МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МІСЬКОГО ГОСПОДАРСТВА імені О. М. БЕКЕТОВА**

А. Л. Литвинов

ПРАКТИКУМ З АРХІТЕКТУРИ КОМП'ЮТЕРНИХ СИСТЕМ

НАВЧАЛЬНИЙ ПОСІБНИК

Харків
ХНУМГ ім. О. М. Бекетова
2020

УДК 004.2+004.72]:37.091.33-027.22(075.8)
Л64

Автор

Литвинов Анатолій Леонідович, доктор технічних наук, професор кафедри комп'ютерних наук та інформаційних технологій Харківського національного університету міського господарства імені О. М. Бекетова

Рецензенти:

Федорович Олег Євгенович*,* доктор технічних наук, професор, завідувач кафедри комп'ютерних наук та інформаційних технологій Національного аерокосмічного університету імені М. Є. Жуковського «Харківський авіаційний інститут»;

Чумаченко Ігор Володимирович*,* доктор технічних наук, професор, завідувач кафедри управління проєктами у міському господарстві і будівництві ХНУМГ ім. О.М. Бекетова»

Рекомендовано до друку Вченою радою ХНУМГ ім. О. М. Бекетова, протокол № 10 від 14 травня 2020 р.

Литвинов А. Л.

Л64 Практикум з архітектури комп'ютерних систем: навч. посібник / А. Л. Литвинов ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2020. – 68 с.

Практикум містить навчальний матеріал із практичних занять із дисципліни «Архітектура комп'ютерних систем». Він відповідає програмі дисципліни для студентів спеціальностей 122 – Комп'ютерні науки, 126 – Інформаційні системи та технології, 151 – Автоматизація та комп'ютерно-інтегровані технології. Кожне практичне заняття супроводжується відповідним теоретичним матеріалом, який доповнює лекційний матеріал. До кожного практичного заняття наведені варіанти виконання, зразок виконання і питання для самодіагностики. Наприкінці наведено список рекомендованої літератури.

УДК 004.2+004.72]:37.091.33-027.22(075.8)

ЗМІСТ

ВСТУП

Курс «Архітектура комп'ютерних систем» дозволяє вивчити базові теоретичні і схемотехнічні принципи побудови та функціонування комп'ютерних систем та сформувати практичні навички дослідження і проєктування вузлів комп'ютерних систем.

Метою викладання навчальної дисципліни «Архітектура комп'ютерних систем» є засвоєння необхідних знань з основ теорії побудови та функціонування архітектури сучасної комп'ютерної техніки, що виконані на базі інтегральної технології, формування твердих практичних навичок щодо оцінки технічного стану комп'ютерної техніки, аналізу умов функціонування та синтезу систем із заданими характеристиками.

Основними завданнями вивчення дисципліни є підготовка висококваліфікованих спеціалістів, які вміють раціонально використовувати сучасні типи комп'ютерів з метою автоматизованого проєктування систем, а також аналізувати, розраховувати, синтезувати та проєктувати складні системи з використанням комп'ютерного та мікропроцесорного обладнання.

У результаті вивчення навчальної дисципліни студент повинен:

а) **знати:** порядок аналізу та синтезу систем із заданими властивостями; основи комп'ютерної інженерії **(**комп'ютерну схемотехніку, архітектуру комп'ютерів, мікропроцесорні системи); роль та місце комп'ютерної схемотехніки в завданнях проєктування складних систем;

б) **вміти:** оцінювати характеристики елементів та вузлів, виявляти та усувати несправності в елементах та схемах складної техніки; розробляти основні типи цифрових електронних систем, розраховувати їхню роботу, параметри та характеристики.

Практичні заняття відповідають робочій програмі дисципліни «Архітектура комп'ютерних систем». Вони дозволяють освоїти теоретичні принципи побудови і функціонування комп'ютерних систем. Кожне практичне заняття супроводжується відповідним теоретичним матеріалом, який доповнює лекційний матеріал. До кожного практичного заняття наведені варіанти виконання, зразок виконання і питання для самодіагностики. Наприкінці наведено список рекомендованої літератури.

1 ТЕХНОЛОГІЯ ПЕРЕТВОРЕННЯ ЧИСЕЛ З ОДНІЄЇ СИСТЕМИ ЧИСЛЕННЯ В ІНШУ

Мета заняття: освоєння технології перетворення чисел з однієї системи числення в іншу

Теоретична частина

1.1.1 Переклад чисел з однієї позиційної системи числення в іншу

Інформація в пам’яті комп’ютера зберігається в двійковому вигляді. В основі цього лежить те, що основні елементи для зберігання інформації – транзистор, конденсаторна комірка, магнітні комірки (домени) – можуть перебувати тільки в двох станах, що відповідає 0 або 1. Однак для користувачів інформація зазвичай виводиться у десятковій, шістнадцятковій або вісімковій системі. На рисунку 1.1 зображений фрагмент лістингу програми на асемблері, у якому вміст комірок пам’яті відображений в шістнадцятковій системі.

```
 1 0000                 ************************************************************
 2 0000                 *
 3 0000                 *   ПОДПРОГРАММА ЧИСТКИ ОЗУ
 4 0000                 *
 5 0000                 *
 6 0000                 ************************************************************
 7 0000                         ORG  0000       УСТАНОВИТЬ НАЧ. АДРЕС ПРОГРАММЫ
 8 0000 3E FF    CLRAM  MVI  A,0FFH     АККУМУЛЯТОР=FF
 9 0002 21 00 14        LXI  H,1400H    В H,L НАЧАЛЬНЫЙ АДРЕС ОЗУ
10 0005 77       CL1    MOV  M,A        FF В ПАМЯТЬ
11 0006 F5              PUSH PSW        СОХРАНИТЬ СОДЕРЖАНИЕ АККУМУЛЯТОРА
12 0007 3E 17           MVI  A,17H
13 0009 BC              CMP  H          ПРОВЕРИТЬ СТАРШИЙ БАЙТ=ПРЕДЕЛ. ЗНАЧ.
14 000A CA 12 00        JZ   LCHK       ДА,ИДТИ НА ПРОВЕРКУ МЛАДШЕГО БАЙТА
15 000D F1              POP  PSW        ВОССТАНОВИТЬ АККУМУЛЯТОР
16 000E 23              INX  H          УВЕЛИЧИТЬ АДРЕС ОЗУ
17 000F C3 05 00        JMP  CL1        ОБРАБОТАТЬ СЛЕДУЮЩИЙ АДРЕС
18 0012 F1       LCHK   POP  PSW        ВОССТАНОВИТЬ АККУМУЛЯТОР
19 0013 BD              CMP  L          ПРОВЕРИТЬ МЛАДШИЙ БАЙТ=FF
20 0014 CA 1B 00        JZ   BEGIN      КОНЕЦ ЭТОЙ ПОДПРОГРАММЫ
21 0017 23              INX  H          УВЕЛИЧИТЬ АДРЕС ОЗУ
22 0018 C3 05 00        JMP  CL1        ПОВТОРИТЬ ЗАНОВО С ДРУГИМ АДРЕСОМ
```

Рисунок 1.1 – Лістинг програми на асемблері

Тому фахівці з комп’ютерної техніки повинні вміти переводити числа з однієї системи числення в іншу. Щоб перевести ціле число з однієї системи числення з основою d1 в іншу з основою d2, необхідно послідовно ділити це число і одержувані частки на основу d2 нової системи доти, поки не вийде частка менше за основу d2 [1, с. 27]. Остання частка – старша цифра числа в новій системі числення з основою d2, а наступні за нею цифри – це залишки від ділення, записуються в послідовності, зворотній їхньому отриманню. Арифметичні дії потрібно виконувати в тій системі числення, у якій записано число, яке перетворюється.

Приклад 1.1. Перевести число 11_{10} у двійкову систему числення (рис. 1.2)

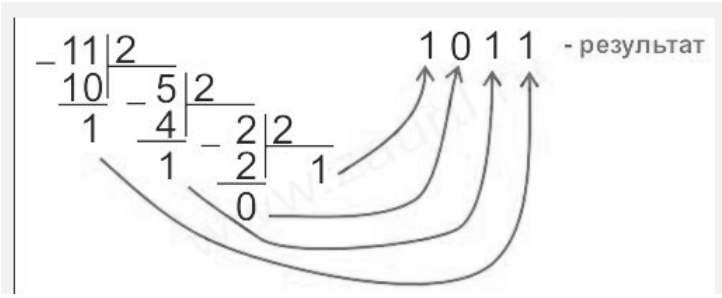

Рисунок 1.2 – Процес перетворення числа
у двійкову систему числення

Відповідь: $11_{10} = 1011_2$.

Приклад 1.2. Перевести число 122_{10} у вісімкову систему числення.

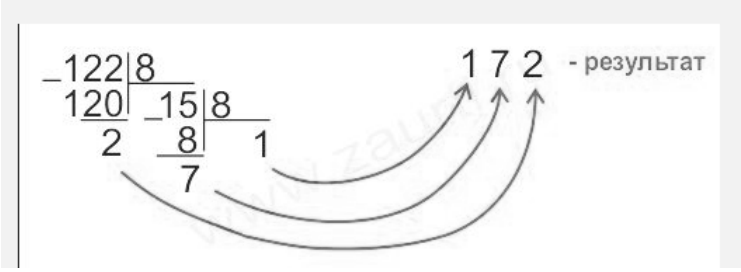

Рисунок 1.3 – Процес перетворення числа
у вісімкову систему числення

Відповідь: $122_{10} = 172_8$.

Приклад 1.3. Перевести число 500_{10} в шістнадцяткову систему числення.

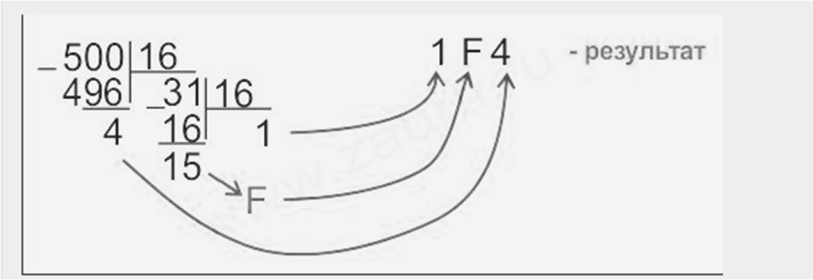

Рисунок 1.4 – Процес перетворення числа
у шістнадцяткову систему числення

Відповідь: $500_{10} = 1F4_{16}$.

Примітка. У шістнадцятковій системі числення позначення цифр трактується так: A – 10_{10}, B – 11_{10}, C – 12_{10}, D – 13_{10}, E – 14_{10}, F – 15_{10}, інші цифри, як і в десятковій системі, тобто $1_{16} = 1_{10}$, $4_{16} = 4_{10}$, $9_{16} = 9_{10}$.

1.1.2 Переклад правильних дробів з однієї позиційної системи числення в іншу

Щоб перевести правильний дріб з системи числення з основою d1 у систему з основою d2, необхідно послідовно множити вихідний дріб і дробові частини добутків, які утворюються, на основу нової системи числення d2. Правильний дріб числа в новій системі числення з основою d2 формується у вигляді цілих частин добутків, які утворюються, починаючи з першого. Якщо при перекладі виходить дріб у вигляді нескінченного або розбіжного ряду, процес можна закінчити після досягнення необхідної точності. При перекладі змішаних чисел необхідно в нову систему перевести окремо цілу і дробову частини за правилами перекладу цілих чисел і правильних дробів, а потім обидва результати об'єднати в одне змішане число в новій системі числення.

Приклад 1.4. Перевести число $0{,}625_{10\,\text{у}}$ двійкову систему числення.

Рисунок 1.5 – Процес перетворення дробового числа в двійкову систему числення

Відповідь: $0{,}625_{10} = 0{,}101_2$.

Приклад 1.5. Перевести число $0{,}6_{10}$ у вісімкову систему числення.

Рисунок 1.6 – Процес перетворення дробового числа у вісімкову систему числення

Відповідь: $0{,}6_{10} = 0{,}463_8$.

Приклад 1.6. Перевести число $0,7_{10}$ у шістнадцяткову систему числення.

