad \text{де } 0 \le |\mathbf{A}_{норм}| < 1. \qquad (3.4)$$

Числа, записані в нормальній формі, завжди менші від одиниці, тому їхня сума, різниця та добуток також будуть в нормальній формі, а отже питання полягає тільки в тому, щоб узгодити масштабні коефіцієнти. (Власне кажучи, сума двох чисел, записаних у нормальній формі може перевищувати 1, однак вона завжди менше 2). Оскільки метод масштабованих коефіцієнтів є частковим випадком чисел з рухомою комою, в сучасних обчислювальних пристроях він не використовується.

3.3. Принципи подання чисел з рухомою комою

Будь-яке дійсне число в десятковій системі числення можна записати в так званій експоненціальній формі:

$$\mathbf{A} = \boldsymbol{M} \times \mathbf{10}^{\boldsymbol{e}}, \qquad (3.5)$$

де $\boldsymbol{M}$ називають мантисою, а $\boldsymbol{e}$ – експонентою. Якщо мантиса – від'ємне чи додатне **дійсне** число, то експонента завжди **ціле** число. Наприклад:

$$2012 = 201{,}2 \cdot 10^{1} = \ldots = 2{,}012 \cdot 10^{3} = 0{,}2012 \cdot 10^{4}$$
$$-0{,}6875 = -6{,}875 \cdot 10^{-1} = -68{,}75 \cdot 10^{-2} = \ldots = -6875 \cdot 10^{-4}.$$

Існує нескінченна кількість записів того чи іншого числа. Саме тому розрізняють так звану нормальну та нормалізовану форми запису числа.

3.3.1. Нормальна та нормалізована мантиси

Нормальною називається така форма запису, в якій мантиса задовольняє умові $0 \le |M| < 1$. Оскільки нормальних форм також можна виділити нескінченну кількість, то виділяють ще одну форму запису – **нормалізовану, для якої мантиса задовольняє умові $1 \le |M| < q$** (q – основа системи числення). Для кожного числа існує тільки одна нормалізована форма запису.

Записуючи число в експоненціальній нормалізованій формі зазвичай опускають основу експоненти, замінюючи її символом E: $\mathbf{A} = \pm M \mathrm{E} \pm e$, наприклад: 2012 = +2,012E+3; -0,6875 = -6,875E-1 тощо.

Числа, подані в двійковій системі числення також можна записати в експоненціальній формі аналогічно, як і в десятковій:

$$\mathbf{A} = M \times 2^{e} = M\mathrm{b} \times 10\mathrm{b}^{e\mathrm{b}}. \qquad (3.6)$$

(У записі слід враховувати, що 10b = 2d, або просто 2; мантиса та експонента можуть бути записані як у двійковій, так і десятковій системах числення). Наприклад:

$$2012\mathrm{d} = 11111011100\mathrm{b} = 111110111\mathrm{b}\cdot 10\mathrm{b}^{10\mathrm{b}} =$$
$$= 11111011{,}1\mathrm{b}\cdot 10\mathrm{b}^{11\mathrm{b}} = \ldots = 1{,}11110111\mathrm{b}\cdot 10\mathrm{b}^{1010\mathrm{b}} =$$
$$= 11111011{,}1\mathrm{b}\cdot 2\mathrm{d}^{3\mathrm{b}} = \ldots = 1{,}11110111\mathrm{b}\cdot 2\mathrm{d}^{10\mathrm{d}},$$
$$-0{,}6875\mathrm{d} = -0{,}1011\mathrm{b} = -1{,}011\mathrm{b}\cdot 10\mathrm{b}^{-1\mathrm{b}} = \ldots = -1011\mathrm{b}\cdot 10\mathrm{b}^{-100\mathrm{b}} = -1011\mathrm{b}\cdot 2\mathrm{d}^{-4\mathrm{d}}.$$

Для двійкової системи числення також розрізняють нормальну ($0 \le |M| < 1$) та нормалізовану ($1 \le |M| < 2 = 10\mathrm{b}$) форми запису. Нормальна мантиса двійкових чисел в експоненціальній формі завжди починається з нуля, за яким слідує кома, а потім набір цифр "0" та "1", які відповідають значенню цієї мантиси. Нормалізована двійкова мантиса завжди починається з "1", за якою слідує кома, а потім набір чисел, які являють собою залишок від мантиси ($M' = M - 1$). Наприклад, мантиси 0,001010b, 0,10101011b і 0,000000001b є нормальними, а мантиси 1,1010b, 1,1111111b та 1,0000001b – нормалізованими. Окремо слід зауважити, що нормалізована мантиса завжди більше нуля, а тому не може бути використана для його подання.

3.3.2. Загальний формат двійкових чисел з рухомою комою

Дійсне число, записане в експоненціальній двійковій формі можна записати в комп'ютері шляхом подання мантиси за допомогою n двійкових розрядів та експоненти за допомогою m розрядів. Ще один додатковий розряд необхідно виділити для запису знаку числа.

Комп'ютерне подання дійсного числа в експоненціальній двійковій формі за допомогою скінченної кількості двійкових розрядів називають двійковим числом з рухомою комою (Floating Point Number – FPN). Числа з рухомою комою – це особливий тип чисел, який немає математичних аналогів.

Зазвичай, всі числа з рухомою комою мають загальний формат, показаний на рис. 3.1:

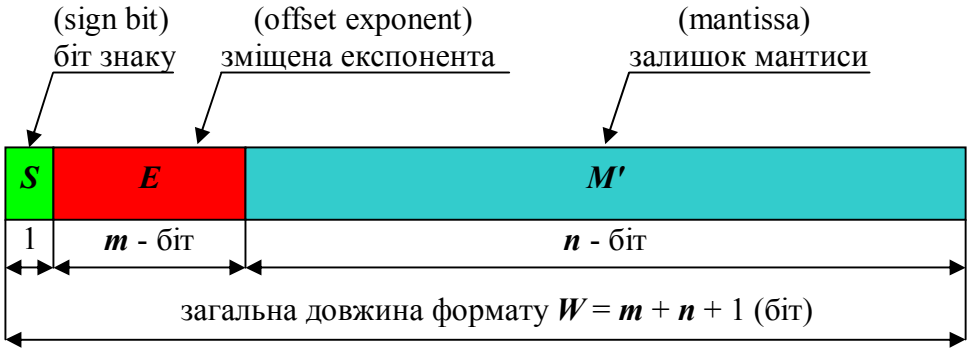


Рисунок 3.1. Загальний формат чисел з рухомою комою

Розрізняють три поля різної довжини: ***S***, ***E*** та ***M'***:
S – біт знаку числа (***S*** = 0 – додатні числа, ***S*** = 1 – від'ємні числа);
E – зміщена експонента. Оскільки експонента ***e*** в формулі (3.6) може набувати як додатних, так і від'ємних значень, то використовують зміщену експоненту ***E***, яка являє собою значення ***e*** в зміщеному коді з від'ємним нулем (зміщене на величину $2^{m-1} - 1$). Отже:

$$E = e + bias = e + 2^{m-1} - 1. \qquad (3.7)$$

Наприклад, якщо для подання експоненти використовують 8 розрядів (***m*** = 8), а експонента в формулі (3.6) становить ***e*** = -9d = -00001001b, то до неї слід додати зміщення, яке рівне $(2^7-1) = 127d = 01111111b$. Таким чином, ***E*** = -9d + 127d = +118d = 01110110b. (Фактично до значення експоненти ***e*** в двійковому коді необхідно додати число, яке складається із нуля в старшому розряді та ***m***-1 одиниць в молодших розрядах).

Мінімальному значення зміщеної експоненти $E_{min} = 000....001b$ відповідає значення $e_{min} = E_{min} - bias = 1 - (2^{m-1} - 1) = -2^{m-1}+2$, а максимальному значенню $E_{max} = 111...110b = 2^m - 2$ – значення $e_{max} = E_{max} - bias = (2^m - 2) - (2^{m-1} - 1) = 2^{m-1} - 1$. Якщо ***m*** = 8, то E_{min} =00000001b (e_{min} =-126d), а E_{max} = 11111110b (e_{max} = +127d). Значення ***E*** = 111...111b використовують для запису нескінченно великих чисел та так званих не-чисел (NaN), а значення ***E*** = 000...000b – для запису так званих денормалізованих чисел. Тому ці значення ***E*** є не описуються формулою (3.7) і є винятковими.

M' – залишок від мантиси. Оскільки нормалізована мантиса завжди починається з одиниці, за якою слідує дробова кома, то недоцільно використовувати додатковий розряд для запису цієї одиниці. Саме тому в поданні числа використовують дробову частину мантиси, а одиниця з комою передбачається неявно. Таким чином, для того, щоб утворити мантису, яка відповідає формулі (3.6), необхідно до ***M'*** додати цю одиницю враховуючи те, що ***M'*** в поданні числа зображено за допомогою ***n*** – розрядного числа:

$$M = 1,+ M'/2^n. \qquad (3.8)$$

Повна кількість розрядів мантиси ***M*** буде складати ***n*** + 1, а загальна формула для обчислення числа визначатиметься як:

$$A = (-1)^S \cdot (1 + M'/2^n) \cdot 2^{(E - 2^{m-1} + 1)}. \qquad (3.9)$$

3.3.3. Математичний зміст мантиси і експоненти

Кількість розрядів експоненти визначає максимальне та мінімальне число (за модулем), яке може бути подане у заданому форматі.

Максимальні числа визначаються як:

$$A_{max} = \pm M_{max} \times 2^{e\max} = \pm(2 - 2^{-n}) \times 2^{2^{m-1}-1} \approx \pm 2^{2^{m-1}}, \qquad (3.10)$$

а мінімальні числа відповідають:

$$A_{min} = \pm M_{min} \times 2^{e\min} = \pm 1 \times 2^{-(2^{m-1}-2)} \approx \pm 2^{-(2^{m-1}-2)}. \qquad (3.11)$$

Якщо $\boldsymbol{m} = 8$, то $\boldsymbol{M}_{max} \approx \pm 2^{128} \approx \pm 3{,}40 \cdot 10^{38}$, $\boldsymbol{M}_{min} \approx \pm 2^{-126} \approx \pm 1{,}17 \cdot 10^{-38}$. Таким чином, динамічний діапазон чисел становить приблизно $38 + 38 = 76$ порядків, що достатньо для більшості обчислень.

Кількість розрядів мантиси визначає точність, з якою число може бути подане або кількість значущих знаків після коми в десятковому записі числа. Можна приблизно оцінити кількість значущих десяткових цифр подання за такою формулою (тут враховано неявну одиницю старшого розряду, як один додатковий розряд):

$$k = (n+1) \cdot \lg 2 \approx 0{,}3 \cdot (n+1). \qquad (3.12)$$

Це означає, що для подання одного десяткового розряду необхідно приблизно 3,32 двійкових розряди ($\log_2 10$). Якщо $\boldsymbol{n} = 23$, тоді $\boldsymbol{k} \approx 7{,}22$ або 7 значущих (significant) десяткових цифр у записі числа.

3.3.4. Похибки подання чисел з рухомою комою

Максимальна абсолютна похибка (3.1) подання визначається вагою найменшого розряду та показником експоненти.

$$\Delta = 2^{-n} \cdot 2^{(E-2^{m-1}+1)} = 2^{-n} \cdot 2^{e}. \qquad (3.13)$$

У межах однакового значення експоненти абсолютна похибка не перевищує $2^{\boldsymbol{e}} \cdot 2^{-\boldsymbol{n}}$. Це означає, що всі числа з однаковими експонентами $\boldsymbol{e}$ рівномірно розподілені в межах інтервалу $[2^{\boldsymbol{e}} \ldots 2^{\boldsymbol{e}+1})$. Однак, у разі збільшення чи зменшення експоненти, абсолютна похибка відповідно буде зростати чи спадати.

Відносна похибка подання визначається за такою формулою:

$$\delta = \frac{\Delta}{A} \approx \frac{2^{-n} \cdot 2^{(E-2^{m-1}+1)}}{(1 + M'/2^{n}) \cdot 2^{(E-2^{m-1}+1)}} = \frac{2^{-n}}{M}. \qquad (3.14)$$

Оскільки $1 \le \boldsymbol{M} < 2$, то відносна похибка не перевищує $2^{-\boldsymbol{n}}$, і в межах всього діапазону подання змінюється не більш ніж вдвічі. Для $\boldsymbol{n} = 23$, максимальна $\delta \approx 1{,}19 \cdot 10^{-7}$.

3.4. Стандарт IEEE754

Стандарт IEEE Std 754-1985, який в оригіналі має назву IEEE Standard for Binary Floating-Point Arithmetic [12] було розроблено з метою стандартизації форматів **двійкових чисел з рухомою комою**, оскільки до його впровадження кожен із виробників процесорів і програмного забезпечення використовував

власний формат таких чисел. Це створювало труднощі з перенесенням програмного забезпечення, оскільки в багатьох випадках арифметичні дії виконувались неправильно, або з великою похибкою. Більшість сучасних процесорів (в тому числі Intel, SPARC, JVM та ін.) містять команди над числами з рухомою комою, які відповідають цьому стандарту.

Стандарт IEEE Std 754-1985 визначає правила:

а) подання нормалізованих додатних та від'ємних чисел з рухомою комою;
б) подання денормалізованих додатних та від'ємних чисел з рухомою комою;
в) подання нульових чисел;
г) подання спеціальної величини "нескінченність" ("infinity");
д) подання спеціальної величини "не число" ("NaN");
е) округлення чисел (4 режими),
а також ряд винятків із перерахованих правил.

3.4.1. Формати чисел з рухомою комою стандарту IEEE754-1985

Стандарт IEEE Std 754-1985 передбачає 4 формати подання чисел з рухомою комою (див. табл. 3.3).

Таблиця 3.3. Рекомендовані формати двійкових чисел з рухомою комою стандарту IEEE754-1985

Формат	***W*** – загальна довжина, біт	***m*** – довжина експоненти, біт	***n*** – довжина залишку мантиси, біт
одинарна точність (single-precision)	32	8	23
одинарна розширена точність (single-extended precision)	≥ 43	≥ 11	≥ 31
подвійна точність (double-precision)	64	11	52
подвійна розширена точність (double-extended precision)	≥ 79 (80)	≥ 15 (15)	≥ 63 (64)

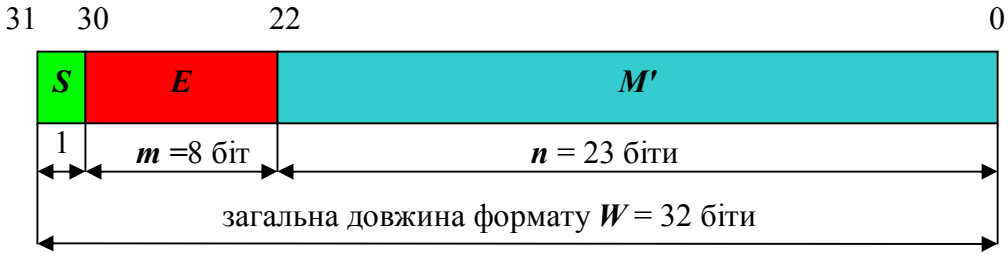


Рисунок 3.2. 32-бітний формат чисел з рухомою комою

Формат з одинарною розширеною точністю дуже рідко використовують на практиці, а з формат з подвійною розширеною точністю зазвичай має довжину 80 біт (15 біт експоненти та 64 біти мантиси плюс один знаковий біт). Основне застосування в техніці та програмуванні, як правило, отримали формати 32 та

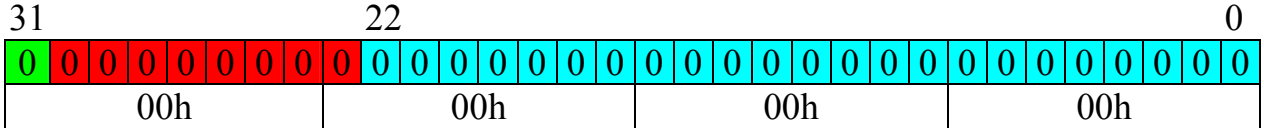
64 біти. Розглянемо для прикладу формат 32 біти (рис.3.2). Всі правила подання чисел у цьому форматі легко поширюються і на будь-який інший формат.

3.4.2. Нормалізовані та денормалізовані числа

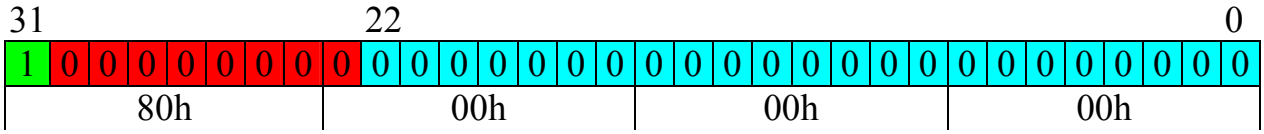
Нормалізовані числа в форматі IEEE754 подають згідно з формулою (3.9). Однак, як випливає з цієї формули, подати число нуль неможливо. Тому стандарт IEEE754 передбачає спеціальний набір бітів для подання числа нуль.

Якщо всі біти мантиси та експоненти (за винятком знакового *S* біту) рівні нулю, то таке число вважають нулем.

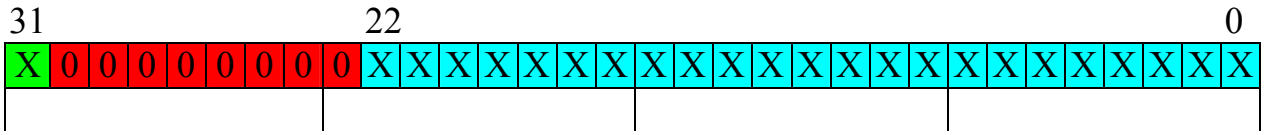
Число 00 00 00 00h вважають числом +0.



Число 80 00 00 00h вважають числом -0.



Числа, в яких всі біти експоненти рівні нулю, а мантиса має ненульовий набір бітів вважають денормалізованими числами:



Денормалізовані числа розраховують згідно з формулою:

$$A = (-1)^S \cdot (0,+M'/2^n) \cdot 2^{(-2^{m-1}+2)}. \qquad (3.15)$$

Денормалізовані числа відрізняються від нормалізованих тим, що **в полі мантиси** денормалізованих чисел **записують дробову частину мантиси числа в нормальному вигляді без уявного нуля та коми після нього**, а для нормалізованих чисел мантиса повинна бути нормалізована. Наприклад, нормальна мантиса 0,0011010101..., а в полі денормалізованої мантиси буде записане число ***M'*** = 0011010101... **Нуль є частковим випадком денормалізованих чисел**.

Денормалізовані числа розширюють діапазон чисел, оскільки мінімальне (за модулем) число матиме значення (***M'*** = 00..01 = 1)

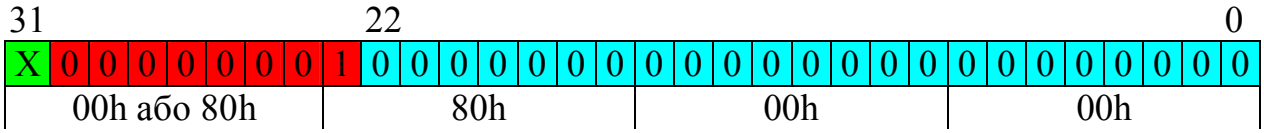
$$A = (-1)^S \cdot (1/2^n) \cdot 2^{(-2^{m-1}+2)} = \pm 2^{(-2^{m-1}+2-n)} = \pm 2^{-(2^{m-1}+n-2)}. \qquad (3.16)$$

Слід зауважити, що для денормалізованих чисел експоненту приймають рівною e_{min} для нормалізованих чисел, а не розраховують згідно з формулою $e_{min} = E_{min}$ - ***bias***. Таким чином, експонента денормалізованих чисел завжди рівна $e_{min} = 1 - e_{max} = -2^{m-1}+2$.

3.4.3. Границі діапазонів нормалізованих та денормалізованих чисел

Мінімальне (за модулем) нормалізоване число:

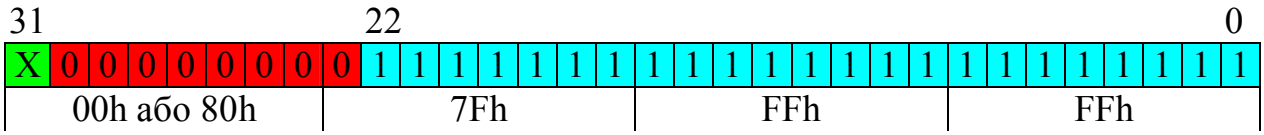
00(80) 80 00 00h = $\pm 2^{-126} \times (1+0/2^{23}) = \pm 2^{-126} \approx \pm 1{,}17549435\text{E-}38$



Максимальне (за модулем) денормалізоване число:

00(80) 7F FF FFh = $\pm 2^{-126} \times (0 + (2^{23}-1)/2^{23}) = \pm(2^{-126} - 2^{-149}) \approx$

$\approx \pm 1{,}17549421\text{E-}38$



Отже, мінімальне нормалізоване число межує з максимальним денормалізованим числом.

Денормалізованим числам притаманний той недолік, що відносна похибка подання таких чисел неоднакова для різних чисел: для максимальних за модулем чисел вона не перевищує похибки нормалізованих чисел, а для чисел, близьких до нуля, – максимальна і сягає 100%. Крім того, денормалізовані числа дуже важко реалізують на апаратному рівні, для цього використовують програмні реалізації, які працюють набагато повільніше. В сучасних процесорах опрацювання денормалізованих чисел відбувається в десятки разів повільніше, ніж нормалізованих чисел.

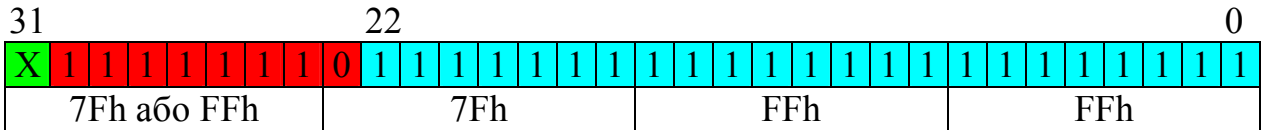
У багатьох випадках денормалізовані числа просто ігнорують, тобто вважають їх машинним нулем. У версії стандарту IEEE754-2008 такі числа називаються "**субнормальними**" (subnormal numbers), тобто числа, що менше від "нормальних".

3.4.4. Діапазон чисел одинарної точності

Динамічний діапазон визначає кількість порядків чисел, які можна подати за допомогою вибраного формату подання. Його обчислюють в такий спосіб:

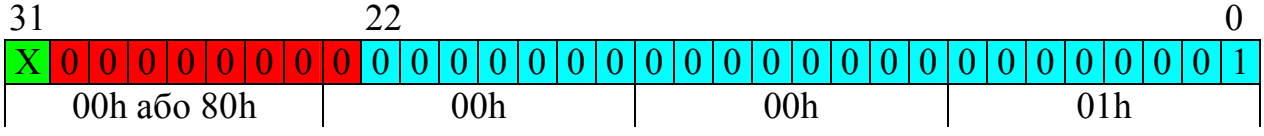
Максимальне (за модулем) нормалізоване число

7F(FF) 7F FF FFh = $\pm 2^{127} \times (2\text{-}2^{-23}) \approx \pm 3{,}40282347\text{E+}38$



Мінімальне (за модулем) денормалізоване число

00(80) 00 00 01h = $\pm 2^{-126} \times 2^{-23} = \pm 2^{-149} \approx \pm 1{,}40129846\text{E-}45$



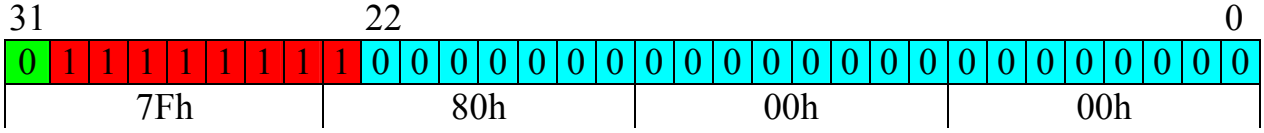
Отже динамічний діапазон чисел одинарної точності (32-бітний формат подання) становить 38 + 45 = 83 десяткові порядки.

3.4.5. Особливі випадки подання

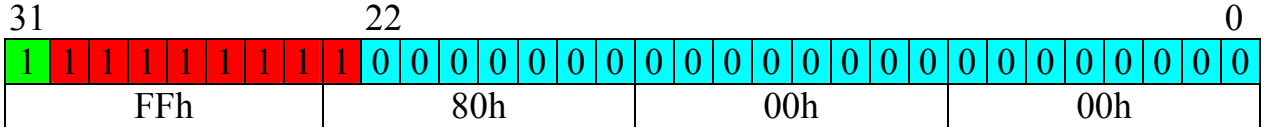
Нескінченність.

Число вважають нескінченно великим, якщо всі розряди експоненти рівні одиниці, а всі розряди мантиси рівні нулю.

Число 7F 80 00 00h вважають числом +∞.



Число FF 80 00 00h вважають числом -∞.



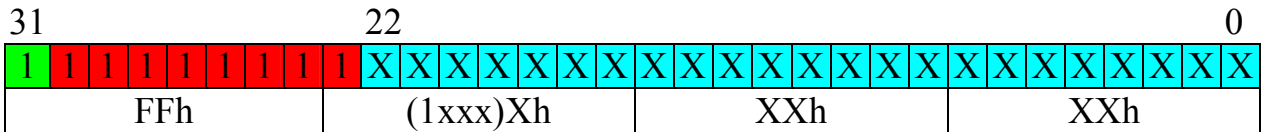
Отримати нескінченність можна у випадку округлення, переповнення розрядної сітки та у разі ділення на нуль. Слід пам'ятати, що:

$$\frac{x}{0} = \begin{cases} +\infty, & x > 0 \\ \text{NaN}, & x = 0 \\ -\infty, & x < 0 \end{cases}.$$

Невизначеність, або "не числа" (NaN)

NaN – це абревіатура від **not a number**. NaN є результатом арифметичних операцій, якщо під час виконання відбулася помилка. NaN подають як число, у якого всі біти експоненти рівні одиниці, а мантиса – ненульовий набір бітів. Більш детально NaN специфіковано стандартом IEEE754-2008, який розрізняє два види **NaN**: qNaN та sNaN (див. п. 4.6, або [9]).

Число FF (1xxx)X XX XXh вважають "не числом" – NaN.



NaN можна отримати в таких випадках:

а) +∞ + (-∞) = NaN; б) ±0 × ±∞ = NaN; в) ±0 / ±0 = NaN;

г) ±∞ / ±∞= NaN; д) $\sqrt{x}$ = NaN,де x < 0.

"Не числа", як це не парадоксально звучить, насправді є числами IEEE754-математики, особливого розділу математики, який описує арифметичні дії з комп'ютерним поданням дійсних чисел у вигляді чисел з рухомою комою.

3.4.6. 64-бітний (Double) формат IEEE754.

Всі розглянуті вище правила та винятки застосовані також і до інших форматів чисел IEEE754. Для формату подвійної точності 64 біти маємо (рис.3.3):

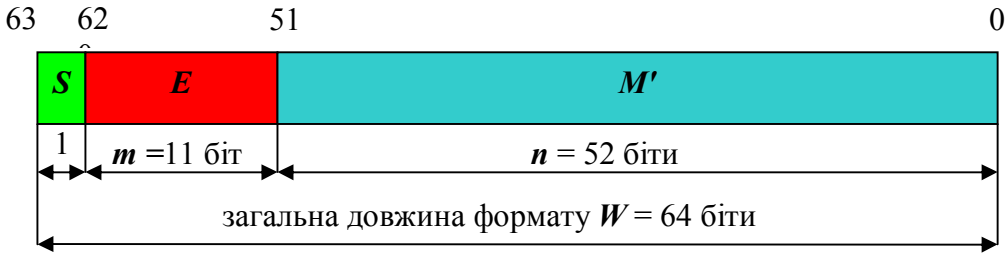


Рисунок 3.3. 64-бітний формат чисел з рухомою комою.

Зміщення експоненти 1023d (011 1111 1111b).

Діапазон експоненти від E = 00000000001b (-1022d) до E = 11111111110b (+1023d).

Максимальне та мінімальне (за модулем) нормалізовані числа:

$$M_{max} \approx \pm 2^{1024} \approx \pm 1{,}7 \cdot 10^{308},\ M_{min} \approx \pm 2^{-1022} \approx \pm 2{,}4 \cdot 10^{-308}.$$

Мінімальне (за модулем) денормалізоване число:

$$7F(FF)\ F0\ 00\ 00\ 00\ 00\ 00\ 01h = \pm 2^{-1022} \times 2^{-52} = \pm 2^{-1074} \approx \pm 2{,}0 \cdot 10^{-323}$$

Динамічний діапазон чисел становить приблизно 308 + 323 = 631 порядок.

Кількість значущих десяткових цифр: $k = (52+1) \cdot \lg 2 \approx 15{,}95 \approx 16$.

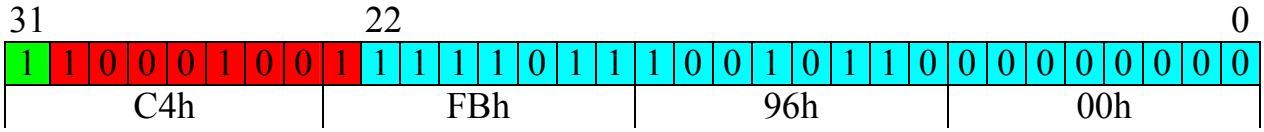
3.5. Правила подання чисел в стандарті IEEE754

Для отримання подання дійсного числа в стандарті IEEE754 необхідно:

1) Перевести модуль заданого числа в двійкову систему числення;
2) Нормалізувати двійкове число, тобто записати його у вигляді $M \times 2^e$, де мантиса $1 \le M < 2$, експоненту e записати в десятковій системі числення;
3) Округлити мантису M до n розрядів після коми згідно з п. 3.6.3. (Якщо кількість дробових розрядів мантиси менше n, то доповнити мантису справа нулями);
4) Сформувати залишок мантиси M', відкинувши від нормалізованої мантиси M зліва "1,";
5) Додати до експоненти зміщення 2^m-1 у десятковій системі числення;
6) Отриману зміщену експоненту E записати у двійковій системі;
7) Враховуючи знак S заданого числа, записати його подання в пам'яті ЕОМ.
8) Отриманий результат записати в шістнадцятірковій системі числення.

Приклад 3.1. Записати число -2012,6875d у форматі одинарної (32 біти) точності (m = 8, n = 23).

1) -2012,6875d = -11111011100,1011b.
2) -11111011100,1011b = -1,11110111001011b×2d^{10d},
3) M = 1,11110111001011000000000,
   ↓ відкинути
4) M' = 11110111001011000000000,
5-6) E = 10d + (2^7 - 1) = 137d = 10001001b,
7) S = 1,



8) -2012,6875d = C4 FB 96 00h

Приклад 3.2 Записати число 1000,3d у форматі подвійної (64 біти) точності. (***m*** = 11, ***n*** = 52).

1) 1000,3d = 1111101000,0001100110011...b

2) 1111101000,0100110011001...b = 1,1111010000100110011001...b×2d^{9d},

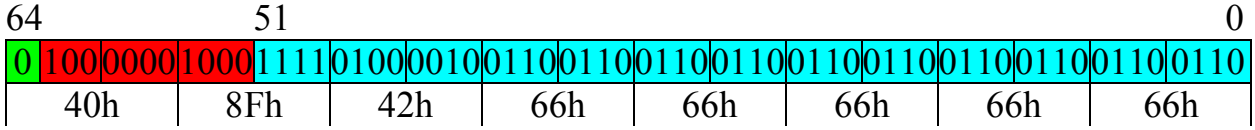
3) ***M*** = 1,1111010000100110011001100110011001100110011001100110,

↓ відкинути

4) ***M'*** = 1111010000100110011001100110011001100110011001100110,

5-6) ***E*** = 9d + (2^{10} - 1) = 1032d = 100000001000b,

7)) ***S*** = 0,



8) 1000,1d = 40 8F 42 66 66 66 66 66h

3.6. Операції над двійковими числами в стандарті IEEE745

В якості прикладу розглянемо основні принципи виконання арифметичних дій додавання (віднімання) і множення (ділення). Для здійснення операцій додавання та віднімання над двійковими числами з рухомою комою необхідно попередньо вирівняти порядки (експоненти) операндів, що вимагає зсуву положення розділової коми у мантиси. Операції множення і ділення вирівнювання порядків операндів не вимагають. Під час виконання операцій можуть виникнути такі особливі ситуації:

1. **Переповнення порядку**. Додатна експонента результату перевищує максимальне значення, яке передбачено форматом. Такий результат може трактуватися, залежно від апаратної реалізації, як sNaN, величина +∞ чи -∞.
2. **Недоповнення порядку**. Від'ємна експонента результату менше мінімального значення, яке передбачено форматом. Отриманий результат може трактуватися, залежно від апаратної реалізації, як sNaN, величина +0 чи -0 (машинний нуль).
3. **Втрата вагомості мантиси**. В процесі вирівнювання порядків мантиса зсувається настільки сильно вправо, що її старший, відмінний від нуля розряд, виходить за межі розрядної сітки. Як буде показано далі, в цьому випадку потрібна певна форма округлення. Зазвичай втрата вагомості порядку не є сигналізуючою, тобто результатом такої дії може бути qNaN, однак у більшості випадків результат додавання (віднімання) рівний більшому за модулем операнду.

Особливістю здійснення арифметичних операцій над двійковими числами з рухомою комою є те, що такі операції не потребують переведення операндів у десяткову систему числення, а здійснюються у форматі двійкових чисел з рухомою комою.

3.6.1. Додавання і віднімання

Алгоритми виконання операцій додавання і віднімання у форматі з рухомою комою складніші, ніж аналогічні алгоритми для чисел з фіксованою комою. Це пов'язано з необхідністю вирівнювання порядків операндів. Алгоритм додавання (віднімання) (рис. 3.4) складається з чотирьох основних етапів.

1. Перевірка на специфічні значення операндів;
2. Зсув мантис для вирівнювання порядків;
3. Додавання або віднімання мантис;
4. Нормалізація результату.

Специфічними значеннями операндів є нуль, ±∞, sNaN та qNaN. На схемі алгоритму (рис. 3.4) показано тільки ті випадки, коли операнди рівні нулю чи ±∞, оскільки результатом додавання (віднімання) чисел, одним із операндів яких є sNaN чи qNaN, завжди буде qNaN.

Якщо одним з операндів є нуль, то результат додавання (віднімання) буде рівний іншому операнду (або його від'ємному значенню). Якщо один з операндів ±∞, а другий – скінченне число, то результат операції також є ±∞. Однак, якщо другий операнд також ±∞, то слід врахувати, що +∞ + (+∞) = +∞ та -∞ + (-∞) = -∞, а +∞ - (+∞) = qNaN, також -∞ + (-∞) = qNaN.

Зсув мантис здійснюють з метою узгодити порядки операндів. Очевидно, що коли порядки операндів відрізняються один від одного, то над їхніми мантисами не можна здійснювати операцій віднімання та додавання. Сказане може бути проілюстроване таким чином:

$$\mathbf{X \pm Y} = (-1)^{S_x} \cdot 2^{E_{x\text{-bias}}} \cdot (1+M'_x/2^n) \pm (-1)^{S_y} \cdot 2^{E_{y\text{-bias}}} \cdot (1+M'_y/2^n) =$$
$$= 2^{E_{x\text{-bias}}} \cdot [(-1)^{S_x} \cdot (1+M'_x/2^n) \pm (-1)^{S_y} \cdot 2^{E_{y}-E_x} \cdot (1+M'_y/2^n)]. \qquad (3.17)$$

При цьому користуються таким правилом: експонента результату повинна бути рівною (до операції нормалізації) експоненті більшого за модулем операнду. Винятковими є ті випадки, коли в результаті додавання (віднімання) отримують помилку **переповнення порядку**. Можливі такі випадки експонент операндів: $E_x > E_y$, $E_y > E_x$ та $E_y = E_x$. Якщо $E_x > E_y$, тоді $E_y - E_x < 0$, а вираз $2^{E_y-E_x} \cdot (1+M'_y/2^n)$ у формулі (3.17) фактично означає, що мантису операнду **Y** слід змістити вправо на $E_x - E_y$ розрядів (поділити на $2^{E_x-E_y}$). Якщо ж $E_y > E_x$, то слід змістити вправо на $E_y - E_x$ розрядів мантису операнду **X**.

Підготовлені таким чином мантиси вже можна додавати (віднімати) з врахуванням знаку операндів. Слід однак зауважити, що якщо мантису змістити вправо на n+1 розрядів (n – кількість біт, які передбачені для подання залишку мантиси), то додавання (віднімання) такого операнду не буде впливати на результат виконання операції, оскільки такий зміщений операнд буде еквівалентний нулю. Тому результат операції буде повністю визначатися другим операндом. **Такий випадок називають втратою вагомості мантиси.** Втрата вагомості мантиси не призводить до появи sNaN, тому таку помилку (похибку обчислення) можна проконтролювати тільки за допомогою програмних засобів, попередньо порівнюючи операнди. Певні види АЛП можуть встановлювати відповідний прапорець регістру ознак результату операції (прапорець стану).

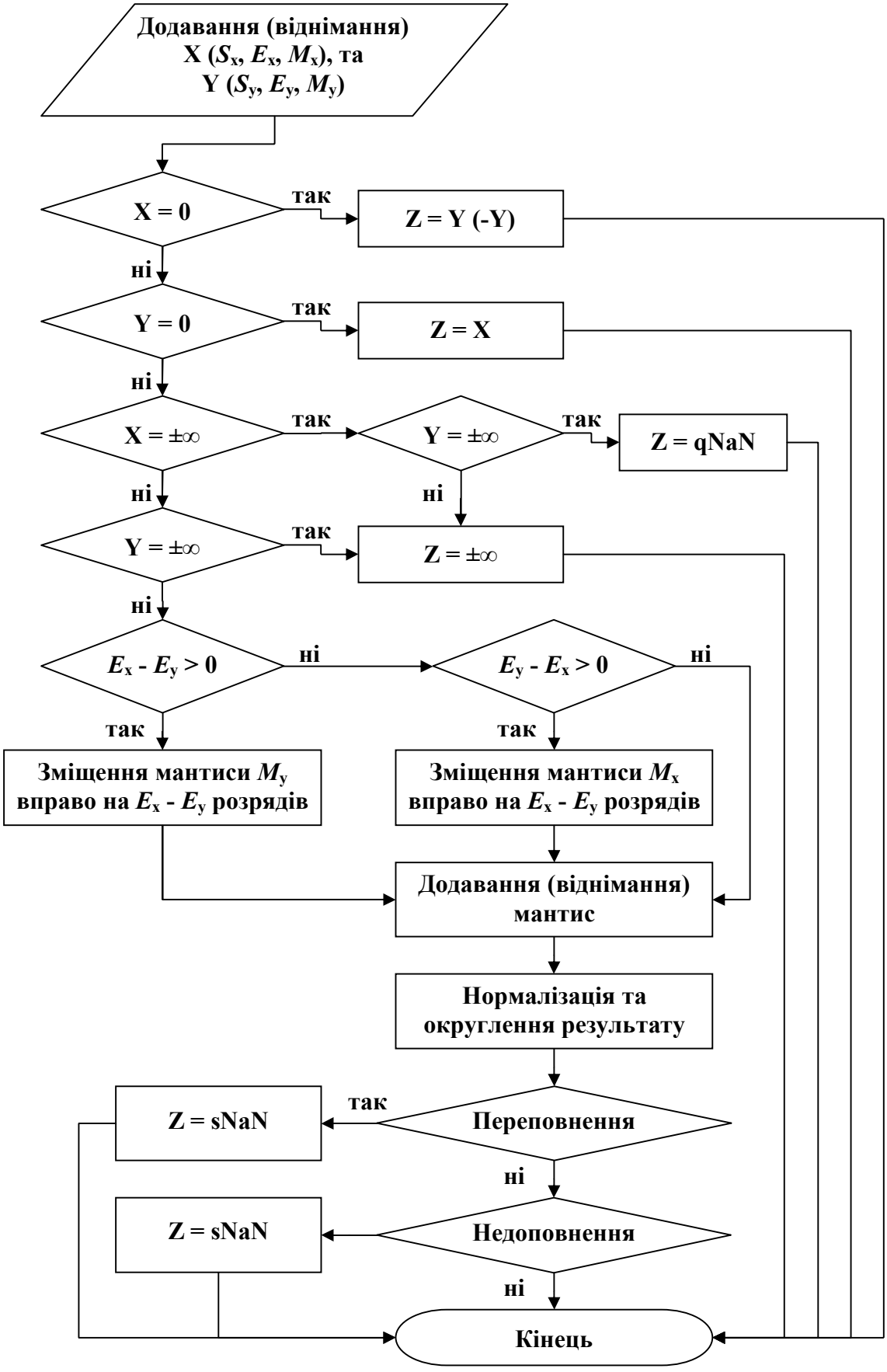


Рисунок 3.4. Алгоритм додавання (віднімання) двійкових чисел з рухомою комою IEEE754.

Додавання (віднімання) мантис після узгодження порядків операндів здійснюють згідно звичайних правил додавання (віднімання) двійкових чисел. Після округлення слід нормалізувати результат, оскільки сума мантис може набувати значень в інтервалі (1...4), а їх різниця – в інтервалі [0...2). Нормалізація результату фактично зводиться до приведення результуючої мантиси $\boldsymbol{M}_{\text{рез}}$ в інтервал [1...2). Якщо $2 \le \boldsymbol{M}_{\text{рез}} < 4$, то зменшуючи $\boldsymbol{M}_{\text{рез}}$ вдвічі (еквівалентно зміщенню коми на один розряд вліво) та одночасно збільшуючи експоненту на 1 ми отримаємо нормалізований результат. Аналогічно, якщо $0 < \boldsymbol{M}_{\text{рез}} < 1$, то виконуючи раз за разом збільшення мантиси вдвічі (зміщення коми на один розряд вправо) з одночасним зменшенням експоненти на 1, через ряд кроків ми отримаємо нормалізований результат. Виняткова ситуація може бути в тому випадку, коли результуюча мантиса рівна нулю. Тоді результатом операції також є нуль (+0).

Оскільки в процесі нормалізації експонента може бути збільшена на 1, то можливий випадок, коли отримана в такий спосіб результуюча експонента перевищує максимальне значення для заданого формату подання. **Такий випадок називають переповненням порядку (overflow)**. Отриманий таким чином результат не може бути подано в межах заданого формату подання у вигляді числа, і його інтерпретують як sNaN.

У разі, коли результуюча експонента менше від мінімального значення для заданого формату маємо **помилку недоповнення порядку (underflow)**, яка також буде інтерпретована як sNaN.

Таким чином, qNaN характеризує помилку, яку можна передбачити за значеннями вхідних операндів, а sNaN – помилка, яка виникає тільки в процесі виконання операції та не може бути передбачена. Більше того, якщо операція над числами (операндами) у вибраному форматі подання призводить до появи sNaN, то записавши ці числа у форматі з вищою точністю (та діапазоном) ми отримаємо скінченне число.

3.6.2. Множення і ділення

Виконання операцій множення та ділення у форматі з рухомою комою здійснюють дещо простіше, ніж додавання чи віднімання таких чисел. Алгоритми множення (рис. 3.5) та ділення (рис. 3.6) також містять чотири основні етапи.

1. Перевірка на специфічні значення операндів;
2. Додавання (віднімання) експонент з врахуванням зміщення;
3. Множення (ділення) мантис;
4. Нормалізація результату.

Специфічними значеннями операндів також є нуль, ±∞, sNaN та qNaN. У будь-яких випадках множення (ділення), коли одним із операндів sNaN чи qNaN, результат завжди буде qNaN (такого роду виняткові випадки на схемах алгоритмів множення (рис. 3.5) та ділення (рис. 3.6) не показані.

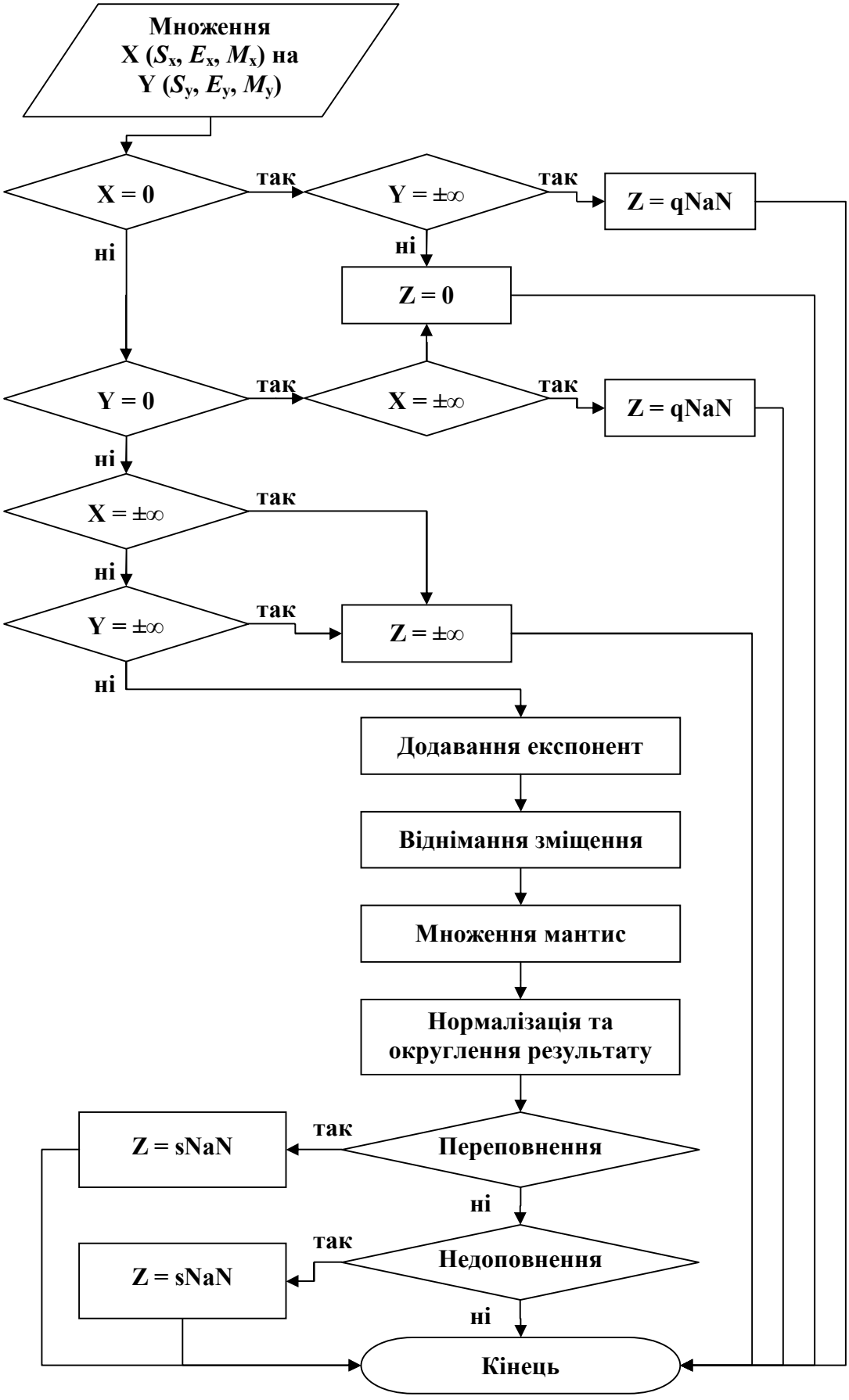


Рисунок 3.5. Алгоритм множення двійкових чисел з рухомою комою IEEE754.

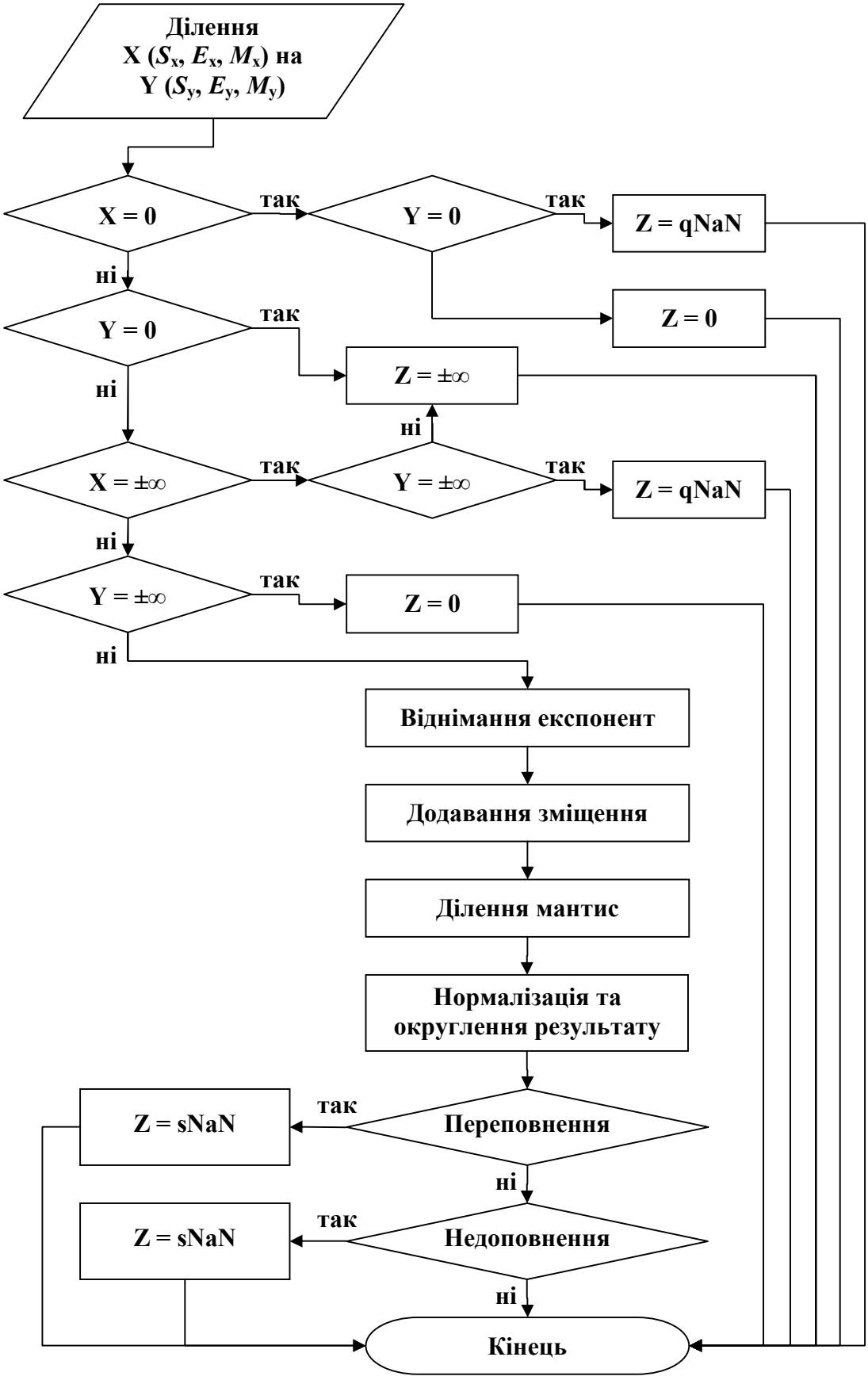


Рисунок 3.6. Алгоритм ділення двійкових чисел з рухомою комою IEEE754.

До виняткових випадків множення також належать:

а) $0 \cdot A = A \cdot 0 = 0$, (A – скінченне число);

б) $0 \cdot \pm\infty = \pm\infty \cdot 0 =$ qNaN;

в) $\pm\infty \cdot \pm\infty = \pm\infty$.

Множення чисел можна продемонструвати таким чином:

$$X \times Y = [(-1)^{Sx} \cdot 2^{Ex\text{-bias}} \cdot (1+M'_x/2^n)] \times [(-1)^{Sy} \cdot 2^{Ey\text{-bias}} \cdot (1+M'_y/2^n)] =$$
$$= (-1)^{Sx+Sy} \cdot 2^{Ex\text{-bias}+Ey\text{-bias}} \cdot [(1+M'_x/2^n) \times (1+M'_y/2^n)]. \qquad (3.18)$$

Знак добутку однозначно визначається знаками співмножників.

Експонента добутку є сума експонент множників. Оскільки експоненти подані в зміщеному коді з від'ємним нулем, то додавання таких експонент слід виконувати згідно правил додавання в такому коді (див. розділ 3.4.2), а саме: здійснюють звичайне додавання зміщених експонент, інвертують старший розряд, та збільшують результат на одиницю молодшого розряду. Фактично це означає, що **до результату додавання експонент**

$$(E_x - \text{bias}) + (E_y - \text{bias}) = (E_x + E_y - \text{bias}) - \text{bias}$$

слід додати зміщення для того, щоб отримати зміщений код суми експонент. Зміщення в коді з від'ємним нулем є число, у якого старший розряд рівний нулю, а всі інші розряди встановлено в "1". Тому додавання такого числа еквівалентно інверсії старшого розряду суми експонент множників з наступним збільшенням отриманого результату на 1.

Множення мантис здійснюють за тим же алгоритмом, що і множення цілих чисел у прямому коді, тобто фактично перемножують числа без знаку, а потім добутку приписують знак "плюс" чи "мінус" залежно від знаків співмножників. Добуток мантис має розрядність вдвічі більшу, ніж кожен із співмножників, тому зайві молодші розряди відкидають, враховуючи правила округлення. Добуток нормалізованих мантис набуває значення з інтервалу чисел [1...4). Після того, як добуток мантис отримано, результат нормалізують і округлюють. Ці операції виконують аналогічно, як і у разі додавання чи віднімання. Необхідно враховувати, що в процесі додавання експонент та нормалізації може виникнути помилка переповнення або недоповнення порядку, що призводить до появи sNaN.

Специфічними (винятковими) значеннями діленого і дільника є:

а) $0 / A = 0$, (A – число не рівне нулю, в тому числі $\pm\infty$);

б) $A / \pm\infty = 0$, (A – скінченне число);

в) $0 / 0 =$ qNaN;

г) $\pm\infty / \pm\infty =$ qNaN.

Ділення чисел можна продемонструвати таким чином:

$$X / Y = [(-1)^{Sx} \cdot 2^{Ex\text{-bias}} \cdot (1+M'_x/2^n)] / [(-1)^{Sy} \cdot 2^{Ey\text{-bias}} \cdot (1+M'_y/2^n)] =$$
$$= (-1)^{Sx-Sy} \cdot 2^{Ex-Ey} \cdot [(1+M'_x/2^n) / (1+M'_y/2^n)]. \qquad (3.19)$$

Знак частки однозначно визначають знаки діленого і дільника.