Рисунок 1.7 – Процес перетворення дробового числа
у шістнадцяткову систему числення

Відповідь: $0,7_{10} = 0, B33З_{16}$.

1.1.3 Переклад двійкових, вісімкових і шістнадцяткових чисел у десяткову систему числення

Для перекладу числа з р-ічної системи числення в десяткову необхідно використовувати таку формулу розкладання:

$$a_n a_{n-1} a_{n-2} \ldots a_1 a_0, b_1 b_2 \ldots b_m = a_n \cdot p^n + a_{n-1} \cdot p^{n-1} + a_{n-2} \cdot p^{n-2} + \ldots$$
$$\ldots + a_1 \cdot p + a_0, b_1 \cdot p^{-1} + b_2 \cdot p^{-2} + \ldots + b_m \cdot p^{-m}$$

Приклад 1.7. Перевести число $101,11_2$ у десяткову систему числення.

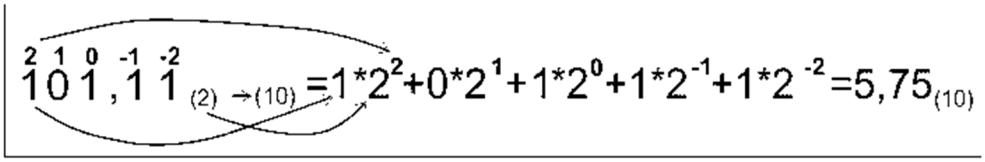

Рисунок 1.7 – Перетворення двійкового числа
у десяткову систему числення

Відповідь: $101,11_2 = 5,75_{10}$.

Приклад 1.8. Перевести число $57,24_8$ у десяткову систему числення.

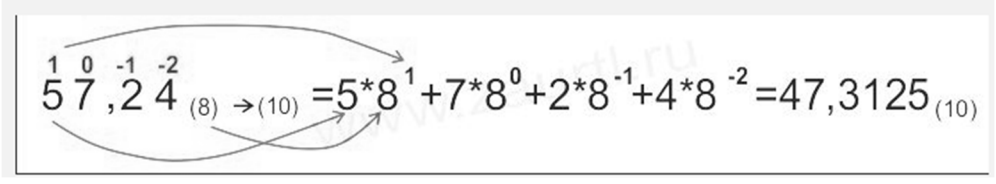

Рисунок 1.8 – Перетворення вісімкового числа
у десяткову систему числення

Відповідь: $57,24_8 = 47,3125_{10}$.

Приклад 1.9. Перевести число 7A, 84_{16} у десяткову систему числення.

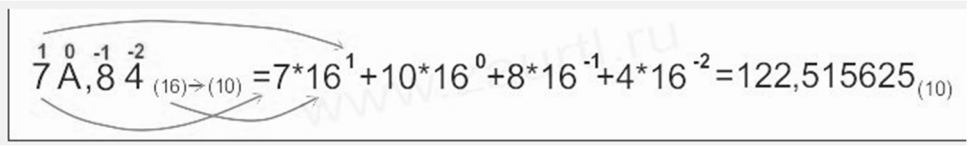

Рисунок 1.9 – Перетворення шістнадцяткового числа
у десяткову систему числення

Відповідь: 7A, 84_{16} = 122,515 625_{10}.

1.1.4 Переклад вісімкових і шістнадцяткових чисел у двійкову систему числення і навпаки

Для перекладу числа з вісімковій системи числення в двійкову необхідно кожну цифру цього числа записати трирозрядним двійковим числом (тріадою).

Приклад 1.10. Записати число $16,24_8$ в двійковій системі числення.

Рисунок 1.10 – Перетворення вісімкового числа в двійкове

Відповідь: $16,24_8 = 1110,0101_2$.

Примітка. Незначущі нулі зліва для цілих чисел і праворуч для дробів не записуються. Для зворотного перекладу двійкового числа в вісімкову систему числення, необхідно вихідне число розбити на тріади вліво і вправо від коми і представити кожну групу цифрою в восьмирічній системі числення. Крайні неповні тріади доповнюють нулями.

Приклад1.11. Записати число $1110,0101_2$ в вісімковій системі числення.

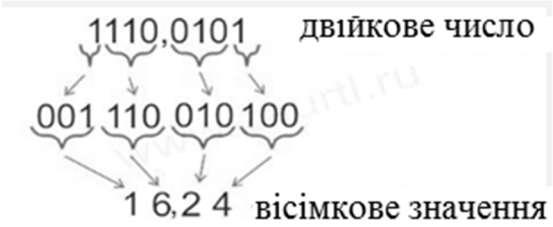

Рисунок 1.11 – Перетворення двійкового числа в вісімкове

Відповідь: 1110,010 1_2= $16,24_8$.

Для перекладу числа з шістнадцятковій системи числення в двійкову необхідно кожну цифру цього числа записати чотирирозрядним двійковим числом (тетрадою).

Приклад 1.12. Записати число 7A, 7E16 у двійковій системі числення.

Рисунок 1.12 – Перетворення шістнадцяткового числа в двійкове

Відповідь: 7A, $7E_{16}$ = +1111010,0111111_2.

Примітка. Незначущі нулі зліва для цілої частини і праворуч для дробової частини у результат не заносяться.

Для зворотного перекладу двійкового числа в шістнадцяткову систему числення необхідно вихідне число розбити на тетради вліво і вправо від коми і представити кожну групу цифрою в шістнадцятковій системі числення. Крайні неповні тріади доповнюють нулями.

Приклад 1.13. Записати число $1111010,0111111_2$ у шістнадцятковій системі числення.

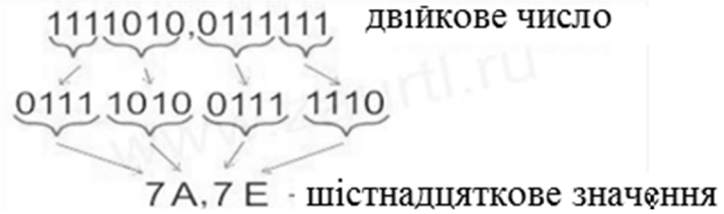

Рисунок 1.13 – Перетворення двійкового числа в шістнадцяткове

Відповідь: $1111010,0111111_2$ = 7A, $7E_{16}$.

2 Завдання на практичне заняття

1. Перетворити число з десяткової системи числення в шістнадцяткову, вісімкову і двійкову системи. Обчислення виконати з точністю до восьмого двійкового дробового порядку. Варіанти виконання подано в таблиці 1.1.

Таблиця 1.1 – Варіанти до першого завдання

Варі-ант	Число	Варі-ант	Число	Варі-ант	Число	Варі-ант	Число
1	36,76	8	64,47	15	61,52	22	53,64
2	68,56	9	51,49	16	67,32	23	58,52
3	74,54	10	74,09	17	54,85	24	43,91
4	47,87	11	57,53	18	69,53	25	65,83
5	63,76	12	48,52	19	62,71	26	59,21
6	37,94	13	65,21	20	56,43	27	77,07
7	41,53	14	52,21	21	42,65	28	46,47

2. Число в шістнадцятковій системі представити в десятковій, вісімковій і двійковій системах числення.

Таблиця 1.2 – Варіанти до другого завдання

Варі-ант	Число	Варі-ант	Число	Варі-ант	Число	Варі-ант	Число
1	3F,7B	8	C4,D7	15	C1,5D	22	AA,6B
2	D8,A6	9	5D,4A	16	D7,E2	23	BD,52
3	7F,E4	10	B4,A9	17	E4,8F	24	BE,91
4	4A,8B	11	5C,5A	18	F9,5A	25	BF,83
5	F3,F6	12	E8,5B	19	A2,C1	26	CA,2A
6	A7,9B	13	A5,2F	20	BA,4B	27	DA,07
7	4F,5C	14	B2,C1	21	CD,65	28	DE,4A

Контрольні запитання і завдання

1. Чому інформація в комп’ютері зберігається в двійковому вигляді ?
2. Що таке двійкова система числення?
3. Що таке вісімкова система числення?
4. Що таке шістнадцяткова система числення?
5. Як позначаються цифри в шістнадцятковій системі числення?
6. Навіщо фахівці з комп’ютерної техніки повинні вміти переводити числа з однієї системи числення в іншу ?
7. Сформулюйте загальний алгоритм перетворення чисел із однієї системи числення в іншу.
8. Сформулюйте алгоритм перетворення чисел із десяткової системи числення в двійкову.
9. Сформулюйте загальний алгоритм перетворення чисел із двійкової системи числення в десяткову.
10. Сформулюйте загальний алгоритм перетворення чисел із двійкової системи числення в вісімкову та шістнадцяткову.

2 ВИКОНАННЯ ОПЕРАЦІЙ СКЛАДАННЯ І ВІДНІМАННЯ В КОМП'ЮТЕРІ

Мета заняття: освоєння технології виконання операцій складання і віднімання в комп'ютері в додатковому коді.

1 Теоретична частина

2.1.1 Представлення чисел у комп'ютері в додатковому коді

Практично будь-яку операцію, що виконується на комп'ютері, можна звести до послідовності операцій складання і віднімання. Наприклад, відомо розкладання функції $\sin x$ (x виражене в радіанах, 1 радіан $=\frac{180^0}{\pi}=57,296^0$, π=3,141592.) у ряд [2, с. 222]:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + ...$$

Де $n! = 1 \cdot 2 \cdot 3 \cdot ... \cdot (n-1) \cdot n$, $x^3 = x \cdot x \cdot x$, $x^5 = x \cdot x \cdot x \cdot x \cdot x$. Тоді

$$\sin 30^0 = \sin \pi \Big/ 6 \approx \frac{\pi}{6} - \left(\pi/6\right)^3 \Big/ 3! + \left(\pi/6\right)^5 \Big/ 5! - \left(\pi/6\right)^7 \Big/ 7! = 0,4999999921 \approx 0,5$$

Отже, можна істотно спростити апаратну частину процесора, звівши виконання складних операцій до виконання відповідних мікропрограм, у яких основні операції арифметичні: складання, віднімання, множення і ділення. Зі свого боку, операції множення і ділення можна звести до послідовності операцій складання і віднімання. Наприклад: $5 \cdot 12 = 12 + 12 + 12 + 12 + 12$. Щоб ще спростити апаратну частину процесора, операцію віднімання в двійковому коді (додавання числа з від'ємним знаком) зводять до операції складання в спеціальному коді. Один із них – додатковий [3, с. 50]. Додатковий код додатного числа співпадає з прямим кодом. У такому вигляді число записується в регістр. Потрібно враховувати, що в регістрі старший розряд відводиться під знак числа, 0 – додатне число, 1 – від'ємне число. Наприклад, потрібно записати число +4 в шестирозрядний регістр. У ньому крайній правий розряд вважається нульовим, старший, п'ятий розряд відводиться під знак, а решта п'ять – під код числа. Перетворимо число 4 в двійковий код. $4_{10} = 100_2$. Третій і четвертий розряди заповнюємо нулями, а в п'ятий заносимо знак – нуль. Таким чином, додатне число 100_2 зберігатиметься в шести розрядному регістрі у вигляді 000100.

Від’ємне число в двійковому виді перетворюються в число, що зберігається в регістрі процесора в додатковому коді, таким чином:

1. Десяткове число перетвориться в двійкове (для прикладу візьмемо -4, регістр шести розрядний). $-4_{10} \rightarrow -100_2$.

2. Ліворуч число доповнюємо нулями, включаючи знаковий розряд: $-100 \rightarrow -000100$.

3. Інвертуємо число, тобто кожен розряд замінюваний на протилежний без урахування знаку : $000100 \rightarrow 111011$.

До сформованого числа додаємо 1 в молодшому, нульовому розряді. Таким чином $111011 \Rightarrow$

Переніс		немає	немає	немає	←1	←1
	--	--	--	--	--	--
	1	1	1	0	1	1
+						
	0	0	0	0	0	1
	==	===	==	==	==	==
Результат	1	1	1	1	0	0

Отже число -4 буде зберігатися в шести розрядному регістрі у вигляді 111100.

Для зворотного перетворення від’ємного числа в додатковому коді в число в прямому коді зі знаком, наведені вище дії потрібно виконувати в зворотному порядку: від числа в додатковому коді віднімається 1, результат інвертується, додається знак і за необхідності перетворюється в число в десятковому поданні. Приклад: перетворимо число в додатковому коді 111100 в прямий код. Від числа віднімається одиниця:

	1	1	1	1	0	0
-						
	0	0	0	0	0	1
		===	==	==	==	==
Результат	1	1	1	0	1	1

Пояснення: з другого розряду зайняли 4 (оскільки в першому знаходиться нуль). $4_{10} - 1_{10} = 011_2$. Отримали 111011. Інвертуємо його і додаємо знак. $111011 \rightarrow -000100$. А це -4_{10}.

2.1.2 Додавання чисел у додатковому коді

Операція складання чисел у двійковому коді виконується аналогічно операції складання в десятковому коді, з урахуванням того, що максимальне

число в одному розряді 1, тобто $1+0=1,\ 0+1=1,\ 1+1=0$ плюс перенесення 1 в наступний старший розряд.

Додатні числа в додатковому коді складаються аналогічно числам у прямому коді. Якщо знак результату став дорівнювати 1, то це означає переповнення розрядної сітки процесора, не виконання операції і переривання програми. Приклад: $5_{10} + 9_{10} = 101_2 + 1001_2 = 000101 + 001001 \Rightarrow$

Перенесення		немає	немає	немає	немає	←1
	---	------	-------	--------	--------	-----
	0	0	0	1	0	1
+						
	0	0	1	0	0	1
		===	==	==	==	==
Результат	0	0	1	1	1	0

Тобто $000101 + 001001 = 001110_2 = 14_2 = 5_{10} + 9_{10}$.

Для складання додатного і від'ємного чисел, спочатку від'ємне число перетворюється в додатковий код, числа складаються; перенесення з старшого розряду ігнорується. Якщо знаковий розряд дорівнює нулю, то результат додатний, і відразу отримали число в прямому коді. Якщо знаковий розряд дорівнює одиниці, то отримали від'ємне число в додатковому коді. Його необхідно перетворити в прямий код із знаком.

Приклади. Складемо 13_{10} + (- 7_{10}). Результат дорівнює $+6_{10}$. Виконаємо це в двійковому вигляді. Перетворимо обидва числа в шестирозрядний додатковий код зі знаком.

$$13_{10} = 1101_2 = 001101, -7_{10} = -111_2 = -000111 \to 111000_{звор} + 1 = 111001_{додатн}.$$

Складаємо обидва числа:

Перенесення	←1	←1	←1	немає	немає	←1
	---	---	---	--------	-------	---
	0	0	1	1	0	1
+						
	1	1	1	0	0	1
	==	===	==	====	====	==
Результат	0	0	0	1	1	0

Отже результат дорівнює 000110_2, а це $+6_{10}$.

Скласти (-13_{10}) і $+7_{10}$. Результат дорівнює - 6_{10}. Виконаємо це в двійковому вигляді. Перетворимо обидва числа в шестирозрядний додатковий код зі знаком.

$$-13_{10} = -1101_2 = -001101 \to 110010_{звор} \to 110011_{додатн},$$
$$+7_{10} = 111_2 = 000111_{прям} \to 000111_{додатн}.$$

Складаємо обидва числа:

Перенесення	немає	немає	немає	немає	←1	←1
	1	1	0	0	1	1
+						
	0	0	0	1	1	1
		===	==	==	==	==
Результат	1	1	1	0	1	0

Отримали результат 111010 у додатковому коді. Він від'ємний, оскільки старший розряд дорівнює одиниці. Перетворимо його в прямий код із знаком. Спочатку віднімаємо 1.

	1	1	1	0	1	0
-						
	0	0	0	0	0	1
		===	==	==	==	==
Результат	1	1	1	0	0	1

Отримали 111001. Інвертуємо його і додаємо знак. $111001 \rightarrow -000110_2$. А це -6_{10}.

Якщо складаються два від'ємних числа в додатковому коді, то може виникнути переповнення розрядної сітки. Ознакою цього є відмінність знакового розряду результату (0) від знакових розрядів доданків, які одночасно дорівнюють 1. У цьому випадку результат ігнорується. Наведемо приклад для шестирозрядного регістра. Максимальні за модулем числа, які можуть у ньому зберігатися, це $\pm 11111_2 = \pm 31_{10}$.

Приклад. (- 20) + (- 17) = - 37 – переповнення. Проведемо обчислення в двійковому вигляді без докладного коментаря.

$$-20_{10} = -010100_2 \rightarrow 101011_{звор} \rightarrow 101011 + 1 = 101100_{додатн},$$
$$-17_{10} = -010001_2 \rightarrow 101110_{звор} \rightarrow 101110 + 1 = 101111_{додатн}.$$

Складаємо обидва числа:

Перенесення	←1	немає	←1	←1	немає	немає
	---	-----	---	---	-----	-----
	1	0	1	1	0	0
+						
	1	0	1	1	1	1
	==	===	==	==	===	===
Результат	0	1	1	0	1	1

Знак результату відрізняється від знака доданків – переповнення.

Завдання на практичне заняття.

Дано числа *b* і *c* у десятковому поданні. Виконати:

1. Знайти $A_{10} = b - c = b + (-c)$ десятковому поданні.

2. Перетворити b і c у двійковий додатковий код і знайти $A_d = b_d + c_d$. Для зберігання чисел використовується шестирозрядний регістр.

3. Перетворити A_d у прямий двійковий код $A_{2\,пр}$.

4. Перетворити $A_{2\,пр}$ у десяткове подання $\widetilde{A}_{10}$.

5. Порівняти A_{10} і $\widetilde{A}_{10}$, зробити висновки і оформити звіт.

Варіанти задачі задані у таблиці 2.1.

Таблиця 2.1 – Варіанти до задачі

Варіант	*b*	*c*	Варіант	*b*	*c*
1	10	25	16	12	27
2	11	30	17	13	28
3	8	21	18	19	22
4	-13	22	19	17	24
5	9	24	20	-13	27
6	11	26	21	11	24
7	17	30	22	9	30
8	15	28	23	10	23
9	13	22	24	14	21
10	14	21	25	15	24
11	-15	26	26	16	25
12	17	24	27	17	30
13	19	30	28	18	29
14	12	29	29	20	27
15	-13	22	30	-21	14

Контрольні запитання

1. Як можна обчислення функції звести до послідовності операцій складання і віднімання ?
2. Навіщо операцію віднімання у комп’ютері зводять до операції додавання?
3. Опишіть технологію перетворення двійкового числа в двійкове у додатковому коді.
4. Як розпізнати, що двійкове число від’ємне?
5. Опишіть технологію складання двох двійкових чисел.
6. Що таке переповнення розрядної сітки комп’ютера?
7. Як розпізнати, що результатом операції додавання є переповнення розрядної сітки комп’ютера?

3 ПЕРЕТВОРЕННЯ ЧИСЕЛ ІЗ ДЕСЯТКОВОЇ СИСТЕМИ ЧИСЛЕННЯ В ДВІЙКОВУ В ФОРМАТІ З ПЛАВАЮЧОЮ КОМОЮ В НОРМАЛІЗОВАНОМУ ВИГЛЯДІ

1 Теоретична частина

У комп'ютерних системах один двійковий розряд, який може набувати два різних значення «0» та «1», є найменшою одиницею інформації, яка називається бітом. Набір відповідної кількості бітів використовується для представлення багаторозрядного двійкового числа (або в загальному випадку – двійкового коду слова). Розрядність слова може бути від одного біта до довільної кількості *n* бітів. Слово із 8 бітів називають байтом. Зазвичай, коли йдеться про комп'ютерну техніку, всі виміри кількості розрядів наводяться в бітах або в байтах. Часто словом іще називають число із 32 бітів, а число із 16 бітів – півсловом

Коли деяке число має 32 біти, то говорять, що воно представлене з одинарною точністю, якщо ж 64 біти – з подвійною точністю.

У сучасних комп'ютерах слово може мати 64 біта, тоді подвійне слово складається з 128 біт.

Числові дані в комп'ютері представляють трьома способами [4, с. 63].

– Як цілі або дробові числа з фіксованою комою, які складаються із деякої кількості бітів;

– Як числа з рухомою (ще деколи називають з «плаваючою») комою кожне з яких має порядок та мантису;

– Як двійково-кодовані десяткові, де байт (чи пів байта) представляє число.

Якщо певне число більше за максимальне, яке може бути представлене певною кількістю розрядів, то значення числа може втрачатися. Таку ситуацію називають переповненням. Розробники комп'ютерних систем повинні передбачити, числа якої величини будуть використовуватись і виділити для їхнього представлення таку кількість розрядів, щоб значення числа не втрачалося.

Сучасні комп'ютери використовують слова розрядністю від одного до 16 байтів. Спеціалізовані комп'ютери можуть використовувати слова розрядністю 128 байт.

Для кодування символів використовуються спеціальні коди, серед яких найпоширеніший – американський стандартний код інформаційного обміну

ASCII, а в менфреймах – розширений двійково-кодований десятковий код обміну EBCDIC. Зазвичай для представлення одного символа використовується один байт.

3.1.1 Числа з фіксованою комою

У разі використання чисел із фіксованою комою вважається, що кома стоїть на певній наперед відомій позиції відносно розрядів числа. Найчастіше вважається, що кома стоїть біля молодшого розряду (у такий спосіб представляються дробові числа), хоча можливе і застосування змішаного варіанта. У такому форматі представляються числа з діапазону -1 < число < 1 (якщо є знаковий розряд) або 0 < число < 1 (якщо знакового розряду немає).

На рисунку 3.1 показано приклад розрядної сітки комп'ютера (формату даних) для представлення двійкових чисел з фіксованою комою. Розряди пронумеровані зліва направо.

Рисунок 3.1 – Розрядна сітка при представленні двійкових чисел із фіксованою комою

Комп'ютери, які опрацьовують числа у форматі з фіксованою комою, є простішими та швидшими порівняно з комп'ютерами, які опрацьовують числа у форматі з рухомою комою,: але в них можливе виникнення проблем через потребу передбачення переповнення. Перші комп'ютери опрацьовували дані з фіксованою комою, крім того, кома за звичай фіксувалась перед старшим розрядом. Зараз представлення чисел із фіксованою комою використовується як єдине ціле в порівняно невеликих за своїми обчисленнями можливостями комп'ютерах, які використовуються для управління технологічними процесами та опрацювання вимірювальної інформації в реальному часу.

У комп'ютерах, призначених для вирішення широкого кола обчислювальних задач, основним є представлення чисел із рухомою комою, яке не вимагає масштабування даних.

Однак у таких комп'ютерах крім представлення чисел у цьому форматі, часто використовується представлення з фіксованою комою, оскільки на виконання операцій з такими числами витрачається менше часу. Крім цього здебільшого формат чисел із фіксованою комою слугує для представлення цілих двійкових чисел (кома ставиться праворуч від молодшого розряду числа) та виконання операцій над ними, що, зокрема, необхідно для операцій над кодами адресу (операції індексної арифметики).

3.1.2 Числа з рухомою комою

Для представлення достатньо малих і великих чисел використовують показникову форму у вигляді $\pm mq^{\pm p}$, де m – мантиса числа, q – основа порядку, p - порядок числа. Наприклад $-0{,}125\times10^{12}=-125\,000\,000\,000$. Зауважимо, що у більшості калькуляторів це буде виглядати як $-0{,}125e12$. У комп'ютері показникова форма числа записується у вигляді числа з рухомою комою. Для зберігання мантиси і порядку, а також знаків відводяться певні розряди.

Очевидно, що діапазон чисел, що подаються в цифрових пристроях з плаваючою комою, значно більший, ніж у пристроях із фіксованою комою:

Для однозначного і максимально точного відображення чисел з рухомою комою їх представляють у нормалізованому вигляді. Якщо виконується нерівність $q-1\le m<1$, а у випадку двійкової системи числення $0{,}5\le m<1$ (старший двійковий розряд мантиси дорівнює 1), то вважається, що число представлене в нормалізованому вигляді. Отже у двійкового нормалізованого числа у форматі з рухомою комою мантиса є правильним дробом і у старшому розряді мантиси завжди стоїть 1. Нормалізація чисел у комп'ютері виконується або апаратно або ж спеціальною програмою. Для цієї мети виділяється:

– по одному розряду для представлення знаку числа S_m і знаку порядку S_p;

– певне число розрядів для представлення значення порядку p;

– розряди для представлення модуля мантиси.