Експонента частки є різниця експонент множників. Віднімання таких експонент слід виконувати згідно з правилами віднімання в зміщеному коді з від'ємним нулем (див. розділ 3.4.2), а саме: до зменшуваного додають від'ємник, у якого всі розряди інвертовані. Старший розряд отриманої суми слід інвертувати. Фактично це означає, що від результату віднімання експонент

$$(E_x - \text{bias}) - (E_y - \text{bias}) = (E_x - E_y)$$

слід відняти зміщення для отримання зміщеного коду частки.

Ділення мантис здійснюються за алгоритмом ділення цілих чисел у прямому коді, тобто мантиси вважають додатними (або без знаку), потім частці приписують знак "плюс" чи "мінус" залежно від знаків діленого та дільника. Ділення здійснюють до тих пір, доки не буде отримано необхідну кількість значущих розрядів частки. Частка нормалізованих мантис набуває значення з інтервалу чисел (0,5...2). Отриману частку мантис нормалізують і округлюють. Ці операції виконують аналогічно, як і у разі додавання, віднімання чи множення. В результаті віднімання експонент та нормалізації також може виникнути помилка переповнення або недоповнення порядку, що призводить до появи sNaN.

3.6.3. Округлення

Результат будь-якої операції над мантисами операндів зазвичай формується блоком FPU (Floating Point Unit), який має вищу розрядність, ніж передбачено форматом подання. Тому для формування результату необхідно в той чи інший спосіб виконати його округлення. Стандартом IEEE754-1985 передбачено чотири альтернативних підходи до виконання округлення:

1) округлення до найближчого числа, яке можна подати у заданому форматі;
2) округлення до +∞;
3) округлення до -∞;
4) округлення до нуля.

Розглянемо ці способи більш детально.

Округлення до найближчого числа полягає у виборі такого наближеного скінченого числа в заданому форматі подання, яке найближче до точного результату обчислення. Наприклад, нехай числа

1) 0011010,00b = 26,00d,
2) 0011100,00b = 28,00d,
3) 0011011,00b = 27,00d,
4) 0011010,10b = 26,50d,
5) 0011011,10b = 27,50d

необхідно округлити до цілих (округлити дробові розряди).

Округлення таких чисел в десятковій системі числення не становить проблеми. Вона виникає тоді, коли будемо округлювати в двійковій системі числення. Якщо щодо перших трьох чисел округлення очевидне, то щодо двох останніх – ні. Згідно зі способом округлення число 0011010,10b = 26,50d є однаково близьке як до числа 0011010b = 26d, так і до числа 0011011b = 27d, а

число 0011011,10b = 27,50d – однаково близьке до чисел 0011011b = 27d та 0011100b = 28d. Звісно, можна округлювати так, як це здійснюють у десятковій системі числення, тобто округлювати до більшого (за модулем) числа, однак ретельний аналіз обчислень з округленнями в двійковій системі числення показує, що раз за разом помилка округлення буде тільки накопичуватись. Це означає, що виконання великої кількості обрахунків з таким способом округленням може призвести до недостовірного результату. Таку **похибку округлення, яка має здатність накопичуватись, називають зміщеною помилкою (похибкою) округлення**. Один з варіантів незміщеного округлення – випадковим чином вибирати округлення до більшого або до меншого. Тоді в середньому результат ряду обчислень буде незміщеним. Однак такий метод не дає можливості отримати повторювані результати обчислень для однієї і тієї ж процедури з одними і тими ж вхідними даними (обчислювальна процедура стає недетермінованою). Тому стандарт IEEE754 регламентує інший підхід, а саме: **Якщо існують два скінчені числа, однаково близькі до результату обчислення, то вибирають те з них, у якого наймолодший розряд мантиси рівний 0**. Таким чином, число 0011010,10b = 26,50d слід округлити до числа 0011010b = 26d, а число 0011011,10b = 27,50d – до числа 0011100b = 28d. Фактично це означає, що числа округлюють до парних значень, тому **такий спосіб округлення також називають округленням до найближчого парного**.

Виняток складає випадок, коли нескінченний точний результат за модулем є величина порядку $2^{Emax}\cdot(2-2^{-n})$ (найбільше за модулем скінчене число для заданого формату подання). У цьому разі число буде округлене до нескінченності зі збереженням знаку результату.

Стандартом IEEE754 передбачено, що округлення до найближчого парного числа повинно здійснюватись за замовчуванням, тобто у всіх випадках, якщо не вибрано іншого способу округлення.

Округлення до +∞ чи до -∞ застосовують для реалізації методики обчислень, відомої як "арифметика інтервальних обчислень", яка передбачає визначення верхньої та нижньої меж значень результату. Якщо здійснюють округлення до +∞, то вибирають таке наближене скінченне число (або +∞) у заданому форматі подання, яке одночасно найближче до точного результату обчислень і **не менше** від нього. Це означає, що число +1011100,1b буде округлене до +1011101b, а число -1011100,1b – до -1011100b. Подібним чином округлюють до -∞, тобто вибирають таке наближене скінченне число (або -∞) у заданому форматі подання, яке одночасно найближче до точного результату обчислення і **не більше** від нього. Таким чином, число +1011100,1b буде округлене до +1011100b, а число -1011100,1b – до -1011101b.

Округлення до нуля полягає у виборі такого наближеного скінченного числа в заданому форматі подання, яке одночасно найближче до точного результату обчислення і **за модулем не більше** від нього. Цей спосіб округлення фактично передбачає обтинання результату – всі лишні розряди просто відкидають. За таким способом округлення отриманий результат за модулем завжди буде меншим або рівним точному значенню, тому у разі виконання тривалих обчислень утворюється зміщення в бік менших значень (до

нуля). Накопичення такої помилки округлення може бути значно вищим, ніж у випадку, розглянутому вище (див. округлення до найближчого), оскільки зміщена помилка округлення до нуля виникає при виконанні кожної операції, а не тільки в особливих випадках.

3.6.4. Приклади додавання (віднімання) двійкових чисел з рухомою комою

Приклад 3.3. Додати двійкові числа з рухомою комою формату 32 біти $\boldsymbol{X}$ = 5BFF5000h та $\boldsymbol{Y}$ = 58F6D800h застосовуючи округлення результату до найближчого числа.

1) Запишемо операнди в бінарному коді, визначимо їхні знаки, зміщені експоненти та повні мантиси (враховуючи, що $\boldsymbol{M} = 1, + \boldsymbol{M'}/2^{23}$):
   $\boldsymbol{X}$ = 0101 1011 1111 1111 0101 0000 0000 0000.
   $\boldsymbol{Y}$ = 0101 1000 1111 0110 1101 1000 0000 0000.
   $\boldsymbol{S}_x$ =0 , $\boldsymbol{E}_x$ = 10110111, $\boldsymbol{M}_x$ = 1,11111110101000000000000.
   $\boldsymbol{S}_y$ =0 , $\boldsymbol{E}_y$ = 10110001, $\boldsymbol{M}_y$ = 1,11101101101100000000000.

2) Оскільки знаки операндів однакові та рівні "плюс", то сума двох додатних чисел є число додатне. Тому $\boldsymbol{S}_{x+y} = 0$.

3) $\boldsymbol{E}_x > \boldsymbol{E}_y$, оскільки 10110111b > 10110001b, тому зміщена експонента суми буде рівна $\boldsymbol{E}_x$, а мантису $\boldsymbol{M}_y$ слід змістити вправо на 10110111b - - 10110001b = 110b = 6d розрядів.

4) До мантиси $\boldsymbol{M}_x$ додаємо зміщену на 6 розрядів вправо мантису $\boldsymbol{M}_y$:

$$\begin{aligned} \boldsymbol{M}_x &= 1,11111110101000000000000 \\ +\boldsymbol{M}_y/2^6 &= +\underline{0,00000111101101101100000000000} \\ & \quad 10,00000110010101101100000000000 \end{aligned}$$

5) Нормалізуємо результат. Оскільки результуюча мантиса > 2, то її слід зменшити вдвічі, змістивши кому на один розряд вліво, одночасно збільшивши зміщену експоненту суми на 1. Тому $\boldsymbol{E}_{x+y}$ = 10111000, а
   $\boldsymbol{M}_{x+y}$ = 1,00000011001010110110000000000.

6) Формуємо залишок мантиси суми відкидаючи зліва "1," та округлюючи її до 23-х розрядів:
   $\boldsymbol{M}_{x+y}$ = 1,00000011001010110110000000000
   відкинути ↑ ↑округлити
В даному випадку спосіб округлення не має значення. Отже
   $\boldsymbol{M'}_{x+y}$ = 00000011001010110110000

7) Записуємо отриманий результат додавання:
   $\underline{\boldsymbol{X} + \boldsymbol{Y} = 0101\ 1100\ 0000\ 0001\ 1001\ 0101\ 1011\ 0000 = 5C0195B0h}$.

Оскільки віднімання можна замінити операцією додавання числа з протилежним знаком, то розглянемо приклад додавання двох чисел з різними знаками.

Приклад 3.4. Додати двійкові числа з рухомою комою формату 32 біти X = 9F18E26Dh та Y = 28B66666h застосовуючи округлення результату до найближчого числа.

1) Запишемо операнди в бінарному коді, визначимо їхні знаки, зміщені експоненти та повні мантиси (враховуючи, що $M = 1, + M'/2^{23}$):

   X = 1001 1111 0001 1000 1110 0010 0110 1101.
   Y = 0010 1000 1011 0110 0110 0110 0110 0110.
   S_x =1 , E_x = 00111110, M_x = 1,00110001110001001101101.
   S_y =0 , E_y = 01010001, M_y = 1,01101100110011001100110.

2) Знаки операндів різні, тому результат матиме знак більшого за модулем операнду. $E_x < E_y$, оскільки 00111110b < 01010001b. Сума буде мати знак більшого за модулем числа Y, тобто $S_{x+y} = 0$ (сума додатна).

3) Оскільки $E_x < E_y$, то зміщена експонента суми буде рівна E_y, а мантису M_x слід змістити вправо на 01010001b - 00111110b = 10011b = 19d розрядів і відняти від мантиси M_y.

4) Від мантиси M_y віднімаємо зміщену на 19 розрядів вправо мантису M_x:

M_y = 1,01101100110011001100110
$- M_x/2^{19}$ = - 0,00000000000000000010011000111000100110110 1
1,01101100110011001010010111000111011001 0011

5) Нормалізуємо результат. Оскільки результуюча мантиса вже є нормалізованою, то
   M_{x+y} = 1,01101100110011001010010111000111011001 0011

6) Формуємо залишок мантиси суми відкидаючи зліва "1," та округлюючи її до 23-х розрядів:
   M_{x+y} = 1,01101100110011001010010111000111011001 0011
   відкинути ↑ ↑ округлити
Залишок, який треба округлити, передбачає округлення таким чином, що наймолодший розряд округленої мантиси матиме значення "1", тому:
   M'_{x+y} = 01101100110011001010011

7) Записуємо отриманий результат додавання:
   $X + Y$ = 0010 1000 1011 0110 0110 0110 0101 0011 = 28B66653h.

3.6.5 Приклади множення (ділення) двійкових чисел з рухомою комою

Приклад 3.5. Перемножити двійкові числа з рухомою комою формату 32 біти X = 41D80000h та Y = C8A02800h застосовуючи округлення результату до найближчого числа.

1) Запишемо операнди в бінарному коді, визначимо їхні знаки, зміщені експоненти та повні мантиси (враховуючи, що $M = 1, + M'/2^{23}$):

   X = 0100 0001 1101 1000 0000 0000 0000 0000.

$\boldsymbol{Y}$ = 1100 1000 1010 0000 0010 1000 0000 0000.
$\boldsymbol{S}_{\mathbf{x}}$ =0 , $\boldsymbol{E}_{\mathbf{x}}$ = 10000011, $\boldsymbol{M}_{\mathbf{x}}$ = 1,10110000000000000000000.
$\boldsymbol{S}_{\mathbf{y}}$ =1 , $\boldsymbol{E}_{\mathbf{y}}$ = 10010001, $\boldsymbol{M}_{\mathbf{y}}$ = 1,01000000010100000000000.

2) Оскільки знаки операндів різні, результат буде мати знак "мінус", тому $\boldsymbol{S}_{\mathbf{x}\times\mathbf{y}} = 1$.

3) Визначимо зміщену експоненту добутку. Додаємо експоненти в зміщеному коді з від'ємним нулем (див. п. 2.8):

$$\begin{array}{rcr} \boldsymbol{E}_{\mathbf{x}} & = & 10000011 \\ +\boldsymbol{E}_{\mathbf{y}} & = & + \ \underline{10010001} \\ & & 100010100 \end{array}$$

відкинути ↓↑ інвертувати розряд суми

$$\begin{array}{rcr} & & 10010100 \\ & & + \ \underline{\qquad\qquad 1} \\ \boldsymbol{E}_{\mathbf{x}\times\mathbf{y}} & = & 10010101 \end{array}$$

4) Перемножимо мантиси. Оскільки крайні праві нулі мантис можна відкинути, маємо:

$$\begin{array}{rcr} \boldsymbol{M}_{\mathbf{y}} & = & 1{,}010000000101 \\ \times \boldsymbol{M}_{\mathbf{x}} & = & \times \ \underline{\qquad\qquad 1{,}1011} \\ & & 1010000000101 \\ & & 1010000000101 \\ & & 1010000000101 \\ & & \underline{1010000000101\qquad\qquad} \\ \boldsymbol{M}_{\mathbf{x}\times\mathbf{y}} & = & 10{,}00011100100001111 \end{array}$$

5) Нормалізуємо результат. Оскільки результуюча мантиса > 2, то її слід зменшити вдвічі, змістивши кому на один розряд вліво, одночасно збільшивши зміщену експоненту добутку на 1. Тому $\boldsymbol{E}_{\mathbf{x}\times\mathbf{y}}$ = 10010110, а
$\boldsymbol{M}_{\mathbf{x}\times\mathbf{y}}$ = 1,00001110010000111000000.

6) Формуємо залишок мантиси добутку, відкидаючи зліва "1,". Округлення в даному випадку не потрібне, оскільки точний результат має менше 23-х розрядів.
$\boldsymbol{M'}_{\mathbf{x}\times\mathbf{y}}$ = 00001110010000111000000

7) Записуємо отриманий результат множення:
$\boldsymbol{X} \times \boldsymbol{Y}$ = 1100 1011 0000 0111 0010 0001 1100 0000 = CB0721C0h.

Приклад 3.6. Поділити двійкові числа з рухомою комою формату 32 біти $\boldsymbol{X}$ = C1200000h на $\boldsymbol{Y}$ = 40400000h застосовуючи округлення результату до найближчого числа.

1) Запишемо операнди в бінарному коді, визначимо їхні знаки, зміщені експоненти та повні мантиси (враховуючи, що $\boldsymbol{M} = 1, + \boldsymbol{M'}/2^{23}$):

$\boldsymbol{X}$ = 1100 0001 0010 0000 0000 0000 0000 0000.

Y = 0100 0000 0100 0000 0000 0000 0000 0000.
S_x =1 , E_x = 10000010, M_x = 1,01000000000000000000000.
S_y =0 , E_y = 10000000, M_y = 1,10000000000000000000000.

2) Оскільки знаки операндів різні, результат матиме знак "мінус", тому $S_{x/y} = 1$.

3) Визначимо зміщену експоненту частки. Віднімання $E_x - E_y$ замінюємо додаванням $E_x + (-E_y)$, однак в E_y всі розряди інвертуємо (див. п. 3.6.2):

E_x = 10000010 10000010
$-E_y$ = - 10000000 + 01111111
100000001
відкинути ↓↑ інвертувати розряд суми
$E_{x/y}$ = 10000001

4) Поділимо мантису M_x на M_y. Оскільки крайні праві нулі мантис можна відкинути, то змістивши коми маємо:

M_x / M_y = 10,100....|11
1 1 |0,110101010101....
1 00
11
100
11
100 і т.д.

5) Нормалізуємо результат. Оскільки результуюча мантиса < 1, її слід збільшити вдвічі, змістивши кому на один розряд вправо, одночасно зменшивши зміщену експоненту частки на 1. Тому $E_{x/y}$ = 10000000, а
$M_{x/y}$ = 1,10101010101010101010101.....

6) Формуємо залишок мантиси частки відкидаючи зліва "1," та округлюючи результат до 23-х розрядів.

$M_{x/y}$ = 1,10101010101010101010101010101...
відкинути ↑ ↑ округлити

Залишок, який треба округлити, передбачає округлення таким чином, що наймолодший розряд округленої мантиси збереже значення "1", тому:

$M'_{x/y}$ = 10101010101010101010101

7) Записуємо отриманий результат ділення:

X / Y = 1100 0000 0101 0101 0101 0101 0101 0101 = C0555555h.

3.7. Реалізація двійкових чисел з рухомою комою в сучасних процесорах

У перших 8-розрядних процесорах арифметику з рухомою комою було реалізовано програмними засобами (підпрограмами операційної системи), однак виконання простої команди додавання (віднімання) чи множення (ділення) потребувало значного часу і сильно сповільнювало роботу процесора. Саме тому в процесі розроблення 16-розрядного процесора Intel 8086 було

вирішено реалізувати цю арифметику на апаратному (схемному) рівні. Результуюча схема вийшла доволі громіздка за технологіями того часу і не могла бути інтеґрована з центральним процесором. Тому її виготовили як окрему мікросхему Intel 8087 і назвали **математичним співпроцесором**.

Співпроцесор забезпечував виконання операцій над числами з рухомою комою і розширював набір команд центрального процесора Intel 8086, оскільки останній міг здійснювати операції тільки з цілими числами. Центральний процесор виконував основну програму, а як тільки треба було виконати команду арифметики з рухомою комою – передавав її виконання співпроцесору.

Починаючи з процесора Intel486DX модуль операцій з рухомою комою було інтеґровано в центральний процесор і названо FPU (Floating Point Unit). Незважаючи на інтеграцію, FPU в процесорах i486 являв собою все той же співпроцесор, однак виконаний на єдиному з центральним процесором кристалі. Тільки починаючи з процесора Pentium моделі MMX блок FPU був по-справжньому інтегрований з центральним процесором. На сьогоднішній день всі сучасні процесори містять інтегрований блок FPU, а деякі з них – навіть декілька таких блоків.

Розділ 4. ДЕСЯТКОВІ ЧИСЛА З РУХОМОЮ КОМОЮ

4.1. Основні недоліки стандарту IEEE754-1985

З моменту затвердження стандарту IEEE754-1985 пройшло вже понад 25 років. За цей час комп'ютерна техніка розвивалась як у напрямі збільшення її швидкодії та обсягів опрацьовуваної інформації, так і в напрямі підвищення надійності та точності обчислень. Одночасно з широким впровадженням стандарту, починаючи з апаратної реалізації математики з рухомою комою у співпроцесорах (Intel 8087) та закінчуючи створенням ефективного програмного забезпечення суперкомп'ютерів та кластерних систем, було виявлено ряд недоліків, які притаманні числам з рухомою комою. Ці недоліки зумовлені тим, що такі числа є наближенням дійсних, реальних чисел, а математику, яка базується на основі стандарту IEEE754-1985, не можна вважати математикою дійсних чисел. Фактично це є математика наближених чисел і саме тому всі обчислення необхідно здійснювати за правилами математичних дій з наближеними числами. Її ще інколи називають IEEE754-математикою. Такі математичні дії повинні передбачати врахування похибки подання чисел та її накопичення в процесі обчислення.

Одним з недоліків IEEE754-математики є обмежений динамічний діапазон чисел з рухомою комою. Це означає, що числа, які більші від $A_{max} = (2 - 2^{-n}) \times 2^{2^{m-1}-1} \approx 2^{2^{m-1}}$ (3.10) та менші від $A_{min} = 2^{-(2^{m-1}+n-2)}$ (3.16) не можуть бути подані за допомогою вибраного формату чисел, який передбачає ***m*** розрядів експоненти та ***n*** розрядів залишку мантиси. Такі числа вважають відповідно машинним нулем або нескінченністю. Якщо в результаті ділення числа ***a*** на число ***b*** було отримано машинний нуль, то результатом операції (***a***/***b***) × 2***b*** також буде машинний нуль, хоча повинно бути число 2***a***. (Фактично маємо випадок порушення асоціативності операції множення).

Другою недоліком є те, що абсолютна похибка подання числа залежить від величини цього числа та визначається згідно (3.13), як $\Delta = 2^{-n} \cdot 2^{(E-2^{m-1}+1)} = 2^{e-n}$, де ***e*** – значення експоненти. Для максимальних значень $e = 2^{m-1} - 1$ абсолютна похибка $\Delta = 2^{(2^{m-1}-1-n)}$ набагато більша від $A_{min} = 2^{-(2^{m-1}+n-1)}$. Якщо до такого числа ***a*** додати число ***b***, яке за величиною менше абсолютної похибки подання числа ***a***, то в результаті отримаємо те саме число ***a***, тобто ***a*** + ***b*** = ***a***. Тут ми маємо випадок, розглянутий в п. 3.6.1, який називають **<u>втратою значимості мантиси</u>**. Таким чином:

$$(\boldsymbol{a} + \boldsymbol{b}) + (-\boldsymbol{a}) = 0 \neq (\boldsymbol{a} + (-\boldsymbol{a})) + \boldsymbol{b} = \boldsymbol{b},$$

а отже маємо невиконання властивості асоціативності додавання. Наприклад:

$$(10^{20} + 1) - 10^{20} = 0 \neq (10^{20} - 10^{20}) + 1 = 1.$$

Третім недоліком є те, що скінченні десяткові дроби в двійковій системі числення зазвичай записуються у вигляді нескінченного дробу, наприклад, 0,3d = 0,0100110011...b. (Виняток складають тільки ті дроби, які можна

записати у вигляді x/y, де y – степінь числа два). Оскільки залишок мантиси має ***n*** розрядів, то такі скінченні в десятковому записі дроби будуть подані наближено, з певною похибкою. Таким чином, виконання простого множення 0,3d × 0,7d = 0,21d насправді здійснюють над наближеними числами і результатом такої операції є наближене число.

Четвертий недолік безпосередньо випливає з правил округлення чисел у стандарті IEEE754. Згідно з цими правилами, результат будь-якої арифметичної операції над IEEE754-числами повинен бути таким, якби ця операція була виконана над точними значеннями цих чисел, а результат округлений до найближчого числа, яке може бути подане в заданому форматі.

Округлення в стандарті (див. п. 3.6.3.) здійснюють не таким способом, як ми привикли у повсякденній практиці. Математично показано, що якщо 0,5d округлювати до 1d (в більшу сторону), то існує набір операцій, за яких похибка округлення буде зростати до нескінченності. Тому в стандарті IEEE754 застосовують правило округлення до парного. Наприклад, 12,5d буде округлене до 12d, а 13,5d – до 14d. У більшості випадків, особливо для чисел подвійної (та вище) точності, похибки округлення є несуттєвими. Однак усі арифметичні дії над числами, які являють собою грошові суми, повинні здійснюватись з точністю до копійки (цента, пенні тощо). Застосування описаного вище округлення може призводити до суттєвих похибок, особливо у випадку нарахування відсотків на банківський депозит або платежів за певну частку послуг згідно з тарифом.

Причиною ще одного недоліку IEEE754-математики є неоднозначне подання нуля, який має два значення: -0 та +0. Таке подання призводить до того, що $-0 < +0$, а отже $1/(-\infty) < 1/(+\infty)$, хоча з точки зору звичайної математики ці величини однакові. Якщо в програмі є цикл, умовою закінчення якого є рівність певної змінної нулю ("звичайний" нуль – це +0), то за певних умов такий цикл буде здійснюватись нескінченну кількість разів, що призведе до "зависання" програми.

Можна навести ще ряд недоліків IEEE754-математики, однак основний висновок, зумовлений їхнім існуванням, полягає в тому, що **врахувати всі похибки за допомогою апаратних засобів ЕОМ неможливо, це можна здійснити тільки шляхом написання досконалого програмного коду.**

Таким чином, програміст, який використовує числа з рухомою комою, повинен мати глибокі знання в області теорії чисел, наближених обчислень, теорії похибок та чітко усвідомлювати до яких результатів може призвести неврахування особливостей здійснення математичних операцій над числами з рухомою комою. Однак не всі програмісти є математиками, а результатами їх помилок часто є фатальні катастрофи. Ряд масштабних аварій, як було доведено, зумовлений винятково неправильним використанням чисел з рухомою комою стандарту IEEE754 [10].

Однією з таких катастроф став вибух ракети "Аріан-5" на 40-й секунді після старту під час її першого запуску у Французькій Гвіані 4 червня 1996 року [19]. Матеріальні втрати, а це крім самої ракети ще чотири супутники "Cluster", оцінюють від 360 до 500 мільйонів доларів. Причиною аварії стало викорис-

тання програмного модуля попередньої ракети "Аріан-4", який відповідав за її вирівнювання та стабілізацію польоту. Цей програмний модуль використовував інший формат чисел, ніж модуль, що відповідав за двигуни. Під час конвертації 64-бітного числа з рухомою комою в 16-бітне ціле число відбулося переповнення розрядної сітки, внаслідок чого бортовий комп'ютер ракети збільшив потужність двигунів і вивів їх на нештатний режим. Ракета отримала неприпустиме аеродинамічне навантаження і почала руйнуватись.

Іншою аварією, яка пов'язана вже з конвертацією десяткових чисел у двійкові, є так звана "Patriot Missile Software Problem" або "Patriot Missile Bug" – помилка системи протиповітряної оборони США "Patriot" під час війни у Перській затоці [20]. І хоча при цьому використовувались числа з фіксованою, а не з рухомою комами, така помилка проявилась би в обох випадках практично однаково.

25 лютого 1991 року база коаліційних сил у Дахрані (Саудівська Аравія) була атакована іракською ракетою Scud B. Зазвичай система Patriot відслідковує пуски ворожих ракет і збиває їх ракетою-перехоплювачем. Помилка виникла в програмному забезпеченні, що відстежує ціль на радарі та обчислює її координати. Для контролю часу використовували 24-бітні числа з фіксованою комою, крок становив 0,1 с. Десяткове число 0,1d, як відомо, не можна точно подати за допомогою скінченної кількості двійкових розрядів, тому час у програмі відрізнявся від реального часу і ця помилка накопичувалася в міру роботи системи. Після 100 годин послідовної роботи різниця становила 0,34 с, що при швидкості ракети 1,7 км/с призвело до розбіжності розрахункового місця цілі та її реального положення приблизно на 0,5 км. У результаті ціль не була перехоплена ракетою Patriot і попала по казармах сил коаліції, внаслідок чого загинуло 28 солдатів США. Цього би не відбулося, якщо би систему регулярно перезавантажували, що було рекомендовано, але не виконано. Цю проблему також можна було вирішити ще під час програмування, якщо в циклі підпрограми, яка відповідає за контроль часу, в якості індексу використовували би не число 0,1d, а ціле, кратне йому число, наприклад 1. Як відомо, цілочисельна арифметика в комп'ютерах є точною, а реальний час завжди можна було б отримати множенням індексу циклу на 0,1d.

Ще одна резонансна катастрофа сталася 23 серпня 1991 року. У Гандсфіорді поблизу берегів Норвегії під час монтування потонула нафтова платформа Sleipner A, що призвело до збитків на 700 мільйонів доларів [21]. Причиною катастрофи вважають неточні розрахунки параметрів конструкції також зумовлені використанням чисел з рухомою комою.

Ці катастрофи належать до тих, причини яких були однозначно ідентифіковані як наслідок некоректних арифметичних дій над числами з рухомою комою. Така ідентифікація досить складна, тому можна тільки здогадуватися у скількох випадках авіакатастроф, пожеж, вибухів тощо, не було виявлено істинних причин аварій.

Чого можна очікувати в майбутньому? Нового Чорнобилю (Фукусіми) через помилки керування атомним реактором? Чи самозапуску ракети з атомною боєголовкою?