Наприклад, можливий такий варіант коли формат числа складається з чотирьох полів (рис. 3.2)

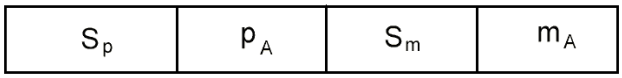

Рисунок 3.2 – Число з рухомою комою

Під час арифметичних операцій над числами з рухомою комою виконуються дії як над порядком, так і над мантисою.

Для спрощення операцій над порядками їх зводять до дій над цілими додатними числами (цілими числами без знаків), застосовуючи уявлення чисел із плаваючою комою з «зміщеним порядком». Зміщений порядок *r* обчислюється за формулою r = ± p + z, де z – зміщення, значення якого підбирається так, щоб у разі зміни значення показника від деякого мінімального значення $-|p_{\min}|$ до максимального $+|p_{\max}|$ зміщений порядок r змінювався від 0 до r_{max}.

Отже, характеристика не змінює свого знаку. У цьому випадку відпадає необхідність у відображенні знаку порядку S_p . Для цього приймається, що $z = 2^{k-1}$, де *к* – число розрядів, виділених для представлення порядку числа у форматі з рухомою комою.

Тоді формат числа з рухомою комою можна подати так, як показано на рис. 3.3 (з використанням трьох полів).

Рисунок 3.3 – Формат числа з рухомою комою і зміщеним порядком

Одиниця старшого розряду нормалізованої мантиси зазвичай не відображається у форматі числа, тобто є уявною. Розряд слова, у якому повинна була бути відображена ця одиниця, використовується як молодший розряд зміщеного порядку, або старший розряд мантиси, що дозволяє збільшити діапазон представлення чисел у форматі з рухомою комою, або точність обчислень.

Таким чином, мантиса в такому варіанті відображається, починаючи з розряду, що йде після старшого. Це потрібно враховувати під час виконання будь-якої операції з мантисою числа, і перед початком операцій відновлювати старший розряд мантиси. Після завершення операцій формування нормалізованого результату у відведеній для нього розрядній сітці, старша одиниця мантиси знову відкидається. Порядок із к-розрядним полем може змінюватися в межах від -2^{k-1} до $2^{k-1}-1$ (табл. 3.1, к = 3).

Таблиця 3.1 – Формування показника

Показник порядку	Прямий код показника	Зміщений порядок (показник $+2^{3-1}$ - 1)	Примітки
+3	011	110	3 + 3 = 6
+2	010	101	2 + 3 = 5
+ 1	001	100	1+3 = 4
0	000	011	0 + 3= 3
-1	101	010	-1+3 = 2
-2	110	001	-2 + 3 =12
-3	111	000	-3 + 3=0

Як зазначалося, зміщений порядок r – це порядок p з надлишком $z = 2^{k-1} - 1$. Він не змінює свого знаку і змінюється від 0 (за $p = -2^{k-1} + 1$) до $2^k - 2$ (за $p = 2^{k-1} - 1$).

Основною перевагою представлення чисел у форматі з рухомою комою є великий діапазон машинних чисел і висока точність їхнього подання. Діапазон визначається довжиною розрядної сітки, виділеної для характеристики, а точність визначається довжиною розрядної сітки, виділеної для мантиси.

Особливості виконання операцій над числами з рухомою комою:

– збільшення мантиси у 2 рази здійснюється зсувом двійкового значення мантиси ліворуч (у бік старших розрядів);

– зменшення мантиси у 2 рази здійснюється зсувом двійкового значення мантиси праворуч (у бік молодших розрядів);

– величина числа не зміниться, якщо збільшити мантису в 2 рази і одночасно зменшити порядок на 1;

– величина числа не зміниться, якщо зменшити мантису в 2 рази і одночасно збільшити порядок на 1.

Розглянемо на наступному прикладі алгоритм перетворення дійсного числа з десяткової системи поданні в двійкову систему числення в нормалізованому форматі з плаваючою комою за таких передумов: розрядність двійкового слова 16, під знак мантиси відводиться старший, 15-й розряд, під порядок відводиться п’ять розрядів з 10-го до 14-й, під мантису без старшого розряду відводиться 10 розрядів, з 0-го до 9-го. Число, яке потрібно перетворити число, це число -13,4910.

16-розрядне двійкове слово має вигляд (рис. 3.4):

розряди	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
зміст	1/0	1/0	1/0	1/0	1/0	1/0	1/0	1/0	1/0	1/0	1/0	1/0	1/0	1/0	1/0	1/0

Рисунок 3.4 – Вигляд двійкового числа у пам’яті комп’ютера

А. Перетворення в двійкову систему числення.

А.1. Оскільки число від’ємне, то в 15-му (знаковому) розряді буде 1 (для додатного числа 15-й розряд буде дорівнювати 0).

А.2. Перетворимо цілу частину числа (13_{10}) у двійкову систему за допомогою послідовного ділення на 2 (рис. 3.5)

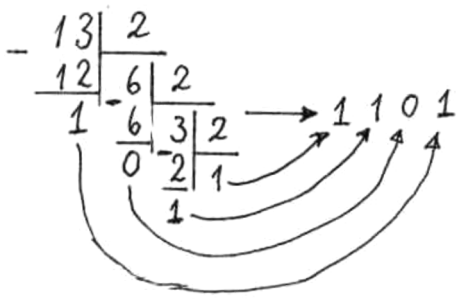

Рисунок 3.5 – Перетворення цілої частини десяткового числа в двійкове

Отже $13_{10} = 1101_2$.

А.3. Перетворимо дробову частину числа ($0{,}4910_{10}$) у двійкове подання за допомогою послідовного множення на 2. Перетворення необхідно виконувати доти, поки кількість цифр у цілій частині двійкового представлення і в дробової частини двійкового представлення не стане дорівнювати 11. Таким чином, перетворення дробової частини здійснюється з певною похибкою (рис. 3.6).

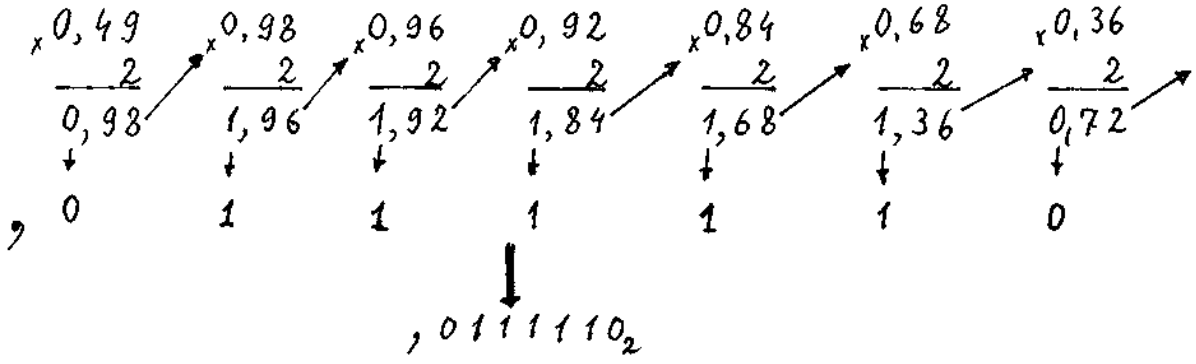

Рисунок 3.6 – Перетворення дробової частини десяткового числа в двійкове подання

Отже $0{,}49_{10} = 0{,}0111110_2$.

А.4. Формуємо модуль вихідного числа в двійковому поданні:

$13{,}49_{10} = 1101{,}0111110_2$.

Б. Нормалізація модуля двійкового представлення числа.

Б.1. Зміщуємо двійкове число вправо на n розрядів, поки перед комою не залишиться 1 і одночасно множимо число на 2^n, де n – це ненормалізований порядок і в цьому прикладі n = 3 і

$$+1101{,}0111110_2 = +1{,}1010111110 \cdot 2^3.$$

☛ ***Примітка.*** Якщо ціла частина числа відсутня, то тоді зміщуємо двійкове число вліво на n розрядів, поки перед комою не з'явиться 1 і одночасно множимо число на 2^{-n}, де -n – це ненормалізований порядок.

Б.2. Розраховуємо зміщення z. $z = 2^{k-1} - 1 = 2^{5-1} - 1 = 15$. До числа n ($3_{10}$) додаємо зміщення ($15_{10}$) і отримуємо зміщений порядок $r = 3 + 15 = 18_{10}$ у десятковій системі числення.

б.3. Переводимо r_{10} у двійкове число (рис. 3.7):

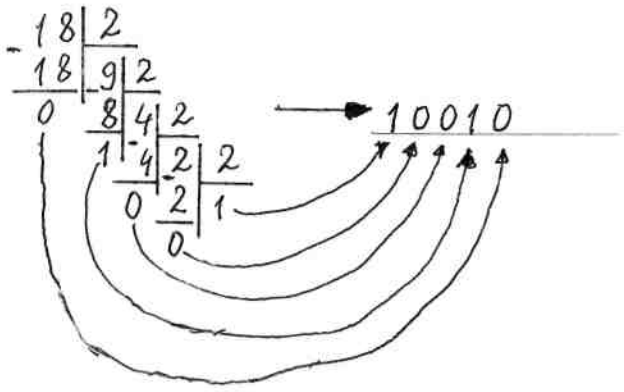

Рисунок 3.7 – Перетворення числа r в двійкове подання

Отже зміщений порядок у двійковій системі числення має вигляд: $r_2 = 10010$.

В. Збірка числа в двійковому форматі.

В1. У 15-й, знаковий розряд запишемо 1 (від'ємне число).

В2. Зміщений порядок дорівнює 10010 (обов'язково п'ять розрядів, розряди, значення яких бракує, доповнюються нулями зліва. Його заносимо в розряди з 10-го до 14-го.

В3. Мантиса дорівнює +1,1010111110. Відкидаємо 1, що стоїть до коми і число після коми (1010111110) заносимо в розряди з 0-го по 13-й.

В4. Отримали двійкове число: 1100101010111110. У пам'ять комп'ютера число запишеться в такий спосіб (рис. 3.8):

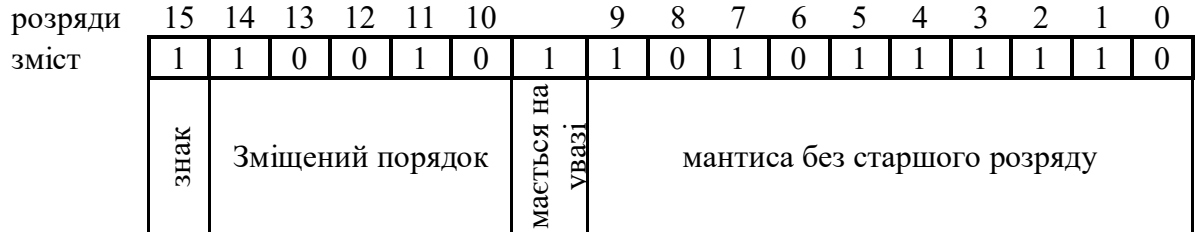

Рисунок 3.8 – Число $-13{,}49_{10}$ у пам'яті комп'ютера

Г. Переведемо число 1100101010111110 з двійкового у шістнадцятковий формат. Розбиваємо вихідне число 1100101010111110 з права наліво по 4 розряди (тетради) і кожну двійкову тетраду замінюємо її шістнадцятковим числом. Отримаємо шістнадцятковий вигляд вихідного числа $-13{,}49_{10}$, як воно буде зберігатися в комп'ютері у форматі з плаваючою комою (рис 3.9):

1	1	0	0	1	0	1	0	1	0	1	1	1	1	1	0
12_{10}				10_{10}				11_{10}				14_{10}			
C				A				B				E			

Рисунок 3.9 – Перетворення числа з двійкового у шістнадцятковий формат

Тобто $CABE_{16}$.

Д. Перевірка. Перетворимо внутрішнє представлення вихідного числа ($CABE_{16}$) у зовнішнє у десятковому поданні.