Оскільки в основі більшості комп'ютерних розрахунків використовують числа з рухомою комою стандарту IEEE754, на нього покладена надзвичайна відповідальність за достовірність обчислень. Саме тому стандарт 1985 року вимагав удосконалення і нових підходів до підвищення надійності та якості подань реальних чисел.

4.2. Четверний (128-бітний) формат, його переваги та недоліки.

Одним із шляхів вирішення проблеми достовірності обчислень є екстенсивний метод, який полягає у нарощуванні розрядності подання. У 1985 р. вважалося, що 80-бітного формату з надлишком достатньо для реалізації високоточних обчислень. Однак з тих пір, відповідно до закону Мура, швидкодія комп'ютерів та об'єм обчислень зросли в десятки тисяч разів. Одночасно зросли і похибки обчислень. Тому стандарт IEEE754-1985 було доповнено 128-бітним форматом. Це доповнення було офіційно затверджено тільки 2008 року, хоча інженери провідних фірм використовували такий формат дещо раніше. Ними було розроблено ряд нових мікропроцесорів, які застосовували такий формат двійкових чисел з рухомою комою [8]. 128-бітний формат іще називають четверним (quadruple), якщо за основу взяти 32-бітний формат.

Для формату четверної точності 128 бітів маємо:

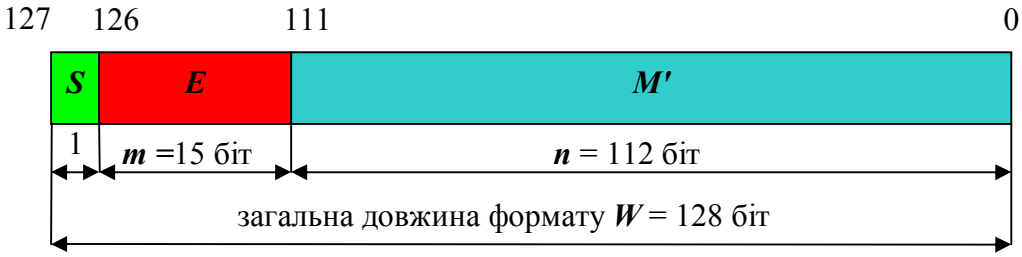


Рисунок 4.1. 128-бітний формат чисел з рухомою комою

Зміщення експоненти 16383d (011 1111 1111 1111b).

Діапазон експоненти:

від E_{min} = 0000 0000 0000 0001b (-16382d)

до E_{max} = 1111 1111 1111 1110b (+16383d).

Нормалізовані числа:

$$A_{max} \approx \pm 2^{16384} \approx \pm 1{,}2 \cdot 10^{4932}, \qquad A_{min} \approx \pm 2^{-16382} \approx \pm 3{,}4 \cdot 10^{-4932}.$$

Мінімальне (за модулем) денормалізоване число

$$\underbrace{\text{7F(FF) F0 00 00 00 00 00 00 00 00 00 00 00 00 00 01h}}_{\text{16 байтів}} =$$

$$= \pm 2^{-16382} \times 2^{-112} = \pm 2^{-16494} \approx \pm 6{,}5 \cdot 10^{-4966}$$

Динамічний діапазон чисел становить приблизно 4932 + 4966 = 9898 порядків. Кількість значущих десяткових цифр подання: $k = (112+1) \cdot \lg 2 \approx 34{,}01 \approx 34$. Принципи подання нуля, нескінченності, не-чисел (NaN) ті ж, що і для одинарного чи подвійного форматів.

Які ж переваги та недоліки дає нам нарощування кількості розрядів подання? На перший погляд основний виграш ми маємо у підвищенні точності обчислень і відповідно меншій похибці, яка купується за збільшення часу виконання арифметичних операцій та об'єму комп'ютерної пам'яті для зберігання даних і результатів обчислень. Мінімальні та максимальні значення такого подання далеко виходять за межі чисел, якими ми оперуємо у повсякденній практиці і які можуть трапитися нам в недалекому майбутньому.

З іншого боку, в IEEE754-математиці бажано використовувати великі формати. Річ у тім, що розходження IEEE754-математики зі звичайною математикою зазвичай виявляється появою дуже великих чисел, до того ж тим більших, чим більший формат подання. Як не дивно, такі розходження легше ідентифікувати. Однак це можливо тільки у разі втручання людини в процес ідентифікації помилок. Зі збільшенням складності та об'ємів обчислень ця ідентифікуюча ознака втрачає своє значення. Отже ми маємо те, з чого починали, і гонитві за довжиною формату немає кінця.

Розробники нового 128-бітного формату запевняють, що його вистачить на найближчих 20 років, однак доволі простий приклад показує ілюзорність таких надій та дає уявлення про те, наскільки можуть бути оманливими результати, здавалося би простих обчислень.

4.3. Приклад Румпа

У 1988 р. проф. Зігфрид Румп, співробітник німецького відділення фірми IBM, досліджуючи алгоритми обчислень з гарантованими межами інтервалів, які містять заздалегідь відомий правильний результат, виявив поліном, який за певного співвідношення значень змінних дає однозначно неправильний результат для всіх стандартних форматів чисел з рухомою комою [12].

$$f = 333{,}75b^6 + a^2(11a^2b^2 - b^6 - 121b^4 - 2) + 5{,}5b^8 + \frac{a}{2b}, \qquad (4.1)$$

де $a = 77617$, $b = 33096$. Використовуючи різні формати чисел з рухомою комою на стандартній для того часу великій обчислювальній системі IBM S/370 для цього поліному було отримано приблизно однакові результати:

32 біти: $f = +1{,}172604$;

64 біти: $f = +1{,}1726039400531786$;

128 біти: $f = +1{,}172603940053178631858834904520 1838$.

Ці результати дозволяють вважати, що надійний результат – приблизно 1,172603, або навіть 1,172603940053. Однак насправді правильний результат (у межах одиниці останньої цифри) відрізняється від продемонстрованих вище не тільки значенням, але навіть і знаком (!):

$$f = -0{,}827396059946821368141165095479816...$$

Причиною такої розбіжності є те, що в даному випадку значення a та b задовольняють простій умові:

$$a^2 = 5{,}5b^2 + 1, \qquad (4.2)$$

а підставивши це значення в (4.1) маємо:

$$f = -5{,}5b^8 - 2 + 5{,}5b^8 + \frac{a}{2b} = \frac{a}{2b} - 2. \qquad (4.3)$$

Число $5{,}5b^8 = 79171113406689613611011347015249428 48$ – 37-ми значне. Це означає, що 34-х значущих десяткових цифр формату 128 біт недостатньо для того, щоби подати це число точно, тому похибка подання числа порядку 1000 набагато перевищує число 2. Таким чином, ці числа перебувають у таких діапазонах подання, здійснення арифметичних дій між числами із яких є некоректним. У результаті виконання дій у форматі 128 біт ми отримуємо

$$f = \frac{a}{2b} = 1{,}17260394005317863185883490452018 38... \qquad (4.4)$$

Цей приклад став класичною ілюстрацією тих проблем, які притаманні сучасним обчисленням з рухомою комою. Обчислення полінома Румпа на комп'ютерах з процесорами Pentium та Sun Sparc дало різний результат залежно від того, яку систему комп'ютерної математики використовували. Наприклад, обчислення полінома Румпа на базі програми, отриманої за допомогою компілятора Fortran 95 фірми Sun Microsystem Inc., дали такі результати:

32 біт: $f = -6.\ 338253 \cdot 10^{29}$;

64 біт: $f = -1.\ 1805916207174113 \cdot 10^{21}$;

128 біт: $f = +1.\ 17260394005317863185883490452018 38$.

Пізніше Румп показав [13], що аналогічні проблеми виникають і у випадку простішого полінома (за аналогічних значень a та b):

$$f = 21b^2 - 2a^2 + 55b^4 - 10a^2b^2 + \frac{a}{2b}. \qquad (4.5)$$

Особливу занепокоєність такого роду результати викликають у зв'язку з широким розповсюдженням за останній час обчислень на базі графічних процесорів (GPU), які для досягнення максимальної продуктивності зазвичай оперують з числами тільки одинарної точності, що різко підвищує ймовірність отримання неправильних результатів. Це не критично для розрахунків, пов'язаних із побудовою графічних зображень, однак графічні процесори дедалі частіше використовують для побудови високопродуктивних паралельних обчислювальних систем, які можуть містити десятки тисяч подібних процесорів, оскільки вони і дешеві, і швидкісні. Найбільш тривожним фактом є те, що практично в усіх математичних пакетах не тільки результат прикладу Румпа є неправильним, але й відсутні будь-які ознаки того, що під час обчислення виникли проблеми. А це означає, що такого роду помилок, які важко помітити, може бути непередбачена кількість.

Одним із способів отримання правильного результату для прикладу Румпа є використання символьних обчислень. Це означає, що в процесі обчислення полінома дійсні числа подають не в традиційному форматі з рухомою комою, а у вигляді раціональних дробів. Тоді поліном Румпа набуває вигляду:

$$f = \frac{33375}{100}b^6 + a^2(11a^2b^2 - b^6 - 121b^4 - 2) + \frac{55}{10}b^8 + \frac{a}{2b}. \qquad (4.6)$$

Якщо для обчислень використовувати саме таку формулу, наприклад у пакеті Maple 8, то отримаємо правильний результат:

$$f = \frac{-54767}{66192} = -0{,}827396059946821... \qquad (4.7)$$

Однак символьні обчислення не завжди можуть вирішити обчислювальну проблему. Якщо у вираз входить функція, яка повертає ірраціональний результат (логарифм, синус, радикал тощо), то її перетворення до дробового вигляду гарантовано призвело би до похибки, і така нефіксована втрата точності стала би причиною отримання невірного результату.

Ще одним способом отримання правильного результату є використання під час обчислень такої розрядності подання даних, яка суттєво перевищує стандартну. Для приклада Румпа 37-ми значне число $5{,}5b^8$ у формулі (4.3) вимагає $37/\lg 2 \approx 37 \cdot 3{,}32 \approx 123$ бітів для його повного подання (див. п. 3.3.3), що дещо більше, ніж 112+1 біт, який виділяють на мантису у форматі числа з рухомою комою IEEE754 128-бітів.

Таким чином, ми приходимо до необхідності форматів чисел з рухомою комою, довжина яких значно перевищує 128 бітів. Сучасні мови програмування можуть працювати з надзвичайно довгими цілими числами, Наприклад існує клас cBigNumber, який реалізує цілі числа необмеженої розрядності для C++ і в якому передбачені всі стандартні операції мови C++, а за допомогою цілих чисел можна реалізовувати і дійсні числа.

Питання полягає тільки в тому, яким чином визначити критерій необхідності застосування того чи іншого формату чисел для ефективної роботи програми, оскільки збільшення розрядності подання вдвічі означає збільшення часу виконання арифметичних дій більш ніж вдвічі. У цьому контексті проблема, поставлена Румпом, залишається відкритою. З чого починали – до того ж і прийшли.

4.4. Особливості стандарту IEEE Std 754-2008

Стандарт IEEE754-2008 [9]:

– врахував ряд помилок і недоліків стандарту 1985 р.;
– встановив два типи форматів чисел з рухомою комою: бінарний та десятковий;
– стандартизував набір базових і розширених форматів чисел;
– описав правила переходу між різними типами форматів, а саме між:
  – цілими числами та числами з рухомою комою;
  – бінарними та десятковими числами з рухомою комою;
  – визначив правила здійснення арифметичних дій над числами з рухомою комою, а також операцій добування квадратного кореня, порівняння чисел та ряд інших.

Порівняно з попереднім стандартом вперше описано **новий тип комп'ютерних чисел – десяткові числа з рухомою комою**, визначено два типа не-чисел, так звані qNaN і sNaN, описано правила їх застосування та дій над ними.

4.5. Бінарні формати стандарту IEEE754-2008

Бінарні формати нового стандарту такі ж, як і в стандарті 1985 р., за винятком деяких нових. Основними, базовими форматами є одинарний (Single, 32 біти), подвійний (Double, 64 біти) та четверний (Quadruple, 128 бітів). Будь-який бінарний формат містить вже відомі нам три поля:

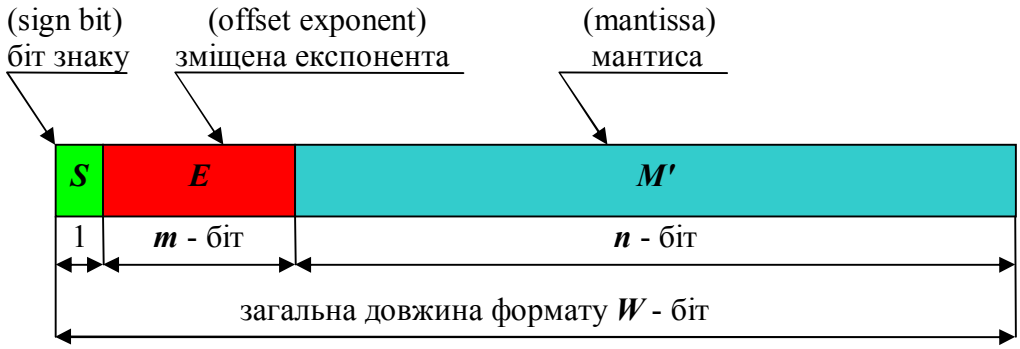


Рисунок 4.2. Загальний формат бінарних чисел з рухомою комою

Окрім базових бінарних форматів стандарт IEEE754-2008 добавляє ще формат половинної точності (Half, 16 бітів) та сімейство множинних форматів, загальна довжина яких становить $W = 32 \times N$, де N – будь-яке натуральне число більше від чотирьох. Таким чином, можливі такі додаткові формати: 32·5 = 160 бітів, 32·6 = 192 біти, 32·7 = 224 біти, 32·8 = 256 бітів і т.д. Однак **формату 32·3 = 96 бітів не існує**. В табл. 4.1. показано можливі формати бінарних чисел з рухомою комою.

Таблиця 4.1. Параметри бінарних форматів IEEE754-2008-чисел

Параметр	Бінарний 16 бітів	Бінарний 32 біти	Бінарний 64 біти	Бінарний 128 бітів	Бінарний W бітів ($W > 128$)
W, загальна довжина формату	16	32	64	128	$W = 32 \times k$, $k > 4$
m, довжина експоненти	5	8	11	15	$round[4\times\log_2(W)]-13$
n, довжина залишку мантиси	10	23	52	112	$W - m - 1$
точність подання в десяткових розрядах	≈ 3	≈ 7	≈ 16	≈ 34	$\approx round[(n+1)\times\lg 2]$
e_{max}, максимальна експонента та зміщення *bias*	15	127	1023	16383	$2^{(W-n-2)}-1$
максимальне число A_{max}	$\approx 10^4$	$\approx 10^{38}$	$\approx \pm 10^{308}$	$\approx \pm 10^{4932}$	$\approx 10^{round[(emax+1)\times\lg 2]}$
Функція *round* округлює аргумент до найближчого цілого числа					

Принципи подання нуля, нескінченності, субнормальних чисел, не-чисел (NaN) залишаються, в основному, ті ж, що і для стандарту 1985 р.:

а) Якщо $\boldsymbol{E} = 2^{\boldsymbol{m}} - 1$ (тобто всі біти експоненти рівні 1) і $\boldsymbol{M'} \neq 0$, то маємо випадок не-чисел (NaN). Однак, на відміну від стандарту 1985 р., розрізняють qNaN (якщо перший біт залишку мантиси рівний 1, а решта – довільний набір бітів) та sNaN (якщо перший біт залишку мантиси рівний 0, а решта – довільний **ненульовий** набір бітів).

б) Якщо $\boldsymbol{E} = 2^{\boldsymbol{m}} - 1$ (тобто всі біти експоненти рівні 1) і $\boldsymbol{M'} = 0$, то маємо випадок $\pm \infty$ (нескінченність, знак якої визначається першим бітом подання $\boldsymbol{s}$).

с) Якщо $1 \leq \boldsymbol{E} \leq 2^{\boldsymbol{m}} - 2$, тоді маємо числа з рухомою комою

$$\mathbf{A} = (-1)^{\boldsymbol{S}} \times (1 + \boldsymbol{M'} \times 2^{-\boldsymbol{n}}) \times 2^{\boldsymbol{E-bias}}.$$

д) Якщо $\boldsymbol{E} = 0$ і $\boldsymbol{M'} \neq 0$, тоді маємо випадок субнормальних чисел

$$\mathbf{A} = (-1)^{\boldsymbol{S}} \times (0 + \boldsymbol{M'} \times 2^{-\boldsymbol{n}}) \times 2^{-\boldsymbol{bias}}.$$

е) Якщо $\boldsymbol{E} = 0$ і $\boldsymbol{M'} = 0$, то такий набір бітів подає +0 або -0 залежно від величини першого біту подання $\boldsymbol{s}$.

Крім описаних вище форматів передбачено також ряд додаткових розширених бінарних форматів, розгляд яких не є завдання цього посібника. Бажаючих поглиблено вивчити це питання відсилаємо до [9], розділ 3.7.

Таким чином, стандарт IEEE754-2008 описує широкий набір бінарних форматів, які можна використовувати для організації обчислень з різною точністю та діапазоном подання чисел.

4.6. Не-числа qNaN та sNaN

В обчисленнях не-число (NaN – Not a Number) є величина числового типу, яка подає невизначені дані чи такі, які не можуть бути відображені, особливо для обчислень з рухомою комою. Фактично NaN свідчить про помилку введення, обчислення чи подання. Систематичне використання NaN було введено стандартом IEEE 754-1985 [7], (розділ 3.1) з метою ефективного опрацювання таких помилок одночасно з поданням інших специфічних величин, таких як нескінченність. Однак NaN не тотожне нескінченності, хоча обидві величини (NaN і $\pm\infty$) використовують в якості спеціальних випадків подання реальних чисел, як чисел з рухомою комою, а також операцій над такими числами. Наприклад, 0/0 є невизначеною величиною як дійсне число, тому його подають за допомогою NaN, квадратний корінь з від'ємного числа є уявне число, тому не може бути подане як число з рухомою комою і також є NaN і, врешті, NaN можна використовувати для подання помилкових величин в обчисленнях.

Існує три види операцій, які можуть повертати NaN – це операції, які визначені не на всій множині чисел з рухомою комою:

1. Операції, де один із операндів є NaN.

2. Невизначені операції:

– ділення 0/0, $\pm\infty/\pm\infty$;

– множення $0\times\pm\infty$, $\pm\infty\times0$;

– додавання $+\infty + (-\infty)$, $(-\infty) + +\infty$, еквівалентні віднімання;
– ряд стандартних функцій мов програмування, наприклад, **pow**, **pown** для специфічних випадків 0^0, 1^∞ та ∞^0 повертають 1, а деякі (**powr**) – NaN.

3. Дійсні операції з комплексним результатом:
– квадратний корінь з від'ємного числа;
– логарифм з від'ємного числа;
– арксинус чи арккосинус з числа, більшого від 1.

У той же час існують такі операції, які є некоректними, наприклад, арифметичні дії над символьними даними, а також такі, які можуть повертати звичайні числа, однак під час їх виконання було отримано арифметичне переповнення (overflow), яке часто подають у вигляді $\pm\infty$, чи арифметичне недоповнення (underflow), яке зазвичай подають найменшим з нормальних чи субнормальних чисел або нулем. Ці специфічні помилки обчислень мають іншу природу, ніж ті, що пов'язані з обмеженнями області визначення операцій. Саме тому стандарт IEEE754-2008 розширив поняття не-числа і передбачив два різних видів NaN, які позначають qNaN (quiet – "тихі") і sNaN (signaling – "сигналізуючі").

qNaN використовують для того, щоби подавати помилки, які виникають е результаті неможливих операцій чи величин. Поява qNaN зазвичай не викликає жодних проблем, оскільки вони можуть проходити через всі етапи обчислення. Виняток становлять випадки, коли qNaN не можуть непомітно проходити через обчислення, наприклад, у форматі перетворення типів або деяких операцій порівняння (де "не очікують" введення NaN).

sNaN сигналізують, що внаслідок виконання операції було отримано певний винятковий результат. sNaN можуть підтримувати розширені особливості, такі як змішування числових і символьних обчислень або інших розширень до базової арифметики з рухомою комою. sNaN може бути перетворено в qNaN і надалі проходити через всі етапи обчислення. Стандарт IEEE754-2008 подає декілька способів використання sNaN:

1. Попереднє заповнення неініціалізованої пам'яті sNaN-ами буде призводити до появи некоректних операцій, якщо буде спроба використати дані до їхньої ініціалізації.
2. Використання sNaN для повідомлень про:
   – отримання чисел, які мають недоповнення (underflowed numbers);
   – отримання чисел, які мають переповнення (overflowed numbers);
   – присвоювання змінній певного формату числа формату вищої точності;
   – отримання комплексних чисел.

Не-числа NaN стандарту IEEE754-2008 (як і у стандарті 1985 р.) подають шляхом встановлення всіх бітів у полі зміщеної експоненти в значення "1" (як для нескінченності) і деяким, не нульовим числом у полі залишку мантиси (щоби відрізнятися від нескінченості). Таке подання допускає визначення множини різних значень величин NaN, які залежать не тільки від того, які біти встановлені в "1" у полі залишку мантиси, але також і від величини старшого, знакового біту. Наприклад, NaN 32-бітного формату з рухомою комою мають такий формат: **s111 1111 1*a*xx xxxx xxxx xxxx xxxx xxxx**, де **s** є знак (який часто

ігнорують програмно чи апаратно), ***a*** – визначає тип NaN, "**x...x**" є додатковою корисною інформацією (яку також часто ігнорують). Якщо ***a*** = 1, то маємо qNaN, якщо ***a*** = 0 і решта бітів залишку мантиси є ненульові, то маємо sNaN. (Стандарт 1985 р. ці числа не розрізняє).

Операції над числами з рухомою комою, крім команд порівняння, у виняткових випадках, описаних вище, зазвичай повертають qNaN. Порівняння чисел з NaN має свої особливості. Операндами можуть бути числа, qNaN та sNaN. Результатом порівняння завжди є qNaN, однак якщо число порівнюють з самим собою, то тотожність ***x*** = ***x*** повертає істинне значення (true) для дійсних чисел, хибне значення (false) для ***x*** = qNaN і значення qNaN у випадку ***x*** = sNaN. Таку властивість можна використати для перевірки чи число ***x*** є qNaN. Існує також несигналізуюча версія такої перевірки. Предикат "Чи є число ***x*** – NaN?" визначає, чи є величина ***x*** не-числом NaN і ніколи не сигналізує про виняток, навіть коли ***x*** є sNaN.

Оскільки операції над NaN майже завжди мають результатом NaN, то отримання в процесі здійснення ряду арифметичних дій результату типу NaN свідчить про помилки, які виникли на певному етапі обчислень. Таким чином, відпадає необхідність контролю помилок на кожному із етапів процесу обчислень, достатньо просто перевірити на наявність помилок результат. Однак, слід зауважити, що залежно від мови і функції, NaN може бути непомітно видалене з виразу, який дає однаковий результат для всіх інших величин з рухомою комою, наприклад, значення степеневої функції x^y, яке у випадку ***y*** = 0 для всіх звичайних чисел з рухомою комою дає значення рівне 1, для ***x*** = NaN також може бути визначене як 1, тому якщо в процесі обчислень використовують такого роду функції, слід здійснювати перевірку на предмет, чи появлялись NaN під час виконання програми, наприклад, встановленням прапорця "Неприпустима операція".

NaN також може бути явно присвоєне змінним, які подають помилки. До введення стандарту IEEE-754 програмісти часто використовували спеціальні величини, зазвичай максимальні значення форматів подань чисел для того, щоби позначати помилки. Такий спосіб не дає гарантії, що ці помилки будуть адекватно опрацьовані. У випадку використання NaN для позначення помилок програміст звільняється від необхідності контролювати результат виконання кожної операції на предмет виявлення цих помилок.

4.7. Принципи подання десяткових чисел з рухомою комою

Більшість людей настільки звичним вважають арабські цифри 0...9 та десяткову систему числення, що про існування інших, можливо зручніших систем числення навіть не підозрюють. Основна кількість різноманітних побутових, фінансових, технічних, наукових обчислень та зберігання даних в архівах зазвичай здійснюють саме в десятковій системі числення. Однак обчислювальні машини працюють у двійковій системі числення, тому неминуче здійснювати насамперед перехід з десяткової в двійкову, а після обчислень, – навпаки: з двійкової в десяткову системи числення для того, щоби результати обчислення були зрозумілі пересічному користувачу.

Використання прямого, інверсного чи доповняльного кодів для подання цілих чисел, а також бінарних форматів чисел з рухомою комою для подання дійсних чисел має ряд позитивних переваг перед іншими способами кодування з огляду на простоту реалізації арифметичних дій за допомогою електронних схем, а також високу швидкість виконання цих дій. Однак, як було показано в п. 4.1, бінарне кодування має ряд суттєвих недоліків, серед яких особливо слід відмітити недоліки, пов'язані з нескінченним двійковим поданням скінченних десяткових дробів та з правилами округлення двійкових чисел, які не відповідають загальноприйнятим правилам округлення десяткових чисел. Одним із способів усунення цих та інших недоліків, є подання десяткових чисел за допомогою такого спеціального бінарного кодування, яке забезпечувало би швидкий та простий перехід між десятковою та двійковою системами числення. Реалізацію арифметичних дій між числами, закодованими в такий спосіб, можна здійснювати як за допомогою апаратних засобів ЕОМ (електронні схеми), так і програмно. Зрозуміло, що таке кодування неминуче ускладнює електронні схеми арифметико-логічних пристроїв мікропроцесорів і збільшує час виконання арифметичних дій.

Якщо вибрано тип кодування, то десяткові числа з рухомою комою реалізують у такий спосіб:

1. Число **A**, яке необхідно подати, слід записати в експоненціальному (напівлогарифмічному) вигляді:

$$\mathrm{A} = (-1)^S \times M \times 10^e, \qquad (4.8)$$

де ***S*** – 0 для додатних та 1 для від'ємних чисел, ***M*** – мантиса, яку записують у вигляді **цілого додатного числа**, що має ***k*** десяткових розрядів, ***e*** – експонента, яка визначає порядок числа. Числа ***M*** та ***e*** записують у десятковій системі числення. Особливість десяткової мантиси в тому, що вона **не містить дробових розрядів**, наприклад число -23,567 буде записане як

$$A = -23{,}567 = (-1)^1 \times 23567 \times 10^{-3}.$$

Однак, якщо кількості розрядів зарезервованих форматом числа не достатньо для запису всієї мантиси, то лишні, молодші розряди будуть відкинуті.

2. Числа ***M*** та ***e*** за допомогою вибраного кодування записують у двійковій системі числення. Зазвичай число ***M*** кодують спеціальним бінарним кодом, а до числа ***e*** застосовують зміщений двійковий код, який забезпечує подання як додатних, так і від'ємних експонент.

3. Виділяють певні поля бітів для запису кодованих знака, експоненти та мантиси. Сукупність цих величин, та їхня загальна довжина задає формат десяткових чисел з рухомою комою.

У результаті отримуємо двійкове подання десяткового числа з рухомою комою.

Увага! Бінарні формати IEEE754-2008 описують двійкові числа з рухомою комою, на відміну від десяткових чисел з рухомою комою, які тільки записують за допомогою спеціального двійкового кодування. Однак двійкові і десяткові числа з рухомою комою є різними поданнями дійсних (реальних) десяткових чисел.