Д1. Кожну шістнадцяткову цифру замінюємо двійковим еквівалентом. Отримуємо: CABE → 1100 1010 1011 1110 → 1100101010111110.

Число від'ємне, оскільки старший розряд дорівнює 1.

Д2. Виділяємо розряди з 14-го до 10-го – 10010. Це зміщений порядок. Переводимо його в десяткове подання. $10010_2 \rightarrow 18$. Віднімаємо зміщення 15. Отримуємо звичайний порядок числа – $p = 3$.

Д3. Виділяємо розряди з 9-го до 0-го – 1010111110; це нормалізована мантиса. Зліва додаємо 1, а після неї ставимо кому. Отримуємо значення вихідного числа без урахування зсуву. 1,1010111110. Переводимо його в десяткове подання. Для полегшення рахунку для кожного розряду записуємо положення після коми (табл. 3.2):

Таблиця 3.2 – Положення кожної цифри двійкового числа після коми

Двійкове число	1	0	1	0	1	1	1	1	1	0
Положення цифр після коми	1	2	3	4	5	6	7	8	9	10

Записуємо число в природному вигляді:

$$1{,}1010111110_2 = 1 + \frac{1}{2} + \frac{0}{2^2} + \frac{1}{2^3} + \frac{0}{2^4} + \frac{1}{2^5} + \frac{1}{2^6} + \frac{1}{2^7} + \frac{1}{2^8} + \frac{1}{2^9} + \frac{0}{2^{10}} =$$

$$= 1 + \frac{1}{2} + \frac{0}{4} + \frac{1}{8} + \frac{0}{16} + \frac{1}{32} + \frac{1}{64} + \frac{1}{128} + \frac{1}{256} + \frac{1}{512} + \frac{0}{1024} \approx 1{,}6855469_{10}.$$

Д4. Число, яке отримали множимо на $2^p = 2^3 = 8$ і записуємо його як від'ємне $1{,}6855468 \times 8 \Rightarrow -13{,}484375$. Порівнюючи це число з вихідним (-13,49), доходимо до висновку, що запис дійсного числа в пам'ять комп'ютера здійснюється з певною похибкою.

Д5. Обчислимо цю похибку:

$$\delta\% = \left| \frac{(-13{,}49) - (-13{,}484375)}{(-13{,}49)} \right| \cdot 100\ \% = 0{,}041\ \%.$$

2 Завдання на практичне заняття

1. Перетворити дійсне число *N* з десяткової системи числення в двійкову систему числення в нормалізованому форматі з плаваючою комою за таких передумов: розрядність двійкового слова – 16 (це півслова 32-розрядного процесора), під знак мантиси відводиться старший, 15-й розряд, під порядок відводиться п'ять розрядів з 10-го до 14-го, під мантису без старшого розряду відводиться 10 розрядів, з 0-го до 9-го.

2. Перевести двійковий код числа у шістнадцятковий формат.

3. Провести перевірку, перетворивши внутрішнє представлення вихідного числа у зовнішнє у десяткове подання.

4. Обчислити відносну похибку при запису дійсного числа в пам'ять комп'ютера.

Варіанти задачі задані у таблиці 3.3.

Таблиця 3.3 – Варіанти до практичного заняття

Варіант	N	Варіант	N	Варіант	N	Варіант	N
1	$-15{,}36_{10}$	9	$-7{,}36_{10}$	17	$-17{,}40_{10}$	25	$21{,}69_{10}$
2	$9{,}44_{10}$	10	$14{,}42_{10}$	18	$24{,}47_{10}$	26	$-29{,}73_{10}$
3	$-7{,}43_{10}$	11	$-17{,}30_{10}$	19	$-28{,}29_{10}$	27	$-25{,}88_{10}$
4	$11{,}39_{10}$	12	$20{,}37_{10}$	20	$25{,}45_{10}$	28	$19{,}72_{10}$
5	$-6{,}54_{10}$	13	$-15{,}38_{10}$	21	$-2{,}49_{10}$	29	$-22{,}81_{10}$
6	$23{,}35_{10}$	14	$10{,}32_{10}$	22	$-18{,}76_{10}$	30	$24{,}64_{10}$
7	$-19{,}33_{10}$	15	$-12{,}41_{10}$	23	$-17{,}73_{10}$	31	$0{,}02_{10}$
8	$6{,}31_{10}$	16	$22{,}34_{10}$	24	$24{,}42_{10}$	32	$-0{,}07_{10}$

Контрольні запитання і завдання

1. Що таке біт, байт?
2. Що таке одинарне слово, подвійне слово?
3. Які способи представлення даних використовують в комп’ютерах?
4. Як представляються двійкові числа у форматі з фіксованою комою?
5. Яке мінімальне і максимальне ціле число може зберігатися у шістнадцятки розрядному регістрі?
6. Яке мінімальне і максимальне число може зберігатися у регістрі з форматом за рисунком 3.8?
7. Для чого використовують показникову форму числа?
8. Як формується мантиса числа у форматі з рухомою комою?
9. Як формується порядок числа у форматі з рухомою комою?
10. Як формується зміщений порядок?
11. Під двійкове число відведено 32 розряди. Під порядок відведено 10 розрядів. Зміщення дорівнює 511. Обчислить мінімальне і максимальне по модулю числа, які можна представити за цих умов.

4 ПРОЄКТУВАННЯ КОМБІНАЦІЙНИХ СХЕМ

Мета заняття: освоєння теорії і практики розробки комбінаційних схем у системі повної логіки І, АБО, НІ

1 Теоретична частина

Модулі, з яких будуються комп'ютерні системи, можна умовно розбити на два види: комбінаційні схеми та цифрові автомати (схеми з пам'яттю). В комбінаційній схемі комбінація сигналів на її вході однозначно визначає комбінацію сигналів на виході. Відповідно, значення сигналу на одній лінії виходу також визначається комбінацією сигналів на вході і цю залежність можна описати булевою функцією.

4.1.1 Булеві функції

Комп'ютерні системи зберігають, обробляють і передають інформацію в двійковому вигляді. Тому основним математичним апаратом, який використовується для опису і проєктування комп'ютерних систем є двійкові або булеві функції (їх ще називають логічними функціями) [5, с. 175].

Двійковою змінною називається величина, яка може приймати тільки два значення – 0 або 1. Двійкові змінні позначаються буквами латинського алфавіту: x, y, z, x_1, x_2, x_3 і тому подібне.

Якщо ототожнювати двійкові змінні з логічними, то для них вводяться три операції: додавання (+), множення • і інверсія. Для них справедливі ряд аксіом і законів.

1. Існують такі 0 і 1, що $\overline{0}=1$ і $\overline{1}=0$, $0\cdot 0=0$, $0+0=0$, $1+1=1$, $1\cdot 1=1$, $1\cdot 0=0\cdot 1=0$, $0+1=1+0=1$.

На основі аксіом, виводяться закони булевої алгебри.

1. Закони нульової множини: $0+a=a;\ 0\cdot a=0;\ 0\cdot a\cdot b\cdot\ldots\cdot w=0$.
2. Закони універсальної множини:

$$1+a=1;\ 1\cdot a=a;\ 1+a+b+\ldots+w=1.$$

3. Закони ідемпотентності (повторення, тавтології):

$$a\cdot a=a;\ a+a=a;\ aaa\ldots aaa=a^n=a;\ a+a+a+\ldots+a=a$$

4. Закон двійкової інверсії: $\overline{\overline{a}}=a$.
5. Закони додатковості: $a\cdot\overline{a}=0$, $a+\overline{a}=1$.
6. Закони поглинання:

$a \cdot (a+b) = a$; $a \cdot (a+b) \cdot (a+c)...(a+w) = a$; $a + ab = a$;

$a + ab + ac + ..aw = a$. 7. $a \cdot (\overline{a} + b) = ab$; $a + \overline{a}b = a + b$.

8. Закони склеювання (поширення)

$a \cdot b + a \cdot \overline{b} = a$; $(a+b) \cdot (a+\overline{b}) = a$.

9. Закони узагальненого склеювання:

9.1. $a \cdot b + \overline{a}c + bc = a \cdot b + a \cdot c$.

9.2. $(a+b) \cdot (\overline{a}+c) \cdot (b+c) = (a+b) \cdot (\overline{a}+c)$.

10. $(a+b) \cdot (\overline{a}+c) = a \cdot c + \overline{a}b$.

11. Закони де Моргана (закони інверсії) для двох змінних:

$\overline{a \cdot b} = \overline{a} + \overline{b}$; $\overline{a+b} = \overline{a} \cdot \overline{b}$.

Для *n* змінних

$\overline{a \cdot b \cdot c \cdot .. \cdot w} = \overline{a} + \overline{b} + \overline{c} + ... + \overline{w}$, $\overline{a + b + c + ...w} = \overline{a} \cdot \overline{b} \cdot \overline{c} \cdot ... \cdot \overline{w}$.

Двійковою функцією $y = f(x_1, x_2, .., x_n)$ (або булевою функцією, на честь математика Буля) називається функція, яка, як і її аргументи (двійкові), може приймати тільки два значення – 0 або 1. Булева функція виражає залежність вихідних змінних від вхідних змінних. Ці функції в залежності від числа вхідних змінних поділяються на функції однієї змінної, двох змінних і багатьох змінних.

Різні комбінації значень вхідних змінних у булевих функціях називаються наборами. Функція є повністю заданою, якщо вказані її значення для всіх наборів.

Наприклад, три вхідні змінні, які приймають значення 0 або 1, можуть дати лише вісім різних сполучень нульових і одиничних значень, тобто вісім наборів. Зіставляючи кожному набору значення функції, що дорівнює 0 або 1, можна отримати табличне завдання цієї функції (табл. 4.1). Така таблиця називається таблицею істинності або таблицею відповідності.

Таблиця 4.1 – Приклад таблиці істинності

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

За такого задавання набори завжди природно впорядковані, це дозволяє визначати функцію тільки останнім стовпцем (який іноді для економії місця записується в рядок). Наприклад, у нашому прикладі функцію $f(x_1, x_2, x_3)$ можна задати так: $f(x_1, x_2, x_3) = (10101110)$.

Такий спосіб задавання булевої функції називається векторним (першим). Кількість змінних і самі набори однозначно відновлюються за кількістю значень функції. Представлення функцій можна записати у шістнадцятковій системі числення: $f(x_1, x_2, x_3) = (10101110) = (AE_{16})$.

Булеву функцію також можна задати перерахуванням номерів своїх наборів, на яких вона приймає значення «істина» (true) або «хибність» (false). Нумерація починається з нуля: 0, 1, 2,…, : $f(x_1, x_2, x_3) = (0, 2, 4, 5, 6)$.

Крім перерахованих вище, використовуються такі способи завдавання булевих функцій: словесний, аналітичний, графічний, схемотехнічний.

За словесного способу значення функції залежно від її аргументів описуються виразом на природній мові. Наприклад, «Функція від трьох змінних дорівнює одиниці (істинна), якщо хоча б будь-які дві змінні мають значення 1».

За аналітичного способу нова функція задається суперпозицією вже відомих булевих функцій:

$$f(x_1, x_2, x_3) = (x_1 + x_2 + \bar{x}_3)(x_1 + \bar{x}_2 + \bar{x}_3)(\bar{x}_1 + \bar{x}_2 + \bar{x}_3).$$

У комп’ютерній техніці найважливішу роль відіграє функція однієї змінної: «заперечення» (функція НІ, NOT), і три функції двох змінних: «кон’юнкція» (функція І, AND), «диз'юнкція» (функція АБО, OR) і «нееквівалентність» (виключаюче АБО, XOR, додавання по модулю 2).

Функція однієї змінної – заперечення (інверсія) змінює двійкове значення свого аргументу на протилежне.

Функція І (кон'юнкція) приймає значення 1, якщо всі аргументи мають одиничне значення $f(x,y)=x\cdot y=xy=x\,\&\,y=(0,0,0,1)$.

Функція АБО (диз'юнкція) приймає значення 1, якщо хоча б один з аргументів має одиничне значення $f(x,y)=x+y=x\vee y=(0,1,1,1)$.

Функція «нееквівалентність» приймає значення 1, коли або змінна *x*, або змінна *y* має значення 1 (але не обидві разом)

$f(x,y)=x\otimes y=(0,\,1,\,1,\,0)$.

Функціям І, АБО, НІ притаманні чудові властивості: з них можна сконструювати будь-яку булеву функцію. Зокрема $x\otimes y=x\bar{y}+\bar{x}y$.

Розглянемо питання аналітичного представлення булевої функції за її таблицею істинності. Для наборів таблиці істинності вводяться поняття «конституент одиниці» і «конституент нуля».

Конституент одиниці для деякого набору – це елементарна кон'юнкція, у яку входять всі змінні, до того ж кожна змінна в нього входить без інверсії, якщо значення цієї змінної в цьому наборі дорівнює одиниці, або з інверсією, якщо значення цієї змінної в цьому наборі дорівнює нулю. Наприклад, набору 00111 відповідає конституент одиниці $\bar{a}\cdot\bar{b}\cdot c\cdot d\cdot e$. Звідси випливає: кожен конституент одиниці як функція змінних дорівнює одиниці тільки на одному відповідному йому наборі. На всіх інших наборах цей конституент одиниці дорівнює нулю.

Конституент нуля для деякого набору значень – це елементарна диз'юнкція, у яку входять всі змінні, до того ж кожна змінна в нього входить без інверсії, якщо значення цієї змінної в цьому наборі дорівнює нулю, або з інверсією, якщо значення цієї змінної в даному наборі дорівнює одиниці. Наприклад, набору 00111 відповідає конституент нуля $a+b+\bar{c}+\bar{d}+\bar{e}$. Звідси випливає: кожен конституент нуля, як функція *n* змінних дорівнює нулю тільки на одному відповідному йому наборі. На всіх інших наборах цей конституент нуля дорівнює одиниці.

Використовуючи поняття «конституент» одиниці і «конституент» нуля по таблиці істинності можна записати аналітичний вираз булевої функції в досконалої диз'юнктивної нормальною формою (ДДНФ) або в досконалої конюнктівної нормальною формою (ДКНФ).

ДДНФ цієї булевої функції називається «диз'юнкція конституентів одиниці» тих наборів значень змінних, де ця функція дорівнює одиниці. З таблиці 4.1 випливає, що

$$f(x_1,x_2,x_3)=\bar{x}_1\cdot\bar{x}_2\cdot\bar{x}_3+\bar{x}_1\cdot x_2\cdot\bar{x}_3+x_1\cdot\bar{x}_2\cdot\bar{x}_3+x_1\cdot\bar{x}_2\cdot x_3+x_1\cdot x_2\cdot\bar{x}_3.$$

ДКНФ цієї булевої функції називається «кон'юнкція конституентів нуля» тих наборів, де дана функція дорівнює нулю. З таблиці 4.1 випливає, що $f(x_1, x_2, x_3) = (x_1 + x_2 + \bar{x}_3) \cdot (x_1 + \bar{x}_2 + \bar{x}_3) \cdot (\bar{x}_1 + \bar{x}_2 + \bar{x}_3)$

Обидва ці вирази еквівалентні один одному. Використовуючи теореми для двійкових змінних, можна спростити вираз для булевої функції. У теорії булевих функцій для цього використовується ряд методів. Одним із них є використання карт Карно.

4.1.2 Застосування карт Карно для мінімізації булевих функцій

Одним з ефективних способів мінімізації логічних функцій при числі змінних не більше шести є використання карт Карно.

4.1.2.1 *Поняття про карту Карно*

Карта Карно становить собою спеціально організовану таблицю істинності, на якої зручно здійснюються операції склеювання при спрощення функції на шляху до мінімальних формам. Вона містить 2^n клітин (квадратів), розташованих у вигляді двовимірної матриці. Кожна клітина, як і рядок у таблиці істинності, відповідає одному набору. Стовпці та рядки таблиці відповідають всіляким наборам значень змінних, крім того ці набори розташовані в такому порядку, що кожний наступний відрізняється від попереднього тільки однією з змінних. Завдяки цьому сусідні клітинки по горизонталі і вертикалі відрізняються значенням лише однієї змінної. Клітинки, розташовані по краях таблиці, також вважаються сусідніми і володіють цією властивістю.

У таблиці 4.2 показані різноманітні набори булевої функції з двома змінними

Таблиця 4.2 – набори булевої функції

a	*b*
0	0
0	1
1	0
1	1

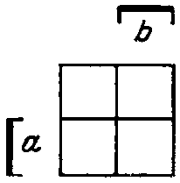

Рисунок 4.1 – Карта Карно на дві змінні

Їй відповідає карта Карно без заповнених в клітинах значень булевої функції (рис. 4.1).

Вхідні змінні розташовуються по зовнішнім бокам карти напроти її рядків і стовпців. За цього значення кожної з вхідних змінних відноситься до всього рядку (або стовпці) і дорівнює 1, якщо навпроти рядка (або стовпця) стоїть під дужкою позначення цієї змінної; для інших рядків (стовпців) значення цієї змінної дорівнює 0. Ці значення вхідних змінних не пишуться на карті, а маються на увазі. Кожному набору ставиться у відповідність клітина карти Карно. У цю клітку записується значення функції (0 або 1) для цього набору, як на рисунку 4.2, де зображена карта Карно для булевої функції трьох змінних $y = f(x, y, z)$

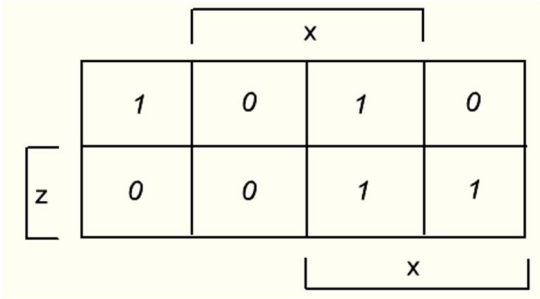

Рисунок 4.2 – Карта Карно на три змінні

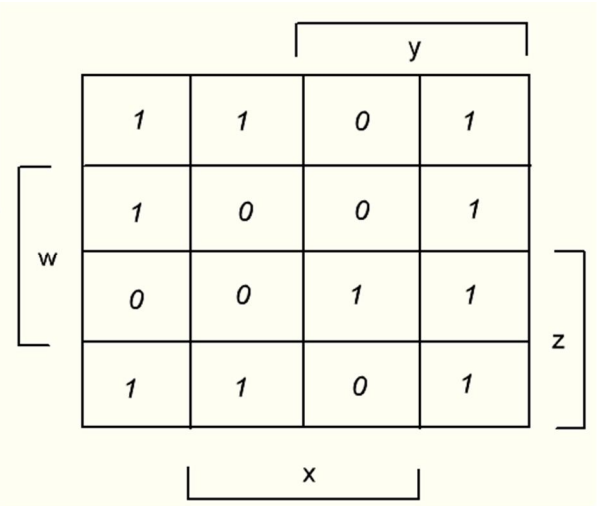

Рисунок 4.3 – Карта Карно на чотири змінних

Варто зазначити, що кожна з вхідних змінних поділяє карту Карно на дві рівні частини, в одній з яких значення цієї змінної дорівнює 1, а в інший 0. Кожній клітці карти відповідає один певний набір, а кожний бік клітини становить кордон між значеннями змінних. На рисунку 4.3 показана карта Карно для чотирьох змінних. Карти, зображені на рисунках 4.2 і 4.3 можуть слугувати еталонними заготовками.

4.1.2.2 *Визначення по карті Карно алгебраїчних виразів булевих функцій*

Для деякої булевої функції, представленої за допомогою карти Карно, можна записати декілька алгебраїчних виразів різної складності в диз'юнктивній або кон'юктивній формі. Для цього використовують прямокутні умовні контуру. Будується контур по одиницях у такий спосіб:

– сусідні дві, чотири, або вісім одиниць обводять загальним контуром;

– контур має бути прямокутним без вигинів або нахилів;

– кожен контур перетворює всі вхідні в нього одиниці в одну і для неї записується елементарна кон'юнкція за таким правилом: ті вхідні змінні, які

входять у координати даного контуру без інверсій, включаються в підсумкову кон'юнкцію безпосередньо, а ті змінні, які входять в координати цього контуру спільно зі своїми інверсіями, виключаються з підсумкової кон'юнкції (рис. 4.4).

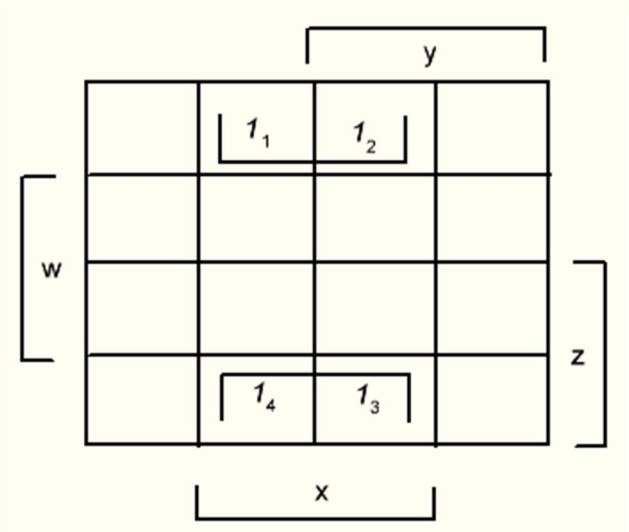

Рисунок 4.4 – Прямокутний контур із розривом у карті Карно

ДДНФ для виділеного контуру має вигляд:

$$F = x\bar{y}\bar{z}\bar{w} + xy\bar{z}\bar{w} + xyz\bar{w} + x\bar{y}z\bar{w}\,.$$

Підсумковий вираз, відповідний такому контуру, має вигляд $F = x\bar{w}$ Цей самий результат можна отримати аналітично, використовуючи операцію склеювання

$$F = x\bar{y}\bar{z}\bar{w} + xy\bar{z}\bar{w} + xyz\bar{w} + x\bar{y}z\bar{w} = x\bar{z}\bar{w}\left(\bar{y} + y\right) + xz\bar{w}\left(y + \bar{y}\right) = x\bar{z}\bar{w}\cdot 1 + xz\bar{w}\cdot 1 =$$
$$= x\bar{w}\left(\bar{z} + z\right) = x\bar{w}\cdot 1 = x\bar{w}.$$

Аналогічно будується контур по сусідніх нулях, але водночас кожен контур перетворює всі вхідні в нього нулі в один і для нього записується елементарна диз'юнкція.

Наведемо правила, якими варто керуватися під час запису виразу для булевої функції:

1. Усі одиниці (при записі функції в диз'юнктивній формі) або всі нулі (при записі функцій в кон'юнктивній формі) мають бути укладені в прямокутні контури.

Поодинокі контури можуть об'єднувати кілька одиниць, але не повинні містити в собі нулів. Нульові контури можуть об'єднувати кілька нулів, але не повинні містити в собі одиниць. Однойменні контури можуть накладатися один на одного, тобто одна і та ж одиниця (або нуль) може входити в кілька одиничних (нульових) контурів.

2. Площа будь-якого контуру має бути симетричною щодо границь змінних, що перетинаються цим контуром. Іншими словами, число клітин в контурі має дорівнювати 2^n, де $n = 0, 1, 2, 3, 4...$, тобто число клітин виражається числами 1, 2, 4, 8, 16, 32 ...

3. Щоб уникнути отримання зайвих контурів, всі клітини яких увійшли вже в інші контури, побудова слід починати з тих одиниць або нулів, які можуть увійти в один єдиний контур.

4. У контури можна об'єднувати тільки сусідні клітини, що містять одиниці або нулі. Дотримання цього правила особливо необхідно перевіряти за числа змінних, більшому чотирьох, коли сусідні клітини можуть бути розташовані не поряд і тому контури можуть зазнавати видимий розрив.

5. Кожній одиничній клітині відповідає кон'юнкція вхідних змінних, що визначають певну клітку. Кожній нульовій клітці відповідає диз'юнкція інверсій вхідних змінних, що визначають цю клітку.

6. Якщо у контурі, який об'єднує дві клітини, одна зі змінних змінює своє значення, то вираз для контуру з двох клітин не залежить від цієї змінної, а представляється всіма іншими змінними. Це правило відноситься і до контурів, що охоплює число клітин більше двох, і має таке формулювання: вираження, відповідні контурам, не містять тих змінних, чиї кордони перетинаються площею, обмеженою цим контуром.