4.8. Двійково-десяткове кодування

Одним із способів кодування десяткових чисел є **двійково-десяткове кодування** (**Binary Coded Decimal, BCD**), яке раніше носило назву **код 8-4-2-1**. Таке кодування є гібридом двійкової та шістнадцяткової системам числення. Зміст цього кодування полягає у заміні кожної десяткової цифри відповідною двійковою тетрадою як в шістнадцятковій системі числення (див. табл. 1.3):

$$259{,}47\text{d} = 0010\ 0101\ 1001{,}0100\ 0111\text{bcd}.$$

Аналогічним чином, двійково-десятковий код переводять у десятковий шляхом розділення бінарного запису на тетради вліво та вправо від коми та заміні кожної тетради відповідною їй десятковою цифрою. Якщо розрядів не вистачає до повної тетради, її доповнюють нулями. Таке переведення можливе тільки в тому випадку, коли значення кожної тетради не перевищує дев'яти.

Над числами, записаними в двійково-десятковому коді, арифметичні дії здійснюють за дещо складнішими правилами, оскільки, наприклад, у десятковій системі 5d + 8d = 13d, а здійснивши додавання в двійковій системі над двійково-десятковими числами 0101bcd + 1000bcd отримаємо значення 1101b, яке в двійково-десятковій системі не відповідає жодній десятковій цифрі. Двійково-десяткове кодування часто використовують у різноманітних вимірювальних пристроях, які видають результати в доступній формі, однак таке кодування має ряд суттєвих недоліків. Про один з них вже було згадано вище і призводить до ускладнення виконання арифметичних дій над такими числами. **Другий недолік** полягає в тому, що двійково-десяткове кодування не є економічним, тобто не використовує всі можливі кодові комбінації двійкових розрядів. Наприклад, для кодування чисел від 0 до 999d необхідно 12 двійково-десяткових розрядів, тоді як для звичайного беззнакового двійкового числа з надлишком вистачає 10-ти розрядів: 11 1110 0111b = 999d, а набори бітів від 11 1110 1000b (1000d) до 11 1111 1111b (1023d) є надлишковими.

Надлишковість двійково-десяткового коду не залежить від кількості десяткових розрядів k і може бути обчислена, якщо врахувати, що для подання одного десяткового розряду необхідно приблизно 3,32 двійкових розряди (точно $\log_2 10$, див. формулу (3.12)). **Надлишковість** двійково-десяткового коду, а також інших надлишкових кодів, **обчислюють як відношення кількості надлишкових розрядів до кількості необхідних**

$$\frac{4-\log_2 10}{\log_2 10}\times 100\% = (4\lg 2 - 1)\times 100\% \approx 20{,}41\%. \qquad (4.9)$$

З іншого боку, **ефективність використання кодових комбінацій** двійково-десяткового коду, яку **обчислюють як відношення кількості використовуваних кодових комбінації до їхньої загальної кількості**, залежить від кількості десяткових розрядів k і становить

$$\frac{10^k}{2^{4k}}\times 100\%. \qquad (4.10)$$

З табл. 4.2 видно, що ефективність використання кодових комбінацій дуже швидко зменшується зі зростанням кількості розрядів. Саме тому **двійково-десяткове кодування не використовують для запису десяткових чисел з**

рухомою комою, оскільки це призводило би до неефективного використання пам'яті та обчислювальних ресурсів ЕОМ, а в кінцевому результаті зменшення точності подання числа та звуження діапазону чисел (оскільки розмір машинного слова є обмеженим).

Таблиця 4.2. Ефективність кодових комбінацій двійково-десяткового коду

Кількість десяткових розрядів ***k***,	1	2	3	4	5	6
Ефективність використання кодових комбінацій, %	62,5	≈ 39,1	≈ 24,4	≈ 15,3	≈ 9,5	≈ 6,0

4.9. Щільно упаковані десяткові числа

Зменшити надлишковість коду та підвищити ефективність використання кодових комбінацій допомагає **щільно упаковане десяткове кодування (Densely Packed Decimal, DPD)**. Таке кодування використовує доволі зручне співвідношення між степенями двійки та десятки:

$$2^{10} = 1024 \cong 1000 = 10^3. \qquad (4.11)$$

Це співвідношення означає, що для кодування 3-х десяткових розрядів з невеликим надлишком вистачає 10-ти двійкових розрядів. Дійсно, для кодування чисел від 0 до 999 необхідно 1000 кодових комбінацій, а ще 24 виявляються "лишніми" і можуть бути використані з іншою метою, наприклад, для кодування помилок обчислень.

Існує два основних способи кодування десяткових тріад двійковими декадами: кодування **Chen-Ho** [15] та кодування **Mike Cowlishaw** [16, 17]. Саме останнє і називають щільно упакованим десятковим кодуванням (DPD). Стандарт IEEE754-2008 використовує саме цей тип кодування, тому розглянемо його детальніше.

Щільно упаковані десяткові числа будують на основі звичайних двійково-десяткових кодів. Спочатку десяткову тріаду кодують звичайним двійково-десятковим кодом, замінюючи кожну десяткову цифру відповідною їй двійковою тетрадою. В такий спосіб отримуємо 12-розрядний двійково-десятковий код BCD. Далі використовують ту властивість двійково-десяткового кодування, що цифри від 0 до 7 включно, так звані **малі цифри**, мають старший біт рівний нулю:

0	–	0000,	4	–	0100,
1	–	0001,	5	–	0101,
2	–	0010,	6	–	0110,
3	–	0011,	7	–	0111.

Отже, якщо старший біт рівний нулю, то число $0abc_{(2)}$ відповідає десятковій цифрі згідно з виразом: $a\times4 + b\times2 + c$.

Цифри 8 та 9, так звані **великі цифри**, кодують тетрадами, в яких старший біт рівний 1, а два наступні біти рівні нулю і тільки наймолодший біт визначає значення цифри:

8	–	1000,	9	–	1001.

Тоді число $100c_{(2)}$ відповідає десятковій цифрі, яку обчислюють згідно з виразом: $8 + c$. Таким чином, якщо старший біт тетради рівний 1, то два наступні розряди не використовуються для кодування цієї десяткової цифри, тому ці розряди можна використати для кодування інших десяткових цифр. Табл. 4.3 демонструє, яким чином 12-розрядний двійково-десятковий код записують за допомогою 10-ти двійкових розрядів. Таке кодування та декодування легко здійснюють за допомогою електронних схем чи програмним способом.

Таблиця 4.3. Правила кодування щільно упакованих десяткових чисел **(DPD)**

Щільно упаковані десяткові числа (DPD)										**Двійково-десяткові числа (BCD)**				
b9	**b8**	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**	**d2**	**d1**	**d0**	**Значення десяткових цифр**	**Опис**
a	**b**	**c**	**d**	**e**	**f**	0	**g**	**h**	**i**	**0abc**	**0def**	**0ghi**	(0–7) (0–7) (0–7)	три малі цифри
a	**b**	**c**	**d**	**e**	**f**	1	0	0	**i**	**0abc**	**0def**	**100i**	(0–7) (0–7) (8–9)	дві малі цифри, одна велика
a	**b**	**c**	**g**	**h**	**f**	1	0	1	**i**	**0abc**	**100f**	**0ghi**	(0–7) (8–9) (0–7)	
g	**h**	**c**	**d**	**e**	**f**	1	1	0	**i**	**100c**	**0def**	**0ghi**	(8–9) (0–7) (0–7)	
a	**b**	**c**	1	0	**f**	1	1	1	**i**	**0abc**	**100f**	**100i**	(0–7) (8–9) (8–9)	одна мала цифра, дві великі
d	**e**	**c**	0	1	**f**	1	1	1	**i**	**100c**	**0def**	**100i**	(8–9) (0–7) (8–9)	
g	**h**	**c**	0	0	**f**	1	1	1	**i**	**100c**	**100f**	**0ghi**	(8–9) (8–9) (0–7)	
x	x	**c**	1	1	**f**	1	1	1	**i**	**100c**	**100f**	**100i**	(8–9) (8–9) (8–9)	три великі цифри

Над щільно упакованими десятковими числами арифметичні дії не можна здійснювати звичайним способом. Тому такі числа конвертують у двійково-десятковий код, у цьому коді здійснюють арифметичні дії, а результат знову перекодовують у щільно упакований код.

В останньому рядку таблиці біти b9 та b8 можуть набувати різних значень, однак їх встановлюють у нульове значення. Всі можливі комбінації бітів за умови, що біти b6, b5, b3, b2, b1 рівні 1, а хоча б один із бітів b9 та b8 не рівний нулю (всього 24 такі комбінації) можна використовувати для додаткового кодування певних ознак чисел чи результатів арифметичних дій.

Надлишковість щільно упакованих десяткових чисел складає:

$$\frac{10-\log_2 1000}{\log_2 1000}\times 100\% = \left(\frac{10}{3}\lg 2 - 1\right)\times 100\% \approx 0{,}34\%, \qquad (4.12)$$

а ефективність використання кодових комбінацій

$$\frac{10^{3p}}{2^{10p}}\times 100\%, \qquad (4.13)$$

де $\boldsymbol{p}$ – кількість десяткових тріад у записі десяткового числа. Для $\boldsymbol{p}=1$ ($\boldsymbol{k}=3$) ефективність становить 97,66%, а для $\boldsymbol{p}=2$ ($\boldsymbol{k}=6$) – 95,37% (порівняйте з 24,4% та 6% відповідно для двійково-десяткового кодування, табл.4.2).

Таким чином, щільно упаковані десяткові числа мають низьку надлишковість кодування та високу ефективність використання кодових комбінацій, що у поєднанні з доволі простими правилами переведення з десяткової системи числення у щільно упакований десятковий код та навпаки, дає переваги над звичайним двійково-десятковим кодуванням. Недоліком щільно упакованих двійкових кодів є доволі складний спосіб виконання арифметичних дій над такими числами, однак сучасні апаратні (схемотехнічні) та програмні засоби ефективно вирішують цю проблему.

4.10. Десяткові числа з рухомою комою стандарту IEEE754-2008

Комп'ютерне подання дійсних чисел, записаних у експоненціальній десятковій формі за допомогою скінченної кількості двійкових розрядів називають десятковим числом з рухомою комою.

У випадку бінарних (двійкових) чисел з рухомою комою дійсне десяткове число спочатку необхідно перевести в двійкову систему числення, подати у вигляді нормалізованої форми (або нормальної форми в окремих випадках), сформувати залишок мантиси та зміщену експоненту (див. п. 3.3.2).

У випадку десяткових чисел з рухомою комою:

1) **Дійсні десяткові числа спочатку округлюють до *k* значущих цифр (*k* визначається форматом подання).**
2) **Формують мантису та експоненту таким чином, щоби мантиса являла собою ціле додатне число.**
3) **Кодують мантису шляхом заміни десяткових тріад на двійкові щільно упаковані декади.**
4) **Експоненту подають за допомогою бінарного зміщеного коду.**

(Оскільки в загальному випадку $k = 3\times p + 1$, то найстаршу десяткову цифру мантиси кодують особливим чином).

Всі десяткові числа з рухомою комою мають загальний формат (рис.4.3):

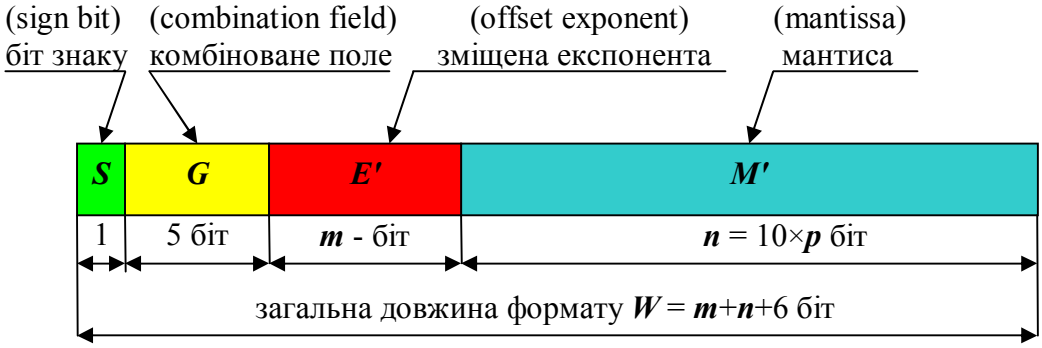


Рисунок 4.3. Загальний формат десяткових чисел з рухомою комою.

Порівняно з форматом бінарних чисел з рухомою комою (рис.3.1) добавлено додаткове комбіноване поле ***G*** довжиною 5 біт. Це комбіноване поле відіграє потрійну роль:

1. Містить першу десяткову цифру мантиси, записану в двійково-десятковому коді.
2. Містить перших два біти зміщеної експоненти.

3. Може кодувати тип числа:

а) звичайне, яке визначається за формулою (4.14),

б) нескінченність,

в) не-число (qNaN або sNaN).

Загальна форма запису десяткових чисел з рухомою комою:

$$A = (-1)^S \times M \times 10^{E-bias}, \qquad (4.14)$$

де ***M*** – ціле додатне число. Таким чином, перша цифра мантиси може набувати значення від 0 до 9.

Якщо у бінарному форматі перша цифра завжди 1 (для нормалізованих мантис) або 0 (для денормалізованих) і її можна неявно відкинути, тобто не виділяти додаткового розряду на її запис, то для десяткового формату першу цифру мантиси відкидати не можна і для її відображення виділяють 4 двійкові розряди. Ці розряди першої цифри мантиси містяться в полі ***G***.

Решта розрядів мантиси (залишок мантиси ***M′***) завжди відображає кількість десяткових цифр кратну 3 або ціле число ***p*** десяткових тріад. Кожну тріаду кодують двійковою декадою у форматі щільно упакованих десяткових чисел. Таким чином, залишок мантиси завжди має довжину кратну 10 бітам: ***n*** = 10×***p*** біт і подає 3×***p*** десяткових розрядів.

Поле ***E′*** містить залишок експоненти, записаної в двійковому зміщеному коді, оскільки старших два біти експоненти також записують у полі ***G***. Отже, повна експонента ***E*** має довжину ***m***+2. Принципи формування комбінованого поля ***G*** демонструє табл. 4.4.

Таблиця 4.4. Кодування комбінованого поля ***G.***

Комбіноване поле ***G*** (5 бітів)	Тип числа	Старші біти експоненти	Перша цифра мантиси в BCD	Десяткове значення першої цифри мантиси
a b c d e	Скінченні числа	a b	0 c d e	від 0 до 7
1 1 a b e		a b	1 0 0 e	8 або 9
1 1 1 1 0	Нескінченність	– –	– – – –	–
1 1 1 1 1	NaN	– –	– – – –	–

Тут слід зауважити таке:

1. Біти a та b не можуть бути одночасно рівні одиниці, вони можуть набувати тільки значень 00, 01 та 10. Оскільки повна довжина експоненти рівна ***m***+2, то мінімальне значення зміщеної експоненти становить 000...000 (0), а максимальне – 101...111 (2^{m+2}-2^{m}-1).
2. Якщо всі біти поля рівні одиниці (NaN), тоді старший біт залишку експоненти ***E′*** вказує на тип не-числа: 0 – qNaN, 1 – sNaN. В цьому разі всі інші біти поля ***E′*** змісту не мають.

Приклад 4.1. Число **X** = -444,5553 записати в форматі десяткових чисел з рухомою комою одинарної точності стандарту IEEE-754-2008.

Розв'язування. Мантиса подання повинна бути цілим числом, тому запишемо число -444,5553 у такому вигляді:

$$-444{,}5553 = (-1)^1 \times 4445553 \times 10^{-4}.$$

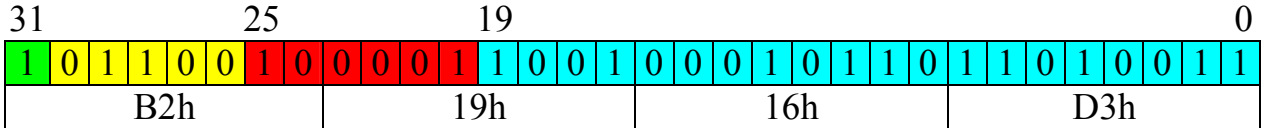
1. Знаковий біт рівний 1 (***S*** = 1).

2. Одинарний стандарт має 7 десяткових розрядів, а число **X** – 7 значущих цифр, тому всі сім цифр кодуються. Найстарша цифра 4 буде відображена в полі ***G***, із шести інших формуємо тріади 445 та 553. Тріада 445 кодується в **BCD** як 0100 0100 0101. Всі цифри малі, тому відповідний їм **DPD**-код становить 1001000101. Аналогічно отримуємо для тріади 553: **BCD** = 0101 0101 0011, а DPD-код становить 1011010011.

Експонента ***e*** = -4, зміщення для заданого формату становить 101d, тому зміщена експонента рівна ***E*** = -4+101 = 97d = 0110 0001b. Отже в поле ***E'*** записуємо молодших 6 бітів: 100001.

Формуємо поле ***G***. Старші біти експоненти рівні 01, а старша цифра мантиси – 4 = 0100. Тому поле матиме вигляд 01100.

Записуємо поля ***S***, ***G***, ***E'*** та ***M'***:



Відповідь: -444,5553 = B21916D3h.

4.11. Когорти

На відміну від двійкового формату чисел з рухомою комою, число у десятковому форматі чисел з рухомою комою може мати велику кількість подань. Наприклад, якщо мантиса ***M*** є кратна 10, а ***e*** < ***emax*** (максимальна експонента), тоді числа (***S***, ***e***, ***M***) та (***S***, ***e***+1, ***M***/10) є двома різними поданнями одного і того ж числа. Така властивість десяткових чисел з рухомою комою випливає з того факту, що мантису ***M*** записують у вигляді **цілого додатного числа**, що має ***k*** десяткових розрядів (див. п. 4.7). Наприклад, число +7,572d може бути записане як

$$+7{,}572 = +7572 \times 10^{-3} = +75720 \times 10^{-4} = +757200 \times 10^{-5} = +7572000 \times 10^{-6} = \ldots$$

Якщо взяти 32-бітний формат десяткового числа з рухомою комою, який передбачає 7 десяткових знаків мантиси, то отримаємо, що дане число може мати 4 різних за формою, але тотожних за значенням подання (0, -3, 7572), (0, -4, 75720), (0, -5, 757200) та (0, -6, 7572000).

Такий набір подань десяткового числа з рухомою комою, члени якого являють собою відмінні одне від одного подання одного і того ж числа називають когортою.

Незважаючи на те, що різні члени когорти мають однакові значення, вони можуть відрізнятися для так званих десятково-специфічних операцій. Когорти різних чисел з рухомою комою можуть мати неоднакову кількість членів. Якщо подання скінченого числа має ***n*** значущих цифр (враховують всі цифри від найстаршої ненульової до наймолодшої ненульової, включно), то когорта матиме щонайбільше ***k*** - ***n*** + 1 членів, де ***k*** є число значущих цифр формату або

точність подання в десяткових розрядах. Наприклад, одноцифрове число з рухомою комою може мати до k різних подань, тоді як число, що містить k цифр і не закінчується нулем, має тільки одне подання. Число, що містить n – цифр може мати дещо менше ніж $k - n + 1$ членів, якщо його значення наближається до меж діапазону подання експонент. Нуль має найбільшу когорту: когорта +0 містить подання для кожної з експонент, когорта -0 формується аналогічно.

Оскільки десяткове число з рухомою комою може мати декілька подань, то арифметика таких чисел передбачає не тільки обчислення достовірного числового результату, але й вибір правильного члена когорти результату обчислень. **Значення результату**, отже і його когорта, визначається операцією і значенням операндів і ніколи **не залежить від подання операндів**, **однак вибір конкретного подання** результату (з його когорти) **залежить від подання операндів**.

Результат обчислення може бути точним і наближеним. Наближений результат завжди округлюють до k значущих цифр, тому він може закінчуватись нулем або ненульовою цифрою. Для всіх обчислювальних операцій, крім вирівнювання, у випадку наближеного результату використовують член когорти з найменш можливим показником експоненти для того, щоби отримати максимальну кількість значущих цифр. Наприклад, результат обчислення +3,36005838292 буде округлений до +3,360058 (єдине подання в когорті – (0, -6, 3360058)), а результат 2345,100349 – до 2345,100 (три подання в когорті: (0, -3, 2345100), (0, -2, 234510) та (0, -1, 23451)). В останньому випадку в якості результату буде вибрано подання (0, -3, 2345100). Тут і надалі слід зауважити, що **нулі мантиси, які стоять справа від ненульової цифри подання вважаються значущими цифрами**.

Якщо результат є точним, то член когорти вибирають залежно від так званої оптимальної експоненти результату. **Оптимальна експонента (preferred exponent)** – це таке значення експоненти E, яке є найменшим з експонент вхідних операндів. Якщо когорта результату обчислення не містить подання з оптимальною експонентою, то буде вибрано член з експонентою, яка якнайближча до оптимальної. Наприклад для чисел +5,1000 (0, -4, 51000) та 48 (0, 0, 48) оптимальна експонента становить -4. У результаті додавання буде отримано результат 53,1000 (0, -4, 531000). Для пари чисел 35 (0, -5, 3500000) та 2000 (0, 0, 2300) оптимальна експонента становить -5, однак когорта результату додавання $35 + 2000 = 2035$ не містить члена (0, -5, 203500000), оскільки кількість цифр мантиси не повинна перевищувати 7. Тому буде вибрано член когорти, який має експоненту -3 (0, -3, 2035000), оскільки значення цієї експоненти найближче до оптимального.

Як було сказано вище, нуль має подання для кожної з експонент, тому для цілого x, залежно від подання нуля, результатом $0 + x$ може бути будь-який член когорти x. Якщо когорта результату обчислення не містить членів з оптимальною експонентою, то буде вибрано таке подання, експонента якого ближча до оптимальної.

Приклад 4.2. Записати всіх членів когорти числа **X** = +0,829 у форматі десяткових чисел з рухомою комою одинарної точності стандарту IEEE754-2008.

Запишемо всі можливі подання числа **X**, враховуючи, що для формату 32 біти максимальна кількість значущих цифр подання $k = 7$, а мантиса є цілим додатним числом:

$$+0{,}829 = +829\times10^{-3} = +8290\times10^{-4} = +82900\times10^{-5} =$$
$$= +829000\times10^{-6} = +8290000\times10^{-7}.$$

Отже когорта числа **X** містить 5 подань:

(0, -3, 829), (0, -4, 8290), (0, -5, 82900), (0, -6, 829000) та (0, -7, 8290000).

1. Число додатне, тому знаковий біт для всіх подань рівний 0 ($S = 0$).

Запишемо перше подання (0, -3, 829):

7 десяткових розрядів мантиси мають вигляд 0000829. Найстарша цифра 0 буде відображена в полі ***G***, із шести інших формуємо тріади 000 та 829.

Для тріади 000: **BCD =** 0000 0000 0000. Всі цифри малі, тому відповідний їм **DPD**-код становить 0000000000. Для тріади 829: **BCD** = 1000 0010 1001. Маємо дві великі і одну малу цифри, а DPD-код становить 0100101111.

Експонента $e = -3$, тому зміщена експонента рівна $E = -3+101 = 98\text{d} =$ = 0110 0010b. Отже в поле ***E′*** записуємо молодших 6 бітів: 100010.

Формуємо поле ***G***. Старші біти експоненти рівні 01, а старша цифра мантиси – 0 = 0000. Тому поле матиме вигляд 01000.

Записуємо поля ***S***, ***G***, ***E′*** та ***M′***:

0 01000 100010 0000000000 0100101111.

Групуємо на тетради:

0010 0010 0010 0000 0000 0001 0010 1111 = 2220012Fh.

Запишемо друге подання (0, -4, 8290):

7 десяткових розрядів мантиси мають вигляд 0008290. Найстарша цифра 0 буде відображена в полі ***G***, із шести інших формуємо тріади 008 та 290.

Для тріади 008: **BCD =** 0000 0000 1000. Одна велика цифри і дві малі, тому відповідний їм **DPD**-код становить 0000001000. Для тріади 290 аналогічно: **BCD** = 0010 1001 0000. Одна велика цифри і дві малі, DPD-код становить 0100011010.

Експонента $e = -4$, тому зміщена експонента рівна $E = -4+101 = 97\text{d} =$ = 0110 0001b. Отже в поле ***E′*** записуємо молодших 6 бітів: 100001.

Формуємо поле ***G***. Старші біти експоненти рівні 01, а старша цифра мантиси – 0 = 0000. Тому поле матиме вигляд 01000.

Записуємо поля ***S***, ***G***, ***E′*** та ***M′***:

0 01000 100001 0000001000 0100011010.

Групуємо на тетради:

0010 0010 0001 0000 0010 0001 0001 1010 = 2210211Ah.

Запишемо третє подання (0, -5, 82900):

7 десяткових розрядів мантиси мають вигляд 0082900. Найстарша цифра 0 буде відображена в полі ***G***, із шести інших формуємо тріади 082 та 900.

Для тріади 082: **BCD =** 0000 1000 0010. Одна велика цифри і дві малі, тому відповідний їм **DPD**-код становить 0000001010. Для тріади 900 аналогічно:

BCD = 1001 0000 0000. Одна велика цифри і дві малі, DPD-код становить 0010001100.

Експонента $e = -5$, тому зміщена експонента рівна $E = -5+101 = 96d = 0110\ 0000b$. Отже в поле E' записуємо молодших 6 бітів: 100000.

Формуємо поле G. Старші біти експоненти рівні 01, а старша цифра мантиси – 0 = 0000. Тому поле матиме вигляд 01000.

Записуємо поля S, G, E' та M':

0 01000 100000 0000001010 0010001100.

Групуємо на тетради:

0010 0010 0000 0000 0010 1000 1000 1100 = 2200288Ch.

Запишемо четверте подання (0, -6, 829000):

7 десяткових розрядів мантиси мають вигляд 0829000. Найстарша цифра 0 буде відображена в полі G, із шести інших формуємо тріади 829 та 000.

Для тріади 829: **BCD** = 1000 0010 1001. Дві великі цифри і одна мала, тому відповідний їм **DPD**-код становить 0100101111. Для тріади 000 аналогічно: **BCD** = 0000 0000 0000. Всі малі цифри, DPD-код становить 0000000000.

Експонента $e = -6$, тому зміщена експонента рівна $E = -6+101 = 95d = 0101\ 1111b$. Отже в поле E' записуємо молодших 6 бітів: 011111.

Формуємо поле G. Старші біти експоненти рівні 01, а старша цифра мантиси – 0 = 0000. Тому поле матиме вигляд 01000.

Записуємо поля S, G, E' та M':

0 01000 011111 0100101111 0000000000.

Групуємо на тетради:

0010 0001 1111 0100 1011 1100 0000 0000 = 21F4BC00h.

Запишемо п'яте подання (0, -7, 8290000):

7 десяткових розрядів мантиси мають вигляд 8290000. Найстарша цифра 8 буде відображена в полі G, із шести інших формуємо тріади 290 та 000.

Для тріади 290: **BCD** = 0010 1001 0000. Дві малі цифри і одна велика, тому відповідний їм **DPD**-код становить 0100011010. Для тріади 000 аналогічно: **BCD** = 0000 0000 0000. Всі малі цифри, DPD-код становить 0000000000.

Експонента $e = -7$, тому зміщена експонента рівна $E = -7+101 = 94d = 0101\ 1110b$. Отже в поле E' записуємо молодших 6 бітів: 011110.

Формуємо поле G. Старші біти експоненти рівні 01, а старша цифра мантиси – 8 = 1000. Ця цифра велика, тому поле матиме вигляд 11010.

Записуємо поля S, G, E' та M':

0 11010 011110 0100011010 0000000000.

Групуємо на тетради:

0110 1001 1110 0100 0110 1000 0000 0000 = 69E46800h.

Відповідь: Когорта числа **X** = +0,829d е форматі 32 біти десяткових чисел з рухомою комою містить п'ять членів:

(0, -3, 829) = 2220012Fh,
(0, -4, 8290) = 2210211Ah,
(0, -5, 82900) = 2200288Ch,
(0, -6, 829000) = 21F4BC00h,

(0, -7, 8290000). = 69E46800h.