7. Вираз булевої функції може бути записано по карті Карно в диз'юнктивній або кон'юнктивній формах.

Диз'юнктивна форма складається у вигляді диз'юнкції кон'юнкцій, що відповідають одиничним контурам, побудованих на карті Карно; кон'юнктивна – у вигляді кон'юнкції диз'юнкцій, відповідних нульових контурів.

8. Для контурів, що охоплюють різну кількість клітин, виходять вирази різної складності. Тому для цієї булевої функції можна записати по її карті Карно кілька різних за складністю алгебраїчних виразів. Найбільш складний вираз відповідає випадку, коли кожній клітині відповідає свій контур. Цей вираз є ДДНФ або ДКНФ цієї функції. Зі збільшенням розмірів контурів алгебраїчні вирази спрощуються. Найпростіший вираз функції виходить при утворенні найбільших контурів. На цій властивості грунтується метод мінімізації логічних функцій за допомогою карт Карно.

Приклад. Визначити алгебраїчний вираз булевої функції по карті Карно, наведеної на рис. 4.5.

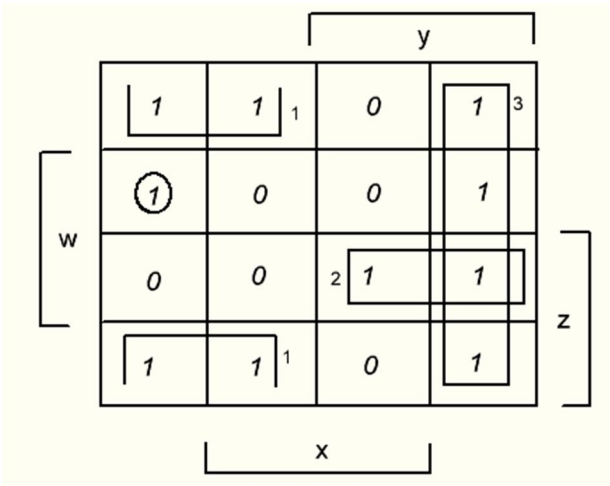

Рисунок 4.5 – Приклад карти Карно

Будемо визначати вираз функції по її одиничним значенням. Можна по одиницях скласти три контури, які відповідають умовам 1, 2, 4. Два з них суцільні, а третій з розривом. Відповідно до правил 5, 6 їм відповідають такі вирази: $\overline{y} \cdot \overline{w},\ y \cdot z \cdot w,\ \overline{x} \cdot y$. Клітці, яка не увійшла ні в один із контурів (обведена кружком), відповідає конституент одиниці – $\overline{x} \cdot \overline{y} \cdot \overline{z} \cdot w$.

Застосовуючи правило 7, можна записати для заданої булевої функції алгебраїчний вираз у диз'юнктивній формі:

$$f = \overline{y} \cdot \overline{w} + y \cdot z \cdot w + \overline{x} \cdot y + \overline{x} \cdot \overline{y} \cdot \overline{z} \cdot w.$$

4.1.2.3 *Використання булевих функцій в цифровій апаратурі*

Практично будь-яка інформація в цифрових обчислювальних системах зберігається і обробляється в двійкових кодах, окремі елементи, які приймають логічне значення 0 або 1. Тому цифрова апаратура проєктується на основі булевої алгебри з використанням логічних функцій. Будь-яку булеву функцію можна реалізувати на основі найпростіших функцій І, НІ та АБО, разом із тим одну і ту ж булеву функцію можна реалізувати по-різному. Булеві функції реалізуються за допомогою дискретних елементів, зазвичай діодів і транзисторів. Наприклад на рисунку 4.6 показана схема на двох діодах, що реалізує булеву функцію $Q = A \cdot B$.

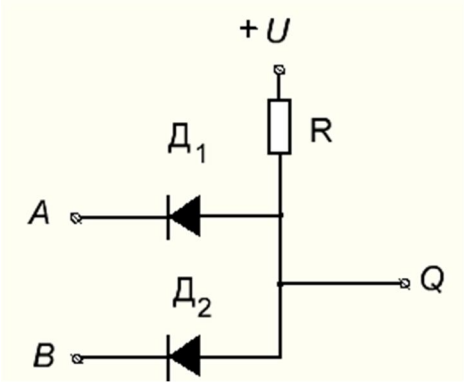

Рисунок 4.6 – Реалізація булевої функції на діодах

Якщо на якомусь вході *A* чи *B* діє низький рівень (двійковий 0), то він буде діяти і на виході *Q*. Високий рівень на виході *Q* (двійкова 1) виникає лише в тому випадку, якщо він буде присутній і на вході *A,* і на вході *B*.

На логічних схемах функції І, АБО і НЕ зображуються наступним так (рис. 4.7):

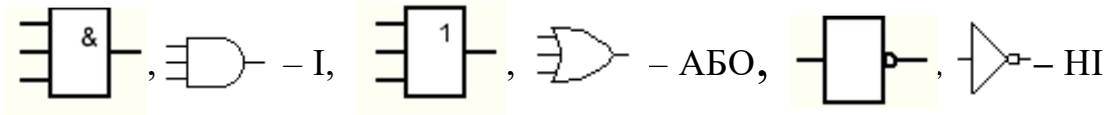

Рисунок 4.7 – Схематичне зображення елементарних функцій

Доцільно, щоб число дискретних елементів, що реалізують ту чи іншу булеву функцію, було мінімальним, тобто перед електронної реалізацією необхідно виконати мінімізацію булевої функції.

2 Завдання на практичне заняття

Булеву функцію, задану в першій векторній формі в шістнадцятковій системі числення, реалізувати апаратно в мінімальної формі, виконавши такі кроки:

А. Функцію перетворити в двійкове подання.

Б. Сформувати для функції таблицю істинності. Заготівля для таблиці істинності має вигляд (табл. 4.3)

Таблиця 4.3 – Заготівля для таблиці істинності

x	y	z	w	f(x,y,z,w)
0	0	0	0	f_0
0	0	0	1	f_1
0	0	1	1	f_3
…	…	…	…	…
1	1	1	1	f_{15}

В. Записати вираз для функції в ДДНФ або в ДКНФ.

Г. Сформувати по таблиці істинності карту Карно. Заготівля карти Карно має вигляд (рис. 4.8) :

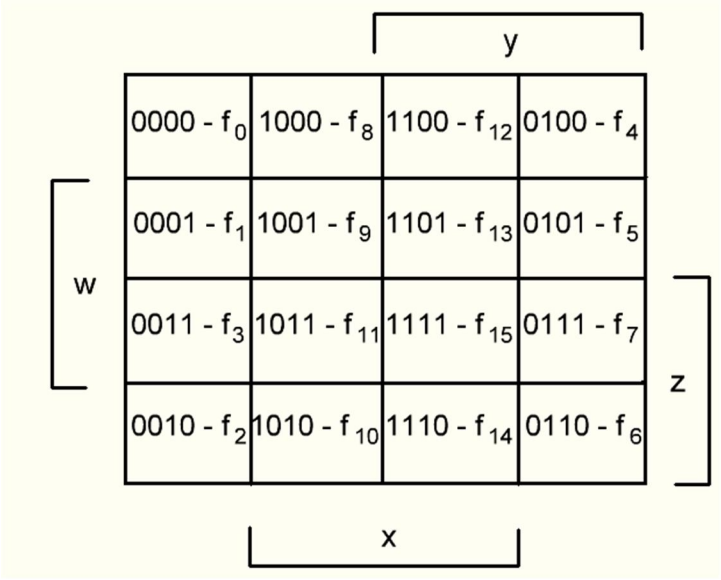

Рисунок 4.8 – Заготівля для карти Карно

Д. Використовуючи карту Карно, записати вираз для функції $f(x,y,z,w)$ у мінімальній формі.

Е. Скласти логічну схему функції, використовуючи елементи І, АБО, НІ.

Ж. Скласти електричну схему функції, використовуючи інтегральні мікросхеми.

Варіанти задачі задані у таблиці 4.4.

Таблиця 4.4 – Варіанти до практичного заняття

Варіант	$f(x,y,z,w)$	Варіант	$f(x,y,z,w)$	Варіант	$f(x,y,z,w)$	Варіант	$f(x,y,z,w)$
1	**3B44**$_{16}$	9	**628B**$_{16}$	17	**4747**$_{16}$	25	**1210**$_{16}$
2	**34B4**$_{16}$	10	**58B1**$_{16}$	18	**0064**$_{16}$	26	**2F50**$_{16}$
3	**2F50**$_{16}$	11	**52B1**$_{16}$	19	**4F44**$_{16}$	27	**C8C9**$_{16}$
4	**3329**$_{16}$	12	**CCC1**$_{16}$	20	**F111**$_{16}$	28	**2F50**$_{16}$
5	**15F4**$_{16}$	13	**5D0E**$_{16}$	21	**58B1**$_{16}$	29	**1D1D**$_{16}$
6	**C8C9**$_{16}$	14	**2323**$_{16}$	22	**0F05**$_{16}$	30	**3131**$_{16}$
7	**2F50**$_{16}$	15	**5300**$_{16}$	23	**628B**$_{16}$	31	**5B01**$_{16}$
8	**C8C9**$_{16}$	16	**3434**$_{16}$	24	**02A2**$_{16}$	32	**888F**$_{16}$

3 Приклад виконання завдання

Булеву функцію $f(x,y,z,w) = 5B01_{16}$, задану в першій векторній формі в шістнадцятковій системі числення, реалізувати апаратно в мінімальної формі.

А. Перетворимо число $5B01_{16}$ в двійкову систему числення:

$5B01_{16} = 0101101100000001_2$

Б. Сформуємо для функції $f(x,y,z,w)=5B01_{16}$ таблицю істинності

x	y	z	w	$f(x,y,z,w)$	x	y	z	w	$f(x,y,z,w)$
0	0	0	0	0	1	0	0	0	0
0	0	0	1	1	1	0	0	1	0
0	0	1	0	0	1	0	1	0	0
0	0	1	1	1	1	0	1	1	0
0	1	0	0	1	1	1	0	0	0
0	1	0	1	0	1	1	0	1	0
0	1	1	0	1	1	1	1	0	0
0	1	1	1	1	1	1	1	1	1

В. Оскільки одиничних значень функції менше, ніж нульових, запишемо аналітичний вираз для функції в досконалої диз'юнктивній нормальній формі

$$f(x,y,z,w)=\bar{x}\cdot\bar{y}\cdot\bar{z}\cdot w+\bar{x}\cdot\bar{y}\cdot z\cdot w+\bar{x}\cdot y\cdot\bar{z}\cdot\bar{w}+\bar{x}\cdot y\cdot z\cdot\bar{w}+\bar{x}\cdot y\cdot z\cdot w+$$
$$+x\cdot y\cdot z\cdot w$$

Г. Формуємо по таблиці істинності карту Карно (рис. 4.9). Оскільки невизначених клітинок немає, то заповнюємо лише одиничні клітини. Замість прямокутних контурів можна використовувати опуклі овальні.

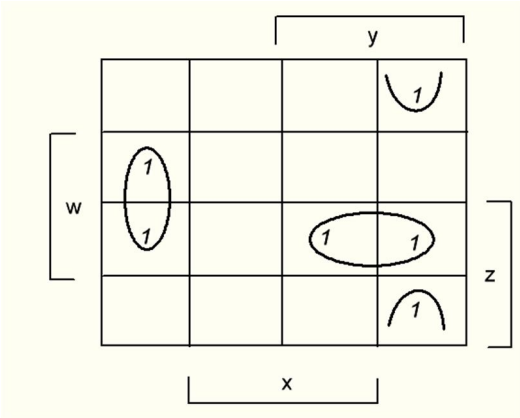

Рисунок 4.9 – Карта Карно для приклада виконання завдання

Д. Укладаємо всі одиниці в прямокутні контуру (їх три). Для кожного контуру складаємо кон'юнкцію змінних і об'єднуємо їх за допомогою диз'юнкції. Отримуємо вираз для функції $f(x,y,z,w)$ у мінімальній формі:

$$f(x,y,z,w)=\bar{x}\cdot\bar{y}\cdot w+y\cdot z\cdot w+\bar{x}\cdot y\cdot\bar{w}$$

Е. Будуємо на елементах І, АБО, НІ для системи Electronics Workbench схему, що реалізує функцію $f(x,y,z,w)$ (рис. 4.10). Схема реалізована в додатній логіці. Одиничним значенням вважається подача напруги в 3,3 В, нульовим – відсутність напруги. До виходу схеми під'єднаний елемент «лампа», який загоряється, коли на його вхід подадуть «одиницю» (3,3 В).