4.12. Властивості десяткових чисел з рухомою комою

Розглянемо для прикладу десятковий формат довжиною 32 біти.

1. Знак займає старший біт подання.
2. Комбіноване поле ***G*** займає 5 наступних бітів.
3. Залишок мантиси займає останніх ***n*** = 20 бітів, а отже він подає $6 = 3 \times n/10$ десяткових цифр. Загальна кількість десяткових цифр всієї мантиси $k = 6 + 1 = 7$. Слід зауважити, що найбільше значення мантиси становить 9999999d, а найменше ненульове – 0000001d. Однак це найменше значення має всього одну значущу цифру подання і описує так звані субнормальні десяткові числа. Найменше значення мантиси, яке має $k = 7$ значущих цифр становить 1000000d.
3. Числа можуть бути нормальними та субнормальними. **Якщо когорта числа** (див. п. 4.11) **містить таке подання, мантиса якого є має *k* розрядів, то таке число називають нормальним. Якщо ж когорта не містить такого подання, то число вважають субнормальним**. Фактично будь яке нормальне число завжди можна подати у межах формату таким, що має ***k*** значущих цифр.
4. Залишок зміщеної експоненти займає ***m*** = 6 бітів, а отже повна зміщена експонента – 8 бітів. Мінімальне значення $E_{\min}$ становить 00000000 = 0, максимальне $E_{\max}$ – 10111111b = 191d. Зміщення ***bias*** обчислюють за формулою $bias = 3 \times 2^{m-1} + k - 2$. Для формату 32 біти $bias = 3 \times 2^{6-1} + 7 - 2 = = 101$. Отже, мінімальне значення зміщеної експоненти $E_{\min} = 0$ відповідає експоненті ***emin*** = -101d, а максимальне $E_{\max}$ = 191d – експоненті ***emax*** = +90d.

Отже, діапазон **нормальних** чисел, тобто таких, що мають ***k*** значущих цифр, для формату 32 біти (***k*** = 7) становить

від найбільшого $\mathbf{A_{max}} = \pm 9999999 \times 10^{+90} = \pm 9{,}999999 \times 10^{+96}$,

до найменшого $\mathbf{A_{min}} = \pm 1000000 \times 10^{-101} = \pm 1{,}000000 \times 10^{-95}$.

Цей приклад показує, що для **нормальних** чисел ***emin*** = -95, а ***emax*** = +96, тобто задовольняється умова стандарту IEEE754-2008: ***emin*** = 1 - ***emax*** (як і для двійкових чисел з рухомою комою).

Діапазон **субнормальних чисел**, тобто таких, що не можуть мати ***k*** значущих цифр, для формату 32 біти (***k*** = 7) становить

від найбільшого $A_{max} = \pm 0999999 \times 10^{-101} = \pm 9{,}99999 \times 10^{-96}$,

до найменшого $A_{min} = \pm 0000001 \times 10^{-101} = \pm 1 \times 10^{-101}$.

Звідси зрозуміло, чому субнормальне число не може містити члена когорти з ***k*** значущими цифрами: для цього довелось би зменшити значення експоненти, а воно і так має мінімальне (для субнормальних чисел) значення.

Абсолютна похибка **нормальних** чисел не перевищує ваги одиниці наймолодшого розряду мантиси і становить $\Delta = 0000001 \times 10^e$, а відносна –

$$\text{від } \delta_{\max} \approx \frac{10^e}{1000000 \times 10^e} = 10^{-6} \text{ до } \delta_{\min} \approx \frac{10^e}{9999999 \times 10^e} \approx 10^{-7}.$$

Порівняємо десяткові та двійкові числа з рухомою комою одного і того ж формату 32 біти (див. пп. 3.3.3, 3.4.2-3.4.4).

Для обох форматів кількість значущих цифр однакова і рівна 7.

Для бінарного формату діапазон нормальних чисел значно менший (від $A_{max} \approx \pm 3{,}40 \cdot 10^{38}$ до $A_{min} \approx \pm 1{,}17 \cdot 10^{-38}$) порівняно з десятковим форматом (від $A_{max} \approx \pm 10^{+97}$ до $A_{min} = \pm 10^{-95}$).

Відносна похибка подання бінарних чисел з рухомою комою на порядок менша ніж для десяткових (для бінарних – максимальна $\delta \approx 1{,}19 \cdot 10^{-7}$).

Перевагою десяткових чисел з рухомою комою є абсолютна точність подання реальних десяткових чисел, кількість значущих цифр яких не перевищує максимально можливу для заданого подання.

Таким чином, десяткові числа з рухомою комою перспективніші, ніж бінарні числа з рухомою комою, оскільки мають більше переваг. Недоліком можна вважати складність подання, що з огляду на швидкий розвиток обчислювальної техніки є тимчасовою проблемою, яку найближчим часом буде вирішено.

4.13. Формати десяткових чисел з рухомою комою стандарту IEEE754-2008

Подібно до бінарних форматів стандарт IEEE754-2008 специфікував також і десяткові формати чисел з рухомою комою. Це три базові формати Decimal 32, Decimal 64 та Decimal 128, а також сімейство множинних форматів, загальна довжина яких $W = 32 \times N$, де N – будь-яке натуральне число. Отже можливі такі довжини форматів: $32 \cdot 1 = 32$ біти, $32 \cdot 2 = 64$ біти, $32 \cdot 3 = 96$ бітів, $32 \cdot 4 = 128$ бітів, $32 \cdot 5 = 160$ бітів, $32 \cdot 6 = 192$ біти, $32 \cdot 7 = 224$ біти, $32 \cdot 8 = 256$ бітів тощо. Таким чином, усі базові формати фактично входять у сімейство множинних. Табл. 4.5. демонструє можливі формати десяткових чисел з рухомою комою.

Таблиця 4.5. Параметри десяткових форматів IEEE754-2008-чисел

Параметр	Decimal 32 біти	Decimal 64 біти	Decimal 128 бітів	Decimal W бітів ($W \geq 32$)
W, загальна довжина формату	32	64	128	W кратне 32 бітам
m+2, довжина експоненти	8	10	14	W/16+6
n, довжина залишку мантиси	20	50	110	15×W/16-10
k, точність подання в десяткових розрядах	7	16	34	9×W/32-2
$emax$, максимальна експонента	96	384	6176	$3 \times 2^{W/16+3}$
$emin$, мінімальна експонента	-95	-383	-6176	$1-3 \times 2^{W/16+3}$
$bias$, зміщення	101	398	6208	$emax$+k-2
максимальне число A_{max}	$\pm 10^{96}$	$\pm 10^{384}$	$\pm 10^{6176}$	$10\text{^}(3 \times 2^{W/16+3})$

Таким чином, стандарт IEEE754-2008 описує широкий набір як бінарних, так і десяткових форматів чисел з рухомою комою, які можна використовувати для організації обчислень з різною точністю та діапазоном подання чисел.

4.14. Реалізація десяткових чисел з рухомою комою в сучасних процесорах

Арифметика десяткових чисел з рухомою комою реалізована в деяких сучасних процесорах. Першим таким процесором став POWER6 – продовження лінійки POWER компанії IBM [18]. Офіційно процесор було оголошено у травні 2007 р. Процесор має два ядра, кожне з яких здатне виконувати два потоки команд одночасно і кожне з яких включає два блоки арифметики цілих чисел, два блоки двійкових обчислень з рухомою комою та один блок десяткових обчислень з рухомою комою. Для реалізації десяткових обчислень задіяні 50 нових команд, які здійснюють математичні операції та переведення з двійкових чисел з рухомою комою в десяткові числа з рухомою комою та навпаки.

4.15. Особливості здійснення арифметичних дій над десятковими числами з рухомою комою у стандарті IEEE754-2008

Арифметичні дії над десятковими числами з рухомою комою здійснюють подібно до алгоритмів аналогічних дій над двійкових чисел з рухомою комою з невеликими відмінностями. Є три основні відмінності:

Перша відмінність полягає в тому, що десяткове число з рухомою комою може мати декілька подань, а отже арифметика таких чисел передбачає не тільки обчислення достовірного числового результату, але й вибір правильного члена когорти результату обчислень. Правила вибору члена когорти описано в п. 4.11.

Друга відмінність полягає в тому, що мантиса десяткових чисел є ціле додатне число, а отже така операція як нормалізація результату обчислень є незастосовна для десяткових чисел з рухомою комою. Замість неї може бути застосована операція вирівнювання, яка полягає у виборі такого члена когорти, що має оптимальну експоненту (див. п. 4.11). Операція вирівнювання є не обов'язковою після виконання арифметичних дій на відміну від операції нормалізації мантиси для двійкових чисел з рухомою комою.

Третя відмінність полягає у здійсненні округлення результату обчислень. Як було описано в п. 3.6.3 стандартом IEEE754-1985 передбачено чотири альтернативних підходи до виконання округлення:

1) округлення до найближчого числа, яке можна подати у використовуваному форматі (округлення до парного у випадку двох найближчих чисел);
2) округлення до $+\infty$;
3) округлення до $-\infty$;
4) округлення до нуля.

Стандарт IEEE754-2008 передбачає такі округлення як для двійкових, так і для десяткових чисел з рухомою комою. Крім того, вводиться ще один спосіб

округлення тільки для десяткових чисел з рухомою комою, а саме **округлення до більшого модуля (roundTiesToAway)**. Таке округлення полягає у виборі такого число з рухомою комою, значення якого ближче до точного результату обчислення, однак, якщо існують два числа з рухомою комою, однаково близькі до точного результату, то вибирають таке з них, **величина якого за модулем є більшою**. Таке округлення нам відоме як звичайне округлення десяткових чисел. Наприклад, +37,5 буде округлене до +38, а 37,4 – до 37. Однак **для від'ємних чисел ми маємо навпаки**: -37,5 буде округлене до -38, хоча мало би бути округлене до -37. Округлення до більшого модуля прямо протилежне округлюванню до нуля (**roundTowardZero**). Тому термін **roundTiesToAway** фактично є **округлюванням від нуля**. Ще однією особливістю стандарту IEEE754-2008 є те, що **за замовчуванням** він передбачає округлення до найближчого (парного) як для двійкових, так і для десяткових чисел з рухомою комою, хоча для останніх вибір способу округлення результату в основному визначається мовами програмування.

Приклад 4.3. Число **X** = -(1/7) записати у форматі десяткових чисел з рухомою комою одинарної точності стандарту IEEE754-2008, використовуючи округлення до більшого модуля та до парного (за замовчуванням).

Розв'язування:

$$-(1/7) = -0{,}142857142857142.....$$

Таке число не допускає точного подання у форматі чисел з рухомою комою, тому його необхідно округлити до $k = 7$ значущих цифр. Отже маємо число $-1428571{,}4285...\times10^{-7}$. Округлення до більшого модуля дає -1428571×10^{-7}, а округлення до парного – -1428572×10^{-7}. Запишемо подання цих чисел.

1) Знаковий біт рівний 1 для обох подань ($S = 1$).

2) 7 десяткових розрядів мантиси мають вигляд 1428571 (або 1428572). Найстарша цифра 1 буде відображена в полі ***G***, із шести інших формуємо тріади 428 та 571 (572).

Для тріади 428: **BCD** = 0100 0010 1000. Дві малі цифри і одна велика, **DPD**-код становить 1000101000.

Для тріади 571: **BCD** = 0101 0111 0001. Три малі цифри, DPD-код становить 1011110001.

Для тріади 572: **BCD** = 0101 0111 0010. Три малі цифри, DPD-код становить 1011110010.

3) Експонента $e = -7$, тому зміщена експонента рівна $E = -7+101 = 94d =$ = 0101 1110b. Отже в поле E' записуємо молодших 6 бітів: 011110.

Формуємо поле ***G***. Старші біти експоненти рівні 01, а старша цифра мантиси – 1 = 0001. Ця цифра мала, тому поле матиме вигляд 01001.

Записуємо поля ***S***, ***G***, E' та M':

-1428571×10^{-7} = 1 01001 011110 1000101000 1011110001.

-1428572×10^{-7} = 1 01001 011110 1000101000 1011110010

Групуємо на тетради:

1010 0101 1110 1000 1010 0010 1111 0001 = A5E8A2F1h

1010 0101 1110 1000 1010 0010 1111 0010 = A5E8A2F2h

Відповідь: -1428571×10^{-7} = A5E8A2F1h, -1428572×10^{-7} = A5E8A2F2h.

4.15.1. Додавання та віднімання

Алгоритми додавання (віднімання) двійкових та десяткових чисел з рухомою комою загалом подібні (з урахуванням відмінностей, перерахованих у п. 4.15). Додавання (віднімання) десяткових чисел з рухомою комою також починають з перевірки на специфічні значення операндів (див. рис. 3.4). Однак зсув мантис для вирівнювання порядків у разі незбігу експонент здійснюють дещо іншим способом. Оскільки (див. п. 4.11) експонента результату повинна бути рівна оптимальній експоненті (або якомога ближче до неї), а оптимальна експонента є найменшою з експонент операндів, то очевидно, що слід зсувати мантису операнду з більшою експонентою. Такий зсув здійснюють **вліво** (на відміну від двійкових чисел з рухомою комою, де зсувають мантису вправо). Враховуючи (4.14), додавання (віднімання) можна описати таким чином:

$$\mathbf{X} \pm \mathbf{Y} = (-1)^{S_x} \cdot 10^{E_x\text{-bias}} \cdot M_x \pm (-1)^{S_y} \cdot 10^{E_y\text{-bias}} \cdot M_y =$$

$$= 10^{E_x\text{-bias}} \cdot [(-1)^{S_x} \cdot M_x \pm (-1)^{S_y} \cdot 10^{E_y-E_x} \cdot M_y]. \qquad (4.15)$$

Якщо $E_y > E_x$, тоді E_x є оптимальною експонентою. У цьому випадку $E_y - E_x > 0$, а вираз $10^{E_y-E_x} \cdot M_y$ у формулі (4.15) фактично означає, що мантису операнду **Y** слід змістити вліво на $E_y - E_x$ розрядів (помножити на $10^{E_y-E_x}$). Якщо ж $E_x > E_y$, тоді E_y є оптимальною експонентою і слід змістити вліво на $E_x - E_y$ розрядів мантису операнду **X**.

Такі зміщені (зсунені) мантиси додають (віднімають) з урахуванням знаку операндів. Додавання (віднімання) десяткових чисел з рухомою комою також може призвести до **втрати вагомості мантиси** (див. 3.6.1). Однак тут слід враховувати не тільки кількість розрядів, на які зміщують мантису порівняно з k (k – точність подання в десяткових розрядах для вибраного формату, див. табл. 4.5), але і кількість значущих цифр у поданні кожного з операндів k_x та k_y (нулі справа від ненульової цифри також вважають значущими).

Загалом випадку втрата вагомості мантиси можлива, коли

$$\begin{cases} E_x - E_y = e_x - e_y \geq k + k_y - k_x, & \text{для } \mathbf{E_x > E_y}, \qquad (4.16, \text{а}) \\ E_y - E_x = e_y - e_x \geq k + k_x - k_y, & \text{для } \mathbf{E_y > E_x}. \qquad (4.16, \text{б}) \end{cases}$$

За умови, що один операнд має максимальну кількість значущих цифр (рівну k), а другий – мінімальну (рівну 1) втрата значимості мантиси буде однозначно відбуватись, якщо $|E_x - E_y| = |e_x - e_y| \geq 2\times k - 1$. Врахувавши також правила округлення, отримуємо дещо більше значення максимальної різниці експонент: $|E_x - E_y| = |e_x - e_y| \geq 2\times k$. Наприклад, якщо додати два числа у форматі 32 біти **X** = (0, -6, 4000000) = +4 та **Y** = (0, 2, 4) = +400, то згідно з (4.16, б) втрати значимості мантиси не буде, оскільки $e_y - e_x = 2 - (-6) = 8 < 7 + 7 - 1 = 13$. Результат додавання становить **X** + **Y** = +404, а враховуючи те, що слід вибрати оптимальну експоненту або найближчу до неї, отримуємо: **X** + **Y** = (0, -4, 4040000).

Отримана результуюча мантиса в (4.15) $\mathbf{[(-1)^{Sx} \cdot M_x \pm (-1)^{Sy} \cdot 10^{Ey-Ex} \cdot M_y]}$ може містити різну кількість значущих цифр k_{x+y}, у тому числі значущі нулі. Якщо $k_{x+y} \le k$, то таку мантису не округлюють, її вважають мантисою результату. В цьому випадку експонентою результату вважають найменшу з експонент (оптимальну експоненту) операндів.

Якщо $k_{x+y} > k$, то таку мантису завжди округлюють до k значущих цифр. Оскільки результатом додавання (віднімання) зміщеної та незміщеної мантис операндів завжди є **ціле число**, то округлюючи справа $k_{x+y} - k$ розрядів (фактично відкидаючи їх з округленням) слід одночасно збільшити значення оптимальної експоненти результату на кількість округлених розрядів $k_{x+y} - k$. Тому експонента результату буде рівна $min(E_x, E_y) + k_{x+y} - k$. Така експонента буде найближчою до оптимальної, як і передбачено стандартом IEEE754-2008. Знак суми (різниці) визначають за результатом додавання (віднімання) мантис.

Додавання (віднімання) десяткових чисел з рухомою комою може мати результатом помилки переповнення та недоповнення порядку аналогічно, як і для двійкових чисел з рухомою комою. Результатом операції в цьому випадку також є sNaN (див. п.3.6.1).

Приклад 4.4. Додати два десяткові числа з рухомою комою формату 32 біти **X** = A2500113h, та **Y** = 2A160000h.

Розв'язування: Запишемо числа побітно, визначимо знаки, експоненти та мантиси операндів:

X = A2500113h = 1010 0010 0101 0000 0000 0001 0001 0011 b;

S_x = 1 – число від'ємне;

G_x = 01000.

Оскільки перші два біти одночасно не рівні "1", то вони подають перших два біти зміщеної експоненти, а перша цифра мантиси є малою (0 – 7) і визначається наступними трьома бітами поля G_x (див. табл. 4.4). Тому першою цифрою мантиси є "0".

E'_x = 100101.

Врахувавши перших два біти з області G_x, отримуємо E_x = 01100101.

M'_x = 0000000000 0100010011.

Перша двійкова декада 0000000000 однозначно репрезентує десяткову тріаду "000" (див. табл. 4.3.).

Друга двійкова декада становить 0100010011. Спочатку перевіряємо значення третього біту декади (нумерація справа наліво починаючи з нуля). Оскільки біт b3 становить "0", то маємо випадок усіх трьох малих цифр. Тому десяткова тріада становить "213".

У кінцевому результаті мантиса числа **X** становить: M_x = 0000213.

Для числа **Y** маємо:

Y = 2A160000h = 0010 1010 0001 0110 0000 0000 0000 0000 b;

S_y = 0 – число додатне;

G_y = 01010.

Оскільки перші два біти одночасно не рівні "1", то вони подають перших два біти зміщеної експоненти, а перша цифра мантиси є малою (0 – 7) і визначається наступними трьома бітами поля $\boldsymbol{G}_{\mathbf{y}}$ (див. табл. 4.4). Тому перша цифра мантиси є "2".

$\boldsymbol{E'}_{\mathbf{y}}$ = 100001.

Врахувавши перших два біти з області $\boldsymbol{G}_{\mathbf{y}}$, отримуємо $\boldsymbol{E}_{\mathbf{y}}$ = 01100001.

$\boldsymbol{M'}_{\mathbf{y}}$ = 0110000000 0000000000

Перша двійкова декада становить 0110000000. Перевіряємо значення третього біту декади. Оскільки він становить "0", то маємо випадок усіх трьох малих цифр. Тому перша десяткова тріада становить "300".

Друга двійкова декада 0000000000 однозначно репрезентує десяткову тріаду "000".

У кінцевому результаті мантиса числа **Y** становить: $\boldsymbol{M}_{\mathbf{y}}$ = 2300000.

Здійснимо додавання чисел. Порівнюємо зміщені експоненти: $\boldsymbol{E}_{\mathbf{x}} > \boldsymbol{E}_{\mathbf{y}}$, оскільки 01100101 > 01100001, тому експонента $\boldsymbol{E}_{\mathbf{y}}$ = 01100001 є оптимальною, а зміщувати вліво слід мантису $\boldsymbol{M}_{\mathbf{x}}$ на 01100101b - 01100001b = 100b = 4d розрядів.

З урахуванням знаків чисел маємо:

$\boldsymbol{M}_{\mathbf{x}} \times 10^3$	=	- 0000213	
$\boldsymbol{M}_{\mathbf{y}}$	=	+ 2300000	
$\boldsymbol{M}_{\mathbf{x}} \times 10^3 + \boldsymbol{M}_{\mathbf{y}}$	=	+ 00000170000	результат додатний.

Такий результат містить 6 значущих цифр (враховуючи нулі справа). Очевидно, що можна обійтися без округлення. Таким чином, мантиса результату становить $\boldsymbol{M}_{\mathbf{x+y}}$ = 0170000, зміщена експонента рівна оптимальній експоненті та становить $\boldsymbol{E}_{\mathbf{x+y}}$ = 01100001b. Подамо результат додавання у форматі десяткових чисел з рухомою комою.

7 десяткових розрядів мантиси мають вигляд 0170000. Найстарша цифра 0 буде відображена в полі $\boldsymbol{G}$, із шести інших формуємо тріади 170 та 000.

Для тріади 170: **BCD =** 0001 0111 0000. Всі малі цифри, тому відповідний їм **DPD**-код становить 0011110000. Для тріади 000 аналогічно: **DPD**-код становить 0000000000.

В поле $\boldsymbol{E'}$ записуємо молодших 6 бітів зміщеної експоненти: 100001.

Формуємо поле $\boldsymbol{G}$. Старші біти експоненти рівні 01, а старша цифра мантиси – 0d = 0000b. Тому поле буде мати вигляд 01000.

Записуємо поля $\boldsymbol{S}$, $\boldsymbol{G}$, $\boldsymbol{E'}$ та $\boldsymbol{M'}$:

0 01000 100001 0011110000 0000000000.

Групуємо на тетради:

0010 0010 0001 0011 1100 0000 0000 0000 = 2213C000h.

Відповідь: **X** + **Y** = A2500113h + 2A160000h = 2213C000h.

Приклад 4.5. Додати два десяткові числа з рухомою комою формату 32 біти **X** = 2260024Fh, та **Y** = 220D6D95h.

Розв'язування: Запишемо числа побітно, визначимо знаки, експоненти та мантиси операндів:

X = 2260024Fh = 0010 0010 0110 0000 0000 0010 0100 1111 b;

$\boldsymbol{S}_{\mathbf{x}}$ = 0 – число додатне;

$\boldsymbol{G}_{\mathbf{x}}$ = 01000.

Оскільки перші два біти одночасно не рівні "1", то вони подають перших два біти зміщеної експоненти, а перша цифра мантиси є малою (0 – 7) і визначається наступними трьома бітами області $\boldsymbol{G}_{\mathbf{x}}$ (див. табл. 4.4). Тому перша цифра мантиси є "0".

$\boldsymbol{E'}_{\mathbf{x}}$ = 100110.

Врахувавши перших два біти з області $\boldsymbol{G}_{\mathbf{x}}$, отримуємо $\boldsymbol{E}_{\mathbf{x}}$ = 01100110.

$\boldsymbol{M'}_{\mathbf{x}}$ = 0000000000 1001001111.

Перша двійкова декада 0000000000 однозначно репрезентує десяткову тріаду "000".

Друга двійкова декада становить 1001001111. Спочатку перевіряємо значення третього біту декади (нумерація справа наліво починаючи з нуля). Оскільки біт b3 становить "1", то перевіряємо біти b2 та b1. Вони також становлять "1". У цьому випадку слід перевірити біти b6 та b5, які становлять "1" та "0", відповідно. Така комбінація ключових бітів вказує, що перша цифра тріади мала, а дві наступні – великі. Згідно з табл. 4.3 визначаємо, що тріада становить "489".

У кінцевому результаті мантиса числа ***X*** становить: $\boldsymbol{M}_{\mathbf{x}}$ = 0000489.

Для числа **Y** маємо:

Y = 220D6D95h = 0010 0010 0000 1101 0110 1101 1001 0101 b;

$\boldsymbol{S}_{\mathbf{y}}$ = 0 – число додатне;

$\boldsymbol{G}_{\mathbf{y}}$ = 01000.

Оскільки перші два біти одночасно не рівні "1", то вони подають перших два біти зміщеної експоненти, а перша цифра мантиси є малою (0 – 7) і визначається наступними трьома бітами області $\boldsymbol{G}_{\mathbf{y}}$ (див. табл. 4.4). Тому перша цифра мантиси є "0".

$\boldsymbol{E'}_{\mathbf{y}}$ = 100000.

Врахувавши перших два біти з області $\boldsymbol{G}_{\mathbf{y}}$, отримуємо $\boldsymbol{E}_{\mathbf{y}}$ = 01100000.

$\boldsymbol{M'}_{\mathbf{y}}$ = 1101011011 0110010101

Перша двійкова декада становить 1101011011. Спочатку перевіряємо значення третього біту декади (нумерація справа наліво починаючи з нуля). Оскільки біт b3 становить "1", то перевіряємо біти b2 та b1. Вони також становлять "0" та "1", відповідно. Така комбінація ключових бітів вказує, що перша і третя цифри тріади малі, а друга – велика. Відповідно до табл. 4.3 визначаємо, що тріада становить "695".

Друга двійкова декада становить 0110010101. Спочатку перевіряємо значення третього біту декади (нумерація справа наліво починаючи з нуля). Оскільки біт b3 становить "0", то всі цифри десяткової тріади є малі. Відповідно до табл. 4.3 визначаємо, що тріада становить "315".

У кінцевому результаті мантиса числа **Y** становить: $\boldsymbol{M}_{\mathbf{y}}$ = 0695315.

Здійснимо додавання чисел:

Порівнюємо зміщені експоненти: $\boldsymbol{E}_{\mathbf{x}} = 01100110$ $\boldsymbol{E}_{\mathbf{y}} = 01100000$. $\boldsymbol{E}_{\mathbf{x}} > \boldsymbol{E}_{\mathbf{y}}$, оскільки 01100110 > 01100000, тому експонента $\boldsymbol{E}_{\mathbf{y}} = 01100000$ є оптимальною, а зміщувати вліво слід мантису $\boldsymbol{M}_{\mathbf{x}}$ на 01100110b - 01100000b = 110b = 6d розрядів.

З урахуванням знаків чисел маємо:

$\boldsymbol{M}_{\mathbf{x}} \times 10^3$	=	+ 0000489	
$\boldsymbol{M}_{\mathbf{y}}$	=	+ 0695315	
$\boldsymbol{M}_{\mathbf{x}} \times 10^3 + \boldsymbol{M}_{\mathbf{y}}$	=	+ 0000489695315	результат додатний.

Такий результат містить 9 значущих цифр. Оскільки формат 32-біти передбачає максимум 7 значущих цифр, то результат слід округлити на два розряди, одночасно збільшивши експоненту результату на 2. Округлення до більшого модуля (як і округлення до парного) дасть результуючу мантису $\boldsymbol{M}_{\mathbf{x+y}} = 4896953$, а зміщена експонента буде становити $\boldsymbol{E}_{\mathbf{x+y}} = \boldsymbol{E}_{\mathbf{y}} + 2 =$ = 01100000b + 10b = 01100010b. Подамо результат додавання у форматі десяткових чисел з рухомою комою.

7 десяткових розрядів мантиси результату мають вигляд 4896953. Найстарша цифра 4 буде відображена в полі ***G***, із шести інших формуємо тріади 896 та 953.

Для тріади 896: **BCD** = 1000 1001 0110. Дві великі цифри і одна мала, тому відповідний їм **DPD**-код становить 1100011110.

Для тріади 953: **BCD** = 1001 0101 0011. Дві малі цифри і одна велика, тому відповідний їм **DPD**-код становить 0111011101.

В поле ***E′*** записуємо молодших 6 бітів зміщеної експоненти: 100010.

Формуємо поле ***G***. Старші біти експоненти рівні 01, а старша цифра мантиси – 4d = 0100b. Тому поле буде мати вигляд 01100.

Записуємо поля ***S***, ***G***, ***E′*** та ***M′***:

0 01100 100010 1100011110 0111011101.

Групуємо на тетради:

0011 0010 0010 1100 0111 1001 1101 1101 = 322C7DDh.

Відповідь: **X** + **Y** = 2260024Fh + 220D6D95h = 322C7DDh.

4.15.2. Множення та ділення

Виконання операцій множення і ділення у форматі десяткових чисел з рухомою комою здійснюється ще простіше, ніж додавання чи віднімання таких чисел і практично збігається з алгоритмами множення (рис. 3.5) та ділення (рис. 3.6) для двійкових чисел з рухомою комою. За винятком відмінностей, перерахованих у п. 4.15, додатковою є відмінність здійснення додавання (у випадку множення) та віднімання (у випадку ділення) експонент операндів.

Добуток операндів **X** та **Y** визначається згідно з:

$$\mathbf{X} \times \mathbf{Y} = [(-1)^{S\mathrm{x}} \cdot 10^{E\mathrm{x\text{-}bias}} \cdot \boldsymbol{M}_{\mathbf{x}}] \times [(-1)^{S\mathrm{y}} \cdot 10^{E\mathrm{y\text{-}bias}} \cdot \boldsymbol{M}_{\mathbf{y}}] =$$

$$= (-1)^{S\mathrm{x}+S\mathrm{y}} \cdot 10^{E\mathrm{x\text{-}bias}+E\mathrm{y\text{-}bias}} \cdot (\boldsymbol{M}_{\mathbf{x}} \times \boldsymbol{M}_{\mathbf{y}}), \qquad (4.17)$$

а частка -

$$\mathbf{X / Y = [(-1)^{Sx} \cdot 10^{Ex\text{-}bias} \cdot M_x] / [(-1)^{Sy} \cdot 10^{Ey\text{-}bias} \cdot M_y] =}$$

$$\mathbf{= (-1)^{Sx\text{-}Sy} \cdot 10^{Ex\text{-}Ey} \cdot (M_x / M_y)}. \qquad (4.18)$$

У випадку двійкових чисел з рухомою комою експоненти подані в зміщеному коді з від'ємним нулем, а додавання чи віднімання таких експонент виконують згідно правил додавання в такому коді (див. розділ 3.4.2). Для зміщеного коду, в якому подані експоненти десяткових чисел з рухомою комою, ці правила незастосовні, оскільки величина зміщення має специфічне значення $\boldsymbol{bias} = 3 \times 2^{m-1} + \boldsymbol{k} - 2$ (див. п. 4.12.). Тому додавання зміщених експонент здійснюють відповідно до формули

$$\boldsymbol{E_{x\times y} = e_{x\times y} + bias = e_x + e_y + bias =}$$

$$\mathbf{= (E_x - bias) + (E_y - bias) + bias = E_x + E_y - bias}, \qquad (4.19)$$

а віднімання – відповідно до

$$\boldsymbol{E_{x/y} = e_{x/y} + bias = e_x - e_y + bias =}$$

$$\mathbf{= (E_x - bias) - (E_y - bias) + bias = E_x - E_y + bias}. \qquad (4.20)$$

Отже, у випадку множення від суми зміщених експонент треба відняти зміщення, а у разі ділення – навпаки: до різниці зміщених експонент треба додати зміщення.

Як і у випадку додавання (віднімання) вибір члена когорти результату обчислення здійснюють згідно з правилами, описаних в п. 4.11, тобто вибирають таке подання числа, експонента якого рівна оптимальній або якомога ближче до неї.

Множення і ділення мантис, правила вибору знаку результату, повідомлення про помилки переповнення та недоповнення порядку подібні до описаних в пп. 3.6.1. та 3.6.2.

Приклад 4.6. Знайти добуток двох десяткових чисел з рухомою комою формату 32 біти **X** = 2A160000h, та **Y** = 220D6D95h. У цьому прикладі з метою зменшення об'єму обчислень, число **X** взято з попереднього прикладу 4.4, а число **Y** – з прикладу 4.5, тому:

$\boldsymbol{X}$ = 2A160000h = 0010 1010 0001 0110 0000 0000 0000 0000 b;
$\boldsymbol{S_x}$ = 0 число додатне;
$\boldsymbol{E_x}$ = 01100001 $\boldsymbol{M_x}$ = 2300000.

$\boldsymbol{Y}$ = 220D6D95h = 0010 0010 0000 1101 0110 1101 1001 0101 b;
$\boldsymbol{S_y}$ = 0 число додатне;
$\boldsymbol{E_y}$ = 01100000 $\boldsymbol{M_y}$ = 0695315.

Знак добутку однозначно визначається знаками співмножників, отже
$\boldsymbol{S_{x\times y}}$ = 0 добуток додатний.

Оптимальна експонента добутку становить $\boldsymbol{E}_{\mathbf{y}} = 01100000\text{b}$. Визначимо зміщену експоненту добутку. Для цього додамо зміщені експоненти множників та віднімемо від їхньої суми величину зміщення, яка для формату 32-біти становить $101\text{d} = 01100101\text{b}$ (див. табл. 4.5). (**Додавання та віднімання здійснюємо у прямому коді!**).

$$
\begin{array}{rr}
\boldsymbol{E}_{\mathbf{x}} = & 01100001\text{b} \\
\boldsymbol{E}_{\mathbf{y}} = & + \underline{01100000\text{b}} \\
 & 11000001\text{b} \\
\boldsymbol{bias} = & - \underline{01100101\text{b}} \\
\boldsymbol{E}_{\mathbf{x}} + \boldsymbol{E}_{\mathbf{y}} - \boldsymbol{bias} = & 01011100\text{b}
\end{array}
$$

Здійснимо множення мантис безпосередньо в десятковій системі числення:

$\boldsymbol{M}_{\mathbf{x}} \times \boldsymbol{M}_{\mathbf{y}} = 2300000 \times 0695315 = 1599224500000.$

Такий результат містить 13 значущих цифр, у тому числі нулі. Формат 32-біти передбачає максимум 7 значущих цифр, отже результат слід округлити на $13 - 7 = 6$ розрядів. Одночасно необхідно збільшити суму зміщених експонент множників на 6, тому зміщена експонента добутку становитиме $\boldsymbol{E}_{\mathbf{x}\times\mathbf{y}} = \boldsymbol{E}_{\mathbf{x}} + \boldsymbol{E}_{\mathbf{y}} - \boldsymbol{bias} + 6\text{d} = 01011100\text{b} + 110\text{b} = 01100010\text{b}$.

Необхідність округлення ненульових розрядів також означає, що **оптимальна експонента не може бути використана** як експонента результату, тому слід використовувати зміщену експоненту добутку.

Результат округлення для даного випадку буде залежати від виду округлення. Округлення до більшого модуля передбачає, що мантиса результату становитиме $\boldsymbol{M}_{\mathbf{x}\times\mathbf{y}} = 1599225$, оскільки число 1599224,500000 однаково близьке як до 1599224,000000, так і до 1599225,000000, а округлення до парного дасть результуючу мантису $\boldsymbol{M}_{\mathbf{x}\times\mathbf{y}} = 1599224$.

Подамо результат додавання в форматі десяткових чисел з рухомою комою для обох варіантів округлення.

7 десяткових розрядів мантиси результату мають вигляд 1599224 або 1599225. Найстарша цифра 1 буде відображена в полі ***G***, із шести інших формуємо тріади 599 та 224 або 225.

Для тріади 599: **BCD** = 0101 1001 1001. Дві малі цифри і одна велика, тому відповідний їм **DPD**-код становить 1011011111.

Для тріади 224: **BCD** = 0010 0010 0100. Всі малі цифри, тому відповідний їм **DPD**-код становить 0100100100.

Для тріади 225: **BCD** = 0010 0010 0101. Всі малі цифри, тому відповідний їм **DPD**-код становить 0100100101.

У поле ***E′*** записуємо 6 молодших бітів зміщеної експоненти: 100010.

Формуємо поле ***G***. Старші біти експоненти рівні 01, а старша цифра мантиси – $1\text{d} = 0001\text{b}$. Тому поле буде мати вигляд 01001.

Записуємо поля ***S***, ***G***, ***E′*** та ***M′***:

Для $\boldsymbol{M}_{\mathbf{x}\times\mathbf{y}} = 1599224$

0 01001 100010 1011011111 0100100100.

Групуємо на тетради:

0010 0110 0010 1011 0111 1101 0010 0100 = 262B7B24h.

Для $M_{x \times y}$ = 1599225

0 01001 100010 1011011111 0100100101.

Групуємо на тетради:

0010 0110 0010 1011 0111 1101 0010 0101 = 262B7B25h.

Відповідь: **X** × **Y** = 2A160000h × 220D6D95h = 262B7B24h (округлення до парного) або 262B7B25h (округлення до більшого модуля).

Приклад 4.7. Поділити два десяткові числа з рухомою комою формату 32 біти **X** = 322C7DDh на **Y** = 2213C000h. У цьому прикладі також використано числа **X** та **Y** попередніх прикладів 4.4 та 4.5, тому:

X = 322C7DDh = 0011 0010 0010 1100 0111 1001 1101 1101.
S_x = 0 число додатне;
E_x = 01100010 M_x = 4896953.

Y = 2213C000h = 0010 0010 0001 0011 1100 0000 0000 0000;
S_y = 0 число додатне;
E_y = 01100001 M_y = 0170000.

Знак добутку однозначно визначається знаками співмножників, отже
$S_{x/y}$ = 0 добуток додатний.

Оптимальна експонента частки рівна E_y = 01100001b. Визначимо зміщену експоненту частки. Для цього від зміщеної експоненти діленого віднімемо зміщену експоненту дільника та додамо до їхньої суми величину зміщення, яке для формату 32-біти становить 101d = 01100101b (див. табл. 4.5). (**Додавання та віднімання здійснюємо в прямому коді!**).

E_x = 01100010b
E_y = - 01100001b
00000001b
bias = + 01100101b
$E_x - E_y + bias$ = 01100110b

Здійснимо ділення мантис безпосередньо в десятковій системі числення

M_x / M_y = 4896953 / 0170000 = 28,8056058823...

Такий результат містить нескінченну кількість значущих цифр. Формат 32-біти передбачає максимум 7 значущих цифр, отже результат слід округлити до семи розрядів. Округлення до більшого модуля (як і до парного) дає результат 28,80561. Оскільки мантиса повинна бути цілим додатним числом, то результат слід помножити на 10^5 одночасно зменшивши різницю зміщених експонент множників на 5. Тому мантиса результату становитиме $M_{x/y}$ = 2880561, а

зміщена експонента частки – $\boldsymbol{E_{x/y}} = \boldsymbol{E_x} - \boldsymbol{E_y} + \boldsymbol{bias}$ - 5d = 01100110b - 101b = = 01100001b.

Необхідність округлення ненульових розрядів також означає, що **оптимальна експонента не може бути використана** як експонента результату, тому слід використовувати зміщену експоненту частки.

Подамо результат додавання у форматі десяткових чисел з рухомою комою для обох варіантів округлення.

7 десяткових розрядів мантиси результату мають вигляд 2880561. Найстарша цифра 2 буде відображена в полі ***G***, із шести інших формуємо тріади 880 та 561.

Для тріади 880: **BCD** = 1000 1000 0000. Дві великі цифри і одна мала, тому відповідний їм **DPD**-код становить 0000001110.

Для тріади 561: **BCD** = 0101 0110 0001. Всі малі цифри, тому відповідний їм **DPD**-код становить 1011100001.

У поле ***E'*** записуємо 6 молодших бітів зміщеної експоненти: 100001.

Формуємо поле ***G***. Старші біти експоненти рівні 01, а старша цифра мантиси – 2d = 0010b. Тому поле буде мати вигляд 01010.

Записуємо поля ***S***, ***G***, ***E'*** та ***M'***:

0 01010 100001 0000001110 1011100001.

Групуємо на тетради:

0010 1010 0001 0000 0011 1010 1110 0001 = 2A103AE1h.

Відповідь: $\boldsymbol{X} / \boldsymbol{Y} = \boldsymbol{X}$ = 322C7DDh / 2213C000h = 2A103AE1h.

4.16. Висновки до розділів 3 і 4

Розглянувши способи подання дійсних чисел за допомогою скінченної кількості двійкових розрядів, так звані числа з рухомою комою, можна зробити ряд висновків щодо переваг і недоліків такого способу подання чисел.

1. Двійкові числа з рухомою комою передбачають набагато простіший спосіб реалізації арифметичних дій за допомогою апаратних засобів, оскільки не потребують додаткового кодування, як у випадку десяткових чисел з рухомою комою. До того ж арифметичні дії над двійковими числами з рухомою комою здійснюються швидше, ніж над десятковими.

2. Недоліком двійкових чисел з рухомою комою є складність переведення їх у десяткову систему числення з метою подання результатів обчислень у звичній для нас формі. У десяткових чисел з рухомою комою такий недолік відсутній.

3. Двійкові числа з рухомою комою передбачають єдине подання будь-якого числа, яке не виходить за межі діапазону подання. Десяткові числа з рухомою комою можуть мати декілька подань числа, сукупність яких називають когортою. Наприклад, число нуль має 2 подання для двійкових чисел з рухомою комою, та $2\times(2^{m+2}-2^{m})$ подань для десяткових чисел з рухомою комою. З огляду на це, ефективність використання коду для двійкових чисел з рухомою комою є максимальною. Це означає, що кількість скінченних чисел, які можна

подати для обох видів чисел з рухомою комою одного і того ж формату (бітової довжини) є набагато більшою для двійкових чисел, ніж для десяткових.

4. Обчислення за допомогою чисел з рухомою комою є наближеними. Це стосується також тих випадків, коли звичайні математичні дії над операндами дають результатом точне число. Подання такого результату у вигляді числа з рухомою комою не завжди буде точним. Десяткові числа з рухомою комою на відміну від двійкових дозволяють у більшості випадків отримати точний результат, оскільки основа їх подання збігається зі звичною для нас десятковою системою числення.

5. Арифметика десяткових чисел з рухомою комою передбачає не тільки отримання максимально точного результату обчислення, але в окремих випадках і, максимальної кількості значущих цифр. Однак у ній також повністю відсутній аналіз похибок обчислення.

Фактично основні недоліки двійкових чисел з рухомою комою в тій чи іншій мірі притаманні також і десятковим числам з рухомою комою за винятком, наприклад, недоліків, пов'язаних з поданням скінчених дробів та похибок округлення (див. третій та четвертий недоліки, викладені в п. 4.1).

Таким чином, десяткові числа з рухомою комою не вирішують проблем щодо отримання точного та достовірного результату. Їхні переваги перед двійковими числами з рухомою комою більшою чи меншою мірою компенсуються їхніми недоліками, а це означає, що проблема пошуку інших способів подання чисел у комп'ютері, які б забезпечили точні та достовірні обчислення, залишається відкритою.

Парадоксальність ситуації полягає в тому, що комп'ютер, як один з найбільш точних пристроїв, створених людиною, в той же час є абсолютно неточним, оскільки використовує наближені обчислення.

Розділ 5. ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

5.1 Загальні вимоги до виконання лабораторних робіт

Виконання лабораторної роботи передбачає:

1) вивчення теоретичного матеріалу;
2) виконання завдань до лабораторної роботи;
3) написання звіту;
4) підготовку відповідей на контрольні запитання;
5) підготовку до здавання теоретичного тестового опитування.

1) Студент повинен опрацювати теоретичний матеріал, необхідний для виконання лабораторної роботи. Пункти теоретичного матеріалу можуть бути вибрані в довільному порядку, хоча у більшості випадків вони збігаються з порядком викладу.

2) Для кожного студента передбачено індивідуальне завдання. Номер варіанту відповідає порядковому номеру студента в **загальному списку** групи.

3) Звіт повинен містити інформацію про студента (група, Ф.І.Б., № лабораторної роботи, № завдання), індивідуальне завдання до лабораторної роботи (повністю) та його вирішення (висновок). В окремих випадках звіт може містити додаткові висновки до лабораторної роботи.

Звіт необхідно оформити **від руки** на чистому аркуші формату А4. Всі звіти вкладають в єдиний файл, який студент в кінці семестру здає викладачу. Допускається використовувати звичайний учнівський зошит у клітинку для запису звітів усіх виконаних лабораторних робіт.

4) Відповіді на контрольні запитання повинні виявити як глибоке теоретичне знання матеріалу, так і вміння його застосувати на практиці.

5) Тестові завдання переважно теоретичного характеру, однак можуть містити практичні завдання з кодування чисел та здійснення над ними певних арифметичних дій. Кожне тестове завдання має чотири варіанти відповіді, з яких вірним є тільки одне. Для отримання мінімальної позитивної оцінки необхідно відповісти щонайменше на половину тестових запитань.

Сумарна оцінка студента формується з оцінки виконання практичного завдання, відповідей на запитання до лабораторної роботи та результатів тестування. Невиконання практичного завдання або повне незнання теоретичного матеріалу або незадовільна оцінка з тесту означає сумарну незадовільну оцінку та не зарахування лабораторної роботи.

5.2. Лабораторна робота № 1

Тема роботи. Унарна, двійкова, вісімкова та шістнадцяткова системи числення.

Мета роботи. Вивчити способи подання числа в двійковій, вісімковій та шістнадцятковій системах числення, методи переведення числа із однієї системи в іншу, а також правила виконання основних арифметичних дій над такими числами.

Теоретичні відомості. Розділ 1 повністю**.**

Завдання до лабораторної роботи:

Завдання 1. Перевести з десяткової в двійкову, вісімкову, та шістнадцяткову системи числення десяткове число, яке формується в такий спосіб:

$$\mathbf{A} = (\text{Дата народження}) \times 10^6 + (\text{місяць народження}) \times 10^4 + (\text{рік народження}) + \text{порядковий номер у списку} / 25.$$

Наприклад, дата народження 20.04.1994, порядковий номер у списку 18. Число **A** = 20041994 + 18/25 = 20041994,72. Результат подати з 10-ма двійковими знаками після коми у двійковій системі числення та відповідною кількістю знаків після коми в інших системах числення.

Завдання 2. Над числами **X** та **Y** виконати операції **X+Y**, **X-Y**, **X•Y** і **X/Y**.

Завдання 3. Число **Z** перевести в двійкову та вісімкову системи числення.

Варіант / Число	1	2	3	4	5
X	11111111b	11100001b	11010010b	11000011b	10110100b
Y	00010001b	00001111b	00001110b	00001101b	00001100b
Z	4AB7h	2C78h	1010h	ABCDh	ABBAh
Варіант / Число	6	7	8	9	10
X	11111100b	11101110b	10110110b	10101000b	10011010b
Y	00010010b	00010001b	00001101b	00001100b	00001011b
Z	55FFh	FC9Ch	90F1h	3727h	4C82h
Варіант / Число	11	12	13	14	15
X	11110111b	11101010b	11011101b	10011100b	10001101b
Y	00010011b	00010010b	00010001b	00001100b	00001011b
Z	F00Fh	1111h	2112h	DA56h	7F20h

Варіант / Число	16	17	18	19	20
X	11111100b	11110000b	11100100b	11011000b	11001100b
Y	00010101b	00010100b	00010011b	00010010b	00010001b
Z	478Fh	CDAFh	D76Ah	49DAh	AF91h
Варіант / Число	21	22	23	24	25
X	11111101b	11110010b	11100111b	11011100b	11010001b
Y	00010111b	00010110b	00010101b	00010100b	00010011b
Z	5DABh	BAD7h	7777h	AB93h	BCDAh

Контрольні запитання.

1. Яку систему числення називають унарною? інверсною унарною? Які цифри належать алфавіту цих систем числення?
2. Де у обчислювальних машинах використовують унарну та інверсну унарну системи числення.
3. До якого виду систем числення належать унарна та інверсна унарна системи – позиційних чи непозиційних?
4. Що називають системою числення, цифрами, алфавітом системи?
5. Які системи числення називають позиційними, а які непозиційними? Навести приклади позиційних та непозиційних систем.
6. Яким чином нумерують розряди числа? Дати означення розряду, ваги, основи системи числення.
7. Записати значення числа за допомогою цифр та вагових коефіцієнтів.
8. Які системи числення є найбільш зручними з точки зору завадостійкості для використання їх в ЕОМ?
9. Чому трійкову систему числення не використовують в сучасних ЕОМ?
10. Для чого в сучасних ЕОМ використовують двійкову, вісімкову та шістнадцяткову системи числення? Яка із них лежить в основі роботи ЕОМ?
11. Яким чином здійснюють додавання та віднімання чисел у позиційних системах числення? (на прикладі двійкової та вісімкової систем).
12. Яким чином здійснюють множення та ділення чисел у позиційних системах числення? (на прикладі двійкової та вісімкової системи).
13. В яких випадках переходу від однієї системи до іншої використовують метод безпосереднього заміщення? В чому він полягає?
14. В чому полягає метод послідовного ділення (множення)?
15. Чому десятковий скінченний дріб, поданий у двійковій системі числення не завжди буває скінченним?
16. За допомогою якого простого методу здійснюють перехід з двійкової системи числення у вісімкову (шістнадцяткову) та навпаки?
17. Яким чином здійснити перехід від вісімкової системи числення до шістнадцяткової чи навпаки?

5.3. Лабораторна робота № 2

Тема роботи. Прямий, інверсний та доповняльний коди двійкових чисел.

Мета роботи. Вивчити способи подання додатних та від'ємних чисел за допомогою прямого, інверсного та доповняльного кодів, а також правила виконання арифметичних операцій віднімання та додавання в цих кодах.

Теоретичні відомості. Розділ 2, пп. 2.1 – 2.5, п. 2.10.

Завдання до лабораторної роботи:

Завдання 1. Числа **X** і **Y** записати в прямому, інверсному та доповняльному кодах використовуючи один знаковий, 11 цілих та 4 дробових розрядів (усього 16 розрядів).

Завдання 2. Над числами **X** і **Y** в інверсному та доповняльному кодах виконати операції **X**+**Y**, **X**-**Y** і **Y**-**X**. Результат перевести в десяткову систему числення. (Операцію віднімання необхідно реалізувати шляхом додавання числа з протилежним знаком).

Варіант / Число	1	2	3	4	5
X	-100,05d	-200,55d	-300,95d	-400,13d	-500,3d
Y	1320,1d	1799,2d	1570,3d	1287,4d	987,5d
Варіант / Число	6	7	8	9	10
X	600,33d	700,99d	800,58d	900,77d	1000,56d
Y	-1276,6d	-483,7d	-678,8d	-977,9d	-478,1d
Варіант / Число	11	12	13	14	15
X	-150,11d	-250,35d	-350,67d	-450,47d	-550,89d
Y	1555,2d	1466,3d	1200,4d	893,5d	1030,6d
Варіант / Число	16	17	18	19	20
X	650,33d	750,25d	850,92d	950,85d	1050,78d
Y	-787,7d	335,8d	-555,9d	-357,1d	-567,2d
Варіант / Число	21	22	23	24	25
X	-111,66d	-222,25d	-333,85d	-444,75d	-555,61d
Y	1333,3d	1222,4d	1111,5d	1453,6d	1256,7d

Контрольні запитання.

1. З якою метою використовують прямий, інверсний та доповняльний коди чисел? Яким чином відрізняють додатні числа від від'ємних?
2. В яких випадках застосовують прямий код чисел? Його переваги та недоліки.
3. Яка суттєва відмінність між числами в прямому та інверсному (чи доповняльному) кодах?
4. Яка величина вагового коефіцієнту знакового розряду в інверсному та доповняльному кодах?
5. Яким чином число, записане в прямому коді перевести в інверсний код?
6. Який діапазон чисел можна подати за допомогою інверсного коду? Скільки існує подань числа нуль?
7. Яким чином здійснюють додавання чисел, записаних в інверсному коді? У чому зміст операції циклічного перенесення?
8. Яким чином реалізують операцію віднімання за допомогою інверсного коду? Переваги та недоліки інверсного коду.
9. З якою метою застосовують доповняльний код чисел? Переваги і недоліки доповняльного коду.
10. Яким чином здійснюють переведення чисел з прямого в доповняльний, та з доповняльного в прямий коди?
11. Які правила додавання чисел, записаних у доповняльному коді?
12. Який діапазон чисел можна подати за допомогою доповняльного коду? Скільки існує подань числа нуль?
13. Які із способів подання двійкових чисел використовують в обчислювальних машинах?
14. Чи допускається запис чисел у інверсному та доповняльному кодах за допомогою шістнадцяткової системи числення?

5.4. Лабораторна робота № 3

Тема роботи. Зміщений код двійкових чисел. Переповнення розрядної сітки.

Мета роботи. Вивчити способи подання додатних і від'ємних чисел за допомогою зміщених кодів, а також правила виконання арифметичних операцій віднімання та додавання в цих кодах. Розглянути, за яких умов можливе переповнення розрядної сітки та основні способи його виявлення.

Теоретичні відомості. Розділ 2, пп. 2.6 – 2.10.

Завдання до лабораторної роботи:

Завдання 1. Числа **X** і **Y** записати в зміщених (з додатнім та від'ємним нулем) кодах використовуючи один знаковий, 11 цілих та 4 дробових розрядів (усього 16 розрядів).

Завдання 2. Над числами **X** і **Y** у зміщених (з додатнім та від'ємним нулем) кодах виконати операції **X**+**Y**, **X**-**Y** і **Y**-**X**. Результат перевести в десяткову систему числення.

Завдання 3. Записати числа **A** і **B** у модифікованому інверсному та модифікованому доповняльному кодах використовуючи 8 числових та два знакові розряди, а також здійснити над ними операції **A**+**B** і **A**-**B**. Перевірити коректність отриманих результатів.

Примітка: Операцію віднімання в завданнях 2, 3 необхідно реалізувати шляхом додавання числа з протилежним знаком.

Варіант / Число	1	2	3	4	5
X	-100,05d	-200,55d	-300,95d	-400,13d	-500,3d
Y	1320,1d	1799,2d	1570,3d	1287,4d	987,5d
A	-200d	230d	150d	-220d	-53d
B	100d	-135d	-170d	67d	218d
Варіант / Число	6	7	8	9	10
X	600,33d	700,99d	800,58d	900,77d	1000,56d
Y	-1276,6d	-483,7d	-678,8d	-977,9d	-478,1d
A	-160d	44d	-89d	77d	-80d
B	170d	-244d	180d	-197d	180d
Варіант / Число	11	12	13	14	15
X	-150,11d	-250,35d	-350,67d	-450,47d	-550,89d
Y	1555,2d	1466,3d	1200,4d	893,5d	1030,6d
A	-65d	165d	-185d	-68d	34d
B	210d	-180d	125d	204d	-234d

Варіант / Число	16	17	18	19	20
X	650,33d	750,25d	850,92d	950,85d	1050,78d
Y	-787,7d	335,8d	-555,9d	-357,1d	-567,2d
A	-145d	-57d	73d	111d	-58d
B	130d	235d	-211d	-222d	215d
Варіант / Число	21	22	23	24	25
X	-111,66d	-222,25d	-333,85d	-444,75d	-555,61d
Y	1333,3d	1222,4d	1111,5d	1453,6d	1256,7d
A	33d	-187d	-86d	199d	-156d
B	-233d	178d	199d	-209d	165d

Контрольні запитання.

1. У чому полягає основний зміст зміщеного коду чисел? В яких випадках використовують зміщений код чисел?
2. Як подають додатні та від'ємні числа в зміщеному коді?
3. У чому зміст "від'ємного нуля" і чим зміщений код з від'ємним нулем відрізняється від коду з додатнім нулем? Яка величина зміщення в таких кодах?
4. Яким чином число, записане в прямому коді, перевести в зміщений код з додатнім нулем? з від'ємним нулем?
5. Який діапазон чисел можна подати за допомогою зміщеного коду? Скільки існує подань числа нуль?
6. Чим відрізняються числа, записані в доповняльному та зміщеному кодах з додатнім нулем? У чому полягає недолік такого додавання?
7. Які правила додавання чисел, записаних у зміщеному коді з додатнім нулем? з від'ємним нулем?
8. У чому полягає найбільш суттєва відмінність чисел, поданих у прямому, інверсному чи доповняльному кодах від чисел, які подані в зміщеному коді?
9. Які переваги і недоліки чисел, записаних у зміщеному коді?
10. В чому сутність переповнення розрядної сітки? Яким чином його виявити?
11. Які коди називають модифікованими? Недоліки і переваги модифікованих кодів. Чи існує модифікований прямий код чисел? Чому?
12. Які правила додавання чисел у модифікованому інверсному та модифікованому доповняльному кодах?
13. Які із способів подання двійкових чисел (кодів) використовують в обчислювальних машинах?
14. Чи існує зміщений код у вісімковій системі числення? у шістнадцятковій системі числення?

5.5. Лабораторна робота № 4

Тема роботи. Двійкові числа з рухомою комою стандарту IEEE754-1985.

Мета роботи. Вивчити способи подання дійсних чисел у комп'ютері. Розглянути базові положення стандарту IEEE754-1985, який описує властивості двійкових чисел з рухомою комою, а також основні формати подання таких чисел. Навчитися здійснювати переведення чисел, записаних у десятковій системі числення у двійкові числа з рухомою комою та навпаки.

Теоретичні відомості. Розділ 3, пп. 3.1 – 3.5, п. 3.6.3.

Завдання до лабораторної роботи:

Завдання 1. Числа **X** і **Y** записати у форматі одинарної та подвійної точності стандарту IEEE754.

Завдання 2. Число **Z** записати в десятковій системі числення, розглядаючи його як ціле число в доповняльному коді.

Завдання 3 Число **Z** записати в десятковій системі числення, розглядаючи його як число одинарної точності стандарту IEEE754.

Варіант / Число	1	2	3	4	5
X	-100,05d	-200,55d	-300,95d	-400,13d	-500,3d
Y	1320,1d	1799,2d	1570,3d	1287,4d	987,5d
Z	75F12500h	FCF22400h	7FC32300h	FAE42200h	7FE52100h
Варіант / Число	6	7	8	9	10
X	600,33d	700,99d	800,58d	900,77d	1000,56d
Y	-1276,6d	-483,7d	-678,8d	-977,9d	-478,1d
Z	77FF2100h	FF193600h	8F081800h	78FF1700h	FF0F1600h
Варіант / Число	11	12	13	14	15
X	-150,11d	-250,35d	-350,67d	-450,47d	-550,89d
Y	1555,2d	1466,3d	1200,4d	893,5d	1030,6d
Z	F7001500h	77121400h	F9131300h	F1F41200h	79F51100h
Варіант / Число	16	17	18	19	20
X	650,33d	750,25d	850,92d	950,85d	1050,78d
Y	-787,7d	335,8d	-555,9d	-357,1d	-567,2d
Z	FE160900h	FD170900h	7CF82300h	7A792400h	F0F02500h

Варіант / Число	21	22	23	24	25
X	-111,66d	-222,25d	-333,85d	-444,75d	-555,61d
Y	1333,3d	1222,4d	1111,5d	1453,6d	1256,7d
Z	72F11400h	73F21500h	74F51600h	75F61700h	77F81900h

Контрольні запитання.

1. Яким чином у комп'ютері подають цілі числа. Назвіть загальновживані формати цілих чисел?
2. Описати спосіб подання дійсних чисел з фіксованою комою? Які його переваги та недоліки.
3. У чому сутність методу масштабованих коефіцієнтів? Його переваги та недоліки.
3. Дати означення абсолютної та відносної похибок подання. Що називають машинним нулем?
4. Запис дійсного числа в експоненціальній формі у десятковій та двійковій системі числення. Математичний зміст мантиси та експоненти.
5. Нормальна та нормалізована форми дійсного числа.
6. Дати визначення двійкового числа з рухомою комою. Описати загальний формат чисел з рухомою комою.
7. Двійкове число з рухомою комою. Пояснити терміни: зміщення, зміщена експонента, залишок мантиси, неявна одиниця.
8. Двійкове число з рухомою комою. Мінімальне та максимальне число (за модулем) подання.
9. Абсолютна та відносна похибки двійкових чисел з рухомою комою.
10. З якою метою розроблено стандарт IEEE754-1985? Які основні формати чисел передбачені цим стандартом?
11. Нормалізовані та денормалізовані числа стандарту IEEE754-1985. Діапазон подання нормалізованих та денормалізованих чисел одинарної та подвійної точності.
12. "Виняткові числа": нуль, нескінченність, NaN.
13. Правила подання чисел у стандарті IEEE754-1985.
14. Правила переведення чисел з стандарту IEEE754-1985 у десяткову систему числення.
15. Способи округлення чисел згідно з стандартом IEEE754-1985.

5.6. Лабораторна робота № 5

Тема роботи. Додавання та віднімання двійкових чисел з рухомою комою.

Мета роботи. Вивчити подання дійсних чисел у вигляді двійкових чисел з рухомою комою та алгоритм здійснення операцій додавання (віднімання) таких чисел. Навчитися здійснювати арифметичні дії додавання та віднімання двійкових чисел з рухомою комою без переведення їх у десяткову систему числення.

Теоретичні відомості. Розділ 3, пп. 3.3 – 3.7, п. 4.6.

Завдання до лабораторної роботи:

Завдання 1. Додати числа **X** і **Y** без переведення їх у десяткову систему числення.

Завдання 2. Від числа **X** відняти число **Z** без переведення їх у десяткову систему числення.

Завдання 2. Перевірити правильність виконання операцій у завданні 1 шляхом переведення значення операндів і результату в десяткову систему числення.

Варіант / Число	1	2	3	4	5
X	C49C4000h	4512C000h	C58DA000h	454E1000h	C4A52000h
Y	41300000h	41B00000h	42040000h	42300000h	425C0000h
Z	C49A2000h	4513C000h	C58D6000h	454C1000h	C4AA0000h
Варіант / Число	6	7	8	9	10
X	44E4A000h	C54D7000h	46120000h	C5EB6800h	45758000h
Y	C1C80000h	C1700000h	C1A00000h	C2840000h	C20C0000h
Z	44E10000h	C54E4000h	4611A000h	C5EC4000h	45746000h
Варіант / Число	11	12	13	14	15
X	C5184000h	45D6A000h	C5AD9800h	45510000h	C60C9800h
Y	42200000h	42480000h	42700000h	42140000h	41400000h
Z	C519C000h	45D61000h	C5ACA800h	45505000h	C60C4400h
Варіант / Число	16	17	18	19	20
X	461C3C00h	C54E4000h	447A0000h	C54DA000h	4604A000h
Y	C2B00000h	C2040000h	C1200000h	C1900000h	C2080000h
Z	461C1000h	C54F8000h	449CA000h	C54B2000h	4604D000h

Варіант / Число	21	22	23	24	25
X	CC5B97000h	45D3B800h	C5F30800h	460E3000h	C5610000h
Y	41900000h	42040000h	41C80000h	42300000h	42040000h
Z	C5BB8000h	45D22800h	C5F36000h	460E9400h	C55F7000h

Контрольні запитання.

1. Нормальна та нормалізована форми дійсного числа.
2. Дати визначення двійкового числа з рухомою комою. Описати загальний формат чисел з рухомою комою.
3. Двійкове число з рухомою комою. Пояснити терміни : зміщення, зміщена експонента, залишок мантиси, неявна одиниця.
4. З якою метою розроблено стандарт IEEE754-1985? Які основні формати чисел передбачені цим стандартом?
5. "Виняткові числа": нуль, нескінченність, NaN.
6. Назвіть основні етапи здійснення операції додавання (віднімання) двійкових чисел з рухомою комою.
7. Пояснити зміст перевірки на специфічні значення операндів.
8. Які специфічні (виняткові) значення операндів операції додавання чи віднімання?
9. У чому зміст вирівнювання порядків? Для чого застосовують зсув мантис?
10. Яким чином здійснюють зсув мантис? У чому зміст втрати значимості мантиси?
11. Чи є помилка втрати значимості мантиси сигналізуючою, тобто чи призводить вона до появи результату sNaN?
12. У чому полягає відмінність qNaN від sNaN? В яких випадках вони появляються?
13. Яким чином формується знак, зміщена експонента та мантиса суми (різниці)?
14. У чому полягає операція нормалізації? В яких випадках під час додавання (віднімання) її слід здійснювати?
15. У чому зміст помилки недоповнення чи переповнення порядку? Яким чином така помилка відображається на результаті виконання операції додавання (віднімання)?
16. Яким чином здійснюють округлення результату виконання операцій. Які чотири способи передбачено стандартом IEEE754-1985?
17. У чому зміст округлення до парного? В яких випадках його застосовують?
18. Яку похибку називають зміщеною похибкою округлення? В яких випадках вона появляється?
19. У чому зміст округлення до $+\infty$? до $-\infty$? Яким чином здійснюють округлення до нуля?

5.7. Лабораторна робота № 6

Тема роботи. Множення та ділення двійкових чисел з рухомою комою.

Мета роботи. Вивчити подання дійсних чисел у вигляді двійкових чисел з рухомою комою та алгоритм здійснення операцій множення і ділення таких чисел. Навчитися здійснювати арифметичні дії множення і ділення двійкових чисел з рухомою комою без переведення їх у десяткову систему числення.

Теоретичні відомості. Розділ 3, пп. 3.3 – 3.7, п. 4.6.

Завдання до лабораторної роботи:

Завдання 1. Над числами **X** і **Y** здійснити операції **X × Y** та **X / Y** без переведення їх у десяткову систему числення.

Завдання 2. Перевірити правильність виконання операцій у завданні 1 шляхом переведення значення операндів і результату в десяткову систему числення.

Варіант / Число	1	2	3	4	5
X	C9064750h	49212250h	C8A122A0h	4856D940h	C8AAE6A0h
Y	C2200000h	C2400000h	C2600000h	C2900000h	C2B00000h
Варіант / Число	6	7	8	9	10
X	48D6D8Ah0	C8A604A0h	48742540h	C884AAA0h	49016550h
Y	42C00000h	42D00000h	42E00000h	42F00000h	43200000h
Варіант / Число	11	12	13	14	15
X	C9323950h	4945C150h	C86A6140h	48BE6EA0h	C9371B50h
Y	C3080000h	C3100000h	C3180000h	C2A00000h	C2A80000h
Варіант / Число	16	17	18	19	20
X	48AFC8A0h	C8EF42A0h	4908B850h	C8B98CA0h	48DBBAA0h
Y	43300000h	42B80000h	42C80000h	43480000h	43580000h
Варіант / Число	21	22	23	24	25
X	C919CF50h	49260450h	C9323950h	48D6D8A0h	C87DE940h
Y	C3680000h	C3700000h	C3280000h	C2F00000h	C3500000h

Контрольні запитання.

1. Нормальна та нормалізована форми дійсного числа.
2. Дати визначення двійкового числа з рухомою комою. Описати загальний формат чисел з рухомою комою.
3. Двійкове число з рухомою комою. Зміщення, зміщена експонента, залишок мантиси, неявна одиниця.
4. З якою метою розроблено стандарт IEEE754-1985? Які основні формати чисел передбачені цим стандартом?
5. "Виняткові числа": нуль, нескінченність, NaN.
6. Назвіть основні етапи здійснення операції множення та ділення двійкових чисел з рухомою комою.
7. У чому полягає основна різниця алгоритмів множення та ділення?
8. Пояснити зміст перевірки на специфічні значення операндів.
9. Які специфічні (виняткові) значення операндів операції множення? ділення? Відповідь обґрунтуйте.
10. Яким чином формують знак, зміщену експоненту та мантису добутку? частки?
11. Яким чином здійснюють додавання (віднімання) експонент операндів? Чому для цього використовують зміщений код?
12. У чому полягає відмінність qNaN від sNaN? В яких випадках вони появляються?
13. У чому полягає операція нормалізації? В яких випадках під час множення чи ділення її слід здійснювати?
14. У чому зміст помилки недоповнення чи переповнення порядку? Яким чином така помилка відображається на результаті виконання операції додавання (віднімання)?
15. Яким чином помилки недоповнення чи переповнення порядку зробити несигналізуючими, тобто такими, що повертають число, а не sNaN?
16. Яким чином здійснюють округлення результату виконання операцій. Які чотири способи передбачено стандартом IEEE754-1985?
17. У чому зміст округлення до парного? В яких випадках його застосовують?
18. Яку похибку називають зміщеною похибкою округлення? В яких випадках вона появляється?
19. У чому зміст округлення до $+\infty$? до $-\infty$? Яким чином здійснюють округлення до нуля?

5.8. Лабораторна робота № 7

Тема роботи. Десяткові числа з рухомою комою стандарту IEEE754-2008.

Мета роботи. Розглянути основні недоліки бінарних чисел з рухомою комою та шляхи їх подолання. Вивчити спосіб подання десяткових чисел з рухомою комою в комп'ютері. Навчитися здійснювати переведення чисел, записаних у десятковій системі числення у десяткові числа з рухомою комою та навпаки.

Теоретичні відомості. Розділ 4, пп. 4.1 – 4.4, п. 4.7 – 4.10.

Завдання до лабораторної роботи:

Завдання 1. Числа **X** та **Y** записати у форматі десяткових чисел з рухомою комою одинарної точності стандарту IEEE-754-2008.

Завдання 2. Десяткове число з рухомою комою **Z,** записане у форматі чисел одинарної точності стандарту IEEE-754-2008 подати у десятковій системі числення.

Варіант / Число	1	2	3	4	5
X	1,3201d	0,17992d	15,703d	12,874d	987,589d
Y	-100,05d	-200,55d	-300,95d	-400,13d	-500,3d
Z	C9064750h	49212250h	C8A122A0h	4856D940h	C8AAE6A0h
Варіант / Число	6	7	8	9	10
X	-12776,6d	-483,777d	-678,8757d	-97777,9d	-477781d
Y	600,33d	700,99d	800,58d	900,77d	1000,56d
Z	48AFC8A0h	C8EF42A0h	4908B850h	C8B98CA0h	48DBBAA0h
Варіант / Число	11	12	13	14	15
X	1555,2d	1466,3d	1200,4d	5668935d	1030,6345d
Y	-150,11d	-250,35d	-350,67d	-450,47d	-550,89d
Z	48D6D8Ah0	C8A604A0h	48742540h	C884AAA0h	49016550h
Варіант / Число	16	17	18	19	20
X	-768787,7d	33885,8d	-555555,9d	-666357,1d	-567,2d
Y	650,33d	750,25d	850,92d	950,85d	1050,78d
Z	C919CF50h	49260450h	C9323950h	48D6D8A0h	C87DE940h
Варіант / Число	21	22	23	24	25
X	19333,3d	1222,499d	188111,5d	1453,6d	1299956,7d
Y	-111,66d	-222,25d	-333,85d	-444,75d	-555,61d
Z	C9323950h	4945C150h	C86A6140h	48BE6EA0h	C9371B50h

Контрольні запитання.

1. Які основні недоліки стандарту IEEE754-1985? У чому вони полягають?
2. До яких наслідків може призводити типова операція "округлення до парного"?
3. У чому недолік подвійного подання нуля (+0, -0)?
4. Які основні характеристики четверного (128-бітного) формату?
5. Які переваги та недоліки нарощування кількості розрядів подання?
6. У чому сутність прикладу Румпа? На які небезпечні чинники процесу обчислення він вказує?
7. Яким чином отримати правильний розвязок прикладу Румпа?
8. У чому найсуттєвіша відмінність стандарту IEEE754-2008 від стандарту IEEE754-1985? Який новий тип чисел з рухомою комою описано у стандарті?
9. Описати базові бінарні формати стандарту IEEE Std 754-2008.
10. Множинні бінарні формати стандарту IEEE Std 754-2008. Якої довжини формат не передбачено?
11. Основні принципи подання десяткових чисел з рухомою комою. Яка основна відмінність між мантисами десяткових та двійкових чисел з рухомою комою?
12. Двійково-десяткове кодування. Його переваги та недоліки.
13. З якою метою використовують щільно упаковані десяткові числа? Яким чином над такими числами здійснюють арифметичні дії?
14. Надлишковість щільно упакованих чисел та ефективність використання кодових комбінацій порівняно з двійково-десятковим кодуванням.
15. Яким чином здійснюють подання дійсних чисел у вигляді десяткових чисел з рухомою комою стандарту IEEE754-2008?
16. Скільки полів кодування містить стандартний формат десяткових чисел з рухомою комою стандарту IEEE754-2008?
17. Яку, нормальну чи нормалізовану мантису, використовують для запису десяткових чисел з рухомою комою?
18. Яка типова величина зміщення експоненти десяткових чисел з рухомою комою та з чим це пов'язано?
19. Для чого і яким чином використовують комбіноване поле та які дані воно подає?
20. Яким чином здійснюють кодування нескінченності та не-чисел формату десяткових чисел з рухомою комою стандарту IEEE754-2008?

5.9. Лабораторна робота № 8

Тема роботи. Арифметичні дії над десятковими числами з рухомою комою.

Мета роботи. Розглянути базові положення стандарту IEEE754-2008, який описує властивості як двійкових, так і десяткових чисел з рухомою комою, а також основні формати подання таких чисел. Навчитися здійснювати арифметичні операції над цими числами.

Теоретичні відомості. Розділ 4, пп. 4.4 – 4.16.

Завдання до лабораторної роботи:

Завдання 1. Над десятковими числами з рухомою комою формату 32-біти **X** і **Y** здійснити операції **X + Y**, **X × Y** та **X / Y**.

Завдання 2. Для числа **Z** записати всі члени когорти подання у форматі 32-біти десяткових чисел з рухомою комою.

Варіант / Число	1	2	3	4	5
X	A1B00EAFh	A5C01DBDh	A9D0481Fh	ADE0625Dh	B1F09FFAh
Y	2DE113A3h	31F1575Fh	36018F39h	3A11A69Bh	3E21DED9h
Z	+0,234	-0,0456	+0,000209	-1,71	+2300
Варіант / Число	6	7	8	9	10
X	B600A7BCh	BA108E3Ch	BE20591Ch	EE30259Ch	A24017DDh
Y	6A31E6E7h	6E41D2B2h	22519F27h	26616778h	2A7123CFh
Z	+190,10	-0,045000	+0,0769	-90,080	-80,00
Варіант / Число	11	12	13	14	15
X	A65ABC03h	AA66F407h	AE707C12h	B2897418h	B69FE827h
Y	2E8E8E44h	329D7C55h	36ACE463h	3ABA6C69h	3ECB6477h
Z	+3,65	-0,008870	+90,000	-9,800	+2,090
Варіант / Число	16	17	18	19	20
X	BAAEF029h	BEB8F023h	EAC43016h	EED67009h	A2EF7405h
Y	6ADB9C79h	6EEAC874h	22FC9C67h	268DE059h	2A9F3C48h
Z	+50,093	-2000,1	+47,80	-45,200	-567
Варіант / Число	21	22	23	24	25
X	A6F4244Bh	AAE9201Bh	AED7685Fh	B2C324F5h	B6B34CA3h
Y	2EA42567h	32B7D5A0h	36A7885Ah	3A99CC4Fh	3E89382Bh
Z	+0,00300	-0,000047	+6,700	-450,0000	+0,056

Контрольні запитання.

1. У чому найсуттєвіша відмінність стандарту IEEE754-2008 від стандарту IEEE754-1985? Який новий тип чисел з рухомою комою описано в стандарті?
2. Основні принципи подання десяткових чисел з рухомою комою. Яка основна відмінність між мантисами десяткових та двійкових чисел з рухомою комою?
3. Який діапазон чисел і з якою похибкою можна подати за допомогою десяткових чисел з рухомою комою? Від чого він залежить?
4. Які основні (базові) формати десяткових чисел з рухомою комою стандарту IEEE754-2008?
5. У чому переваги та недоліки десяткових чисел з рухомою комою перед бінарними?
6. За допомогою якої системи числення десяткові числа з рухомою комою подають у пам'яті комп'ютера?
7. Дати визначення когорти десяткових чисел з рухомою комою. З чим пов'язана така особливість цих чисел?
8. Яку кількість членів може містити когорта скінченого ненульового числа? числа нуль? нескінченності?
9. Чим відрізняються між собою члени когорти?
10. Яким чином серед членів когорти результату обчислення вибирають той, який буде подавати результат?
11. Яким чином вибирають оптимальну експоненту?
12. В чому полягає операція вирівнювання мантиси?
13. Основні відмінності виконання арифметичних дій над десятковими числами з рухомою комою в стандарті IEEE754-2008 від таких дій над двійковими числами з рухомою комою.
14. Який спосіб округлення спеціально для десяткових чисел з рухомою комою введено стандартом IEEE754-2008? У чому він полягає?
15. Який спосіб округлення за замовчуванням передбачає стандарт IEEE754-2008 для двійкових чисел з рухомою комою? для десяткових чисел з рухомою комою?
16. Два види не-чисел: qNaN та sNaN. Яким чином подають ці числа згідно з IEEE754-2008 у двійкових та десяткових числах з рухомою комою?
17. Результатом яких операцій може бути NaN?
18. В яких випадках використовують qNaN, а в яких – sNaN?
19. Яким чином можна використовувати qNaN та sNaN?
20. У чому полягає особливість порівняння чисел з NaN?

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Римська система числення. Вікіпедія, вільна енциклопедія [інтернет ресурс] – режим доступу: https://uk.wikipedia.org/wiki/Римська_система_числення
2. Старослов'янська система числення. Вікіпедія, вільна енциклопедія [інтернет ресурс] – режим доступу: https://uk.wikipedia.org/wiki/Старослов'янська_система_числення
3. Унарна система числення. Вікіпедія, вільна енциклопедія [інтернет ресурс] – режим доступу: https://uk.wikipedia.org/wiki/Унарна_система_числення
4. Бузунов Ю. А. Принципы построения цифровых вычислительных машин. / Ю. А. Бузунов., Е. Н. Вавилов. – К.: Техніка, 1972. – 316 с.
5. Каган Б. М. Электронные вычислительные машины и системы : учеб. пособие для вузов. / Б. М.Каган – М.: Энергоатомиздат, 1991. – 592 с.
6. Никамин В. А. Аналогово-цифровые и цифро-аналоговые преобразователи : справочник. / В. А. Никамин – СПб.: КОРОНА принт; М.: Альтекс-А, 2003. – 224 с.
7. IEEE Std 754-1985. IEEE Standard for Binary Floating-Point Arithmetic. Approved March 21, 1985. – New York.: IEEE, 1985. – 20 с.
8. 128 біт. . Вікіпедія, вільна енциклопедія [інтернет ресурс] – режим доступу: https://uk.wikipedia.org/wiki/128_біт
9. IEEE Std 754TM – 2008. IEEE Standard for Floating-Point Arithmetic. Approved August 29, 2008. New York. : IEEE, 2008. – 58 с.
10. Юровицкий В.М. IEEE754-тика угрожает человечеству., МФТИ, РГСУ, Москва, [інтернет ресурс] – режим доступу: www.yur..ru
11. Аноприенко А. Я. Пример Румпа в контексте традиционных, интервальных и постбинарных вычислений. / А.Я. Аноприенко, В.А. Гранковский, Иваница. // Наук. праці ДонНТУ Сер. : Проблеми моделювання та автоматизації проектування, 2001. – 9 (179).
12. Rump S. M. Algorithm for verified inclusions: Theory and practice Reliability in Computing. / R. E. Moore ed., San Diego. : Academic press, 1988. – p. 109-126.
13. Rump S. M. Rigorous results using floating-point arithmetic. / Rump S. M. // Acta Numerica. – 2010. – v.19. – p. 287–449.
14. Cowlishaw M. Decimal Arithmetic Encoding Strawman 4d. IBM UK Laboratories, 2003, [інтернет ресурс] – режим доступу: mfc@uk.ibm.com
15. Tien Chi Chen (March 29, 1971). Decimal Number Compression. Internal IBM memo to Dr. Irving T. Ho, 4pp, IBM.
16. Cowlishaw M. F. (May 2002). "Densely packed decimal encoding". IEEE Proceedings – Computers and Digital Techniques (Institution of Electrical Engineers) 149 (3): 102–104. doi:10.1049/ip-cdt:20020407. ISSN 1350-2387.
17. Cowlishaw M. F. (2000-10-03). "Summary of Densely Packed Decimal encoding". Retrieved 2008-09-10.
18. POWER6. Вікіпедія, вільна енциклопедія [інтернет ресурс] – режим доступу: https://en.wikipedia.org/wiki/POWER6
19. Помилка Аріан 5. Вікіпедія, вільна енциклопедія [інтернет ресурс] – режим доступу: https://uk.wikipedia.org/wiki/Помилка_Аріан_5

20. Patriot Missile Defense. Software Problem Led to System Failure at Dhahran, Saudi Arabia. Report to the Chairman, Subcommittee on Investigations and Oversight, Committee on Science, Space, and Technology, House of Representatives. – February 1992.
21. Sleipner A. Вікіпедія, вільна енциклопедія [інтернет ресурс] – режим доступу: https://en.wikipedia.org/wiki/Sleipner_A
22. Arnold D. N. The sinking of the Sleipner A offshore platform. URL: http://www.ima.umn.edu/~arnold/disasters/sleipner.html. Accessed on: June 27, 2008.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Мельник А.О. Архітектура комп'ютера. / А. О. Мельник – Луцьк : Волинська обласна друкарня, 2008. – 470 с.
2. Таненбаум Э. Архитектура компьютера : 5-е изд./ Э. Таненбаум – СПб. : Питер, 2007. – 844 с.
3. Гашков С.Б. Системы счисления и их применение. / С. Б. Гашков – М. : МЦНМО, 2004.- 52 с.
4. Гилмор Ч. Введение в микропроцессорную технику. / Ч. Гилмор – М. : Мир, 1984. – 334 с.
5. Шауман А.М. Основы машинной арифметики. / А. М. Шауман – Л. : Из-во Ленингр. ун-та, 1979. – 312 с.
6. Поспелов Д.А. Арифметические основы вычислительных машин дискретного действия: учеб. пособие для втузов. / Д. А. Поспелов – М. : Высшая школа, 1970. -308 с.

Терлецький Андрій Іванович
Фрик Оксана Богданівна

АРХІТЕКТУРА КОМП'ЮТЕРІВ

Способи подання чисел у комп'ютері

методичні рекомендації до виконання лабораторних робіт
для студентів напряму "Комп'ютерна інженерія"

Підписано до друку 26.02.2018 Формат 60×84/16
Папір офсетний. Друк цифровий.
Гарнітура «Таймс». Умов. друк. арк. 6,51.
Тираж 100. Зам. № 150 від 26.02.2018

Віддруковано: приватний підприємець Голіней О.М.
76008, м. Івано-Франківськ,
Вул. Галицька, 128
Тел.: (0342) 58-04-32