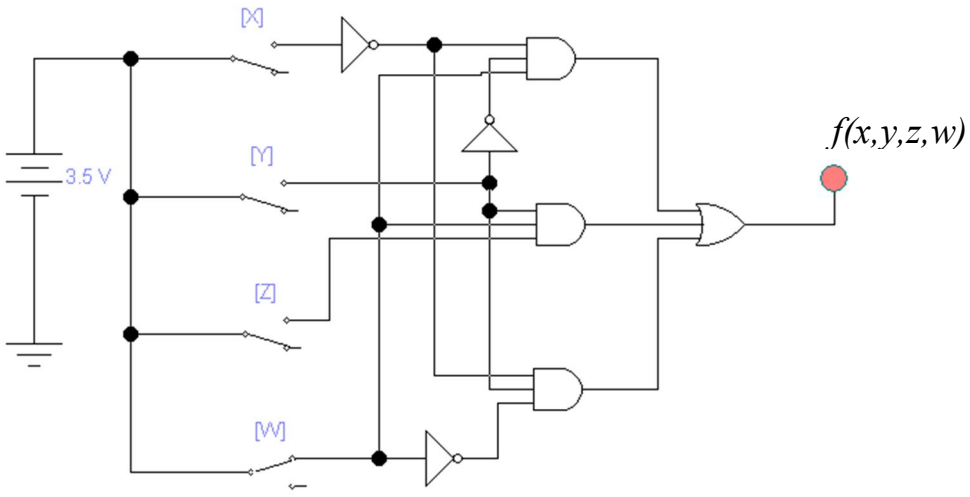

Рисунок 4.10 – Логічна схема, яка реалізує функцію $f(x,y,z,w)$

Змінні x, y, z, імітуються відповідними перемикачами. Лампа на виході загоряється, коли функція набуде значення «одиниця».

Контрольні запитання

1. Яка змінна називається двійковою?
2. Які синоніми використовуються для двійковій змінної?
3. Як у цифровий апаратурі реалізується двійкова змінна?
4. Назвіть основні аксіоми і теореми булевої алгебри
6. Сформулюйте закони де Моргана
8. Які функції називаються булеві і як їх задають?
10. Назвіть найбільш важливі булеві функції
11. Опишіть структуру таблиці істинності
12. Що таке конституент одиниці?
13. Що таке конституент нуля?
14. Як скласти алгебраїчний вираз для булевої функції за її таблицею істинності?
15. Що таке карта Карно і як її скласти?
17. Які властивості має карта Карно?
18. Що таке прямокутний контур у карті Карно?
19. Що відповідає прямокутному контуру карти Карно?
20. Як скласти алгебраїчний вираз у мінімальній формі за картою Карно?
21. Як реалізуються булеві функції в цифровий апаратурі?
23. Який порядок апаратної реалізації булевих функцій?

5 ВИКОРИСТАННЯ БУЛЕВИХ ФУНКЦІЙ ДЛЯ ШИФРУВАННЯ ТЕКСТУ

Мета заняття: освоєння технології кодування текстової інформації в коді ASCII, набуття навичок сумування по модулю два і освоєння шифрування тексту шляхом гамування

5.1 Теоретична частина

5.1.1 Основні поняття криптографії

Проблемою захисту інформації шляхом її перетворення займається криптологія (kryptos – таємний, logos – наука). Криптологія розділяється на два напрямки – криптографію і криптоаналіз [6, с. 10]. Мета цих напрямків прямо протилежна.

Криптографія займається пошуком і дослідженням математичних методів і алгоритмів перетворення інформації з метою її захисту від незаконних користувачів. Закінчена система методів і алгоритмів перетворення інформації з метою її захисту від незаконних користувачів називається криптосистемою (шифром). Сам процес перетворення інформації з метою її захисту від незаконних користувачів називається шифруванням. Змінний елемент шифру називається ключем. Ключі позбавляють користувача від розробки власних криптосистем при шифруванні інформації (що є практично неможливим). Зворотне перетворення інформації при відомому ключі називається розшифруванням.

Сфера інтересів криптоаналізу – дослідження можливостей дізнатися вміст шифрованого інформації (дешифрування) без знання ключів.

Сучасна криптографія охоплює чотири великих розділи:

1. Симетричні криптосистеми.
2. Криптосистеми з відкритим ключем.
3. Системи електронного підпису.
4. Управління ключами.

Основні напрямки використання криптографічних методів – передача конфіденційної інформації по каналах зв’язку (наприклад, електронна пошта), встановлення аутентичності переданих повідомлень, зберігання інформації (документів, баз даних) на носіях у зашифрованому вигляді.

Як інформація, що підлягає шифруванню і розшифруванню, будуть розглядатися тексти, побудовані на деякому алфавіті. Ці терміни трактуються так:

Алфавіт - кінцева множина знаків, використовуваних для кодування інформації.

Текст – упорядкований набір з елементів алфавіту.

Як приклади алфавітів, що використовуються в сучасних інформаційних системах, можна навести такі:

– алфавіт Z33 – 32 літери російського алфавіту і пробіл;

– алфавіт Z256 – символи, що входять в стандартні коди ASCII і КОИ-8;

– бінарний алфавіт – Z2 = {0,1};

– восімковий алфавіт або шістнадцятковий алфавіт.

Шифрування – перетворюючий процес: вихідний текст, який носить також назву відкритого тексту, замінюється шифрованим текстом (рис. 5.1).

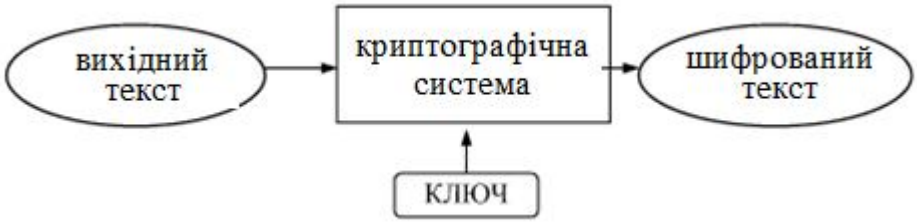

Рисунок 5.1 – Схема шифрування вихідного тексту

Розшифрування – зворотний процес шифруванню. На основі ключа шифрований текст перетвориться у вихідний (рис. 5.2).

Рисунок 5.2 – Схема розшифрування вихідного тексту

Тепер можна уточнити поняття ключа. Ключ – змінний інформаційний елемент криптосистеми, за допомогою якого здійснюється безперешкодне шифрування і розшифрування текстів.

Криптосистеми поділяються на симетричні і з відкритим ключем.

У симетричних криптосистемах і для шифрування, і для розшифрування використовується один і той же ключ.

У системах з відкритим ключем використовуються два ключі – відкритий і закритий, які математично пов’язані один з одним. Інформація шифрується за допомогою відкритого ключа, що доступний усім бажаючим, а

розшифровується за допомогою закритого ключа, відомого тільки одержувачу.

5.1.2 Кодування текстової інформації

Кодування текстової інформації це процес перетворення символів текстового повідомлення з форми, зручної для безпосереднього використання, у форму, зручну для передачі, зберігання або автоматичної переробки. Код символу – це певне число, зіставлене конкретному символу.

Систем кодувань безліч. У телекомунікаційних системах історично першою була азбука Морзе. У комп'ютерних системах першим стандартом кодування символів текстового повідомлення був ASCII (American standard code for information interchange).

За цим стандартом кожен символ займає 1 байт, всього є 256 символів. Коди символів представлені в спеціальній таблиці. Кожен символ має свій внутрішній код, який співпадає з порядковим номером символа у таблиці.

Таблиця складається з трьох частин.

Символи з номерами від 0 до 31 є керуючими. З їх допомогою здійснюється управління процесом виведення тексту на екран або друк, подача звукового сигналу, розмітка тексту і т.п.

Стандартна частина таблиці містить символи з номерами від 32 до 127. Це малі та великі літери латинського алфавіту, десяткові цифри, розділові знаки, всілякі дужки, комерційні та інші символи. Символ 32 – пробіл, тобто порожня позиція в тексті. Усі інші відображаються певними знаками.

У третій частині таблиці ASCII, яка містить символи з номерами з 128 до 255, знаходяться літери національних алфавітів та додаткові символи. Ця частина таблиці різна для різних країн та різних кодових таблиць, встановлених в операційній системі.

Для країн СНД найчастіше використовується таблиця кодів ASCII, яка наведена в таблиці 5.1.

Таблиця 5.1 – Кодування символів ASCII

Символ	Вісімкове кодування	Десяткове кодування	Примітка	Символ	Вісімкове кодування	Десяткове кодування	Символ	Вісімкове кодування	Десяткове кодування	Символ	Вісімкове кодування	Десяткове кодування
NUL	0	0		@	100	64	А	200	128	└	300	192
SOH	1	1	Початок заголовка	A	101	65	Б	201	129	┴	301	193
STX	2	2	Початок текста	B	102	66	В	202	130	┬	302	194
ETX	3	3	Кінець текста	C	103	67	Г	203	131	├	303	195

Продовження таблиці 5.1

Символ	Вісімкове кодування	Десяткове кодування	Примітка	Символ	Вісімкове кодування	Десяткове кодування	Символ	Вісімкове кодування	Десяткове кодування	Символ	Вісімкове кодування	Десяткове кодування
EOT	4	4	Кінець передачі	D	104	68	Д	204	132	─	304	196
ENQ	5	5	Прошу підтвердження	E	105	69	Е	205	133	┼	305	197
ACK	6	6	Підтверджую	F	106	70	Ж	206	134	╞	306	198
BEL	7	7	Звуковой сигнал	G	107	71	З	207	135	╟	307	199
BS	10	8	Повернення на символ	H	110	72	И	210	136	╚	310	200
TAB	11	9	Горизонтальна табуляція	I	111	73	Й	211	137	╔	311	201
LF	12	10	Переклад рядка	J	112	74	К	212	138	╩	312	202
VT	13	11	Вертикальна табуляція	K	113	75	Л	213	139	╦	313	203
FF	14	12	Прогін сторінки	L	114	76	М	214	140	╠	314	204
CR	15	13	Повернення каретки	M	115	77	Н	215	141	═	315	205
SO	16	14	Початок використання національного кодування	N	116	78	О	216	142	╬	316	206
SI	17	15	Назад до SO	O	117	79	П	217	143	╧	317	207
DLE	20	16	Звільнення каналу даних	P	120	80	Р	220	144	╨	320	208
DC1	21	17	Включити пристрій 1	Q	121	81	С	221	145	╤	321	209
DC2	22	18	Включити пристрій 2	R	122	82	Т	222	146	╥	322	210
DC3	23	19	Включити пристрій 3	S	123	83	У	223	147	╙	323	211
DC4	24	20	Включити пристрій 4	T	124	84	Ф	224	148	╘	324	212
NAK	25	21	Не підтверджую	U	125	85	Х	225	149	╒	325	213
SYN	26	22	Синхронізація	V	126	86	Ц	226	150	╓	326	214
ETB	27	23	Кінець текстового блоку	W	127	87	Ч	227	151	╫	327	215
CAN	30	24	Скасуванняа	X	130	88	Ш	230	152	╪	330	216
EM	31	25	Кінець носія	Y	131	89	Щ	231	153	┘	331	217
SUB	32	26	Підставити	Z	132	90	Ъ	232	154	┌	332	218
ESC	33	27	Escape, скасувати	[	133	91	Ы	233	155	█	333	219
FS	34	28	Роздільник файлів	\	134	92	Ь	234	156	▄	334	220
GS	35	29	Роздільник груп	]	135	93	Э	235	157	▌	335	221
RS	36	30	Роздільник записів	^	136	94	Ю	236	158	▐	336	222
US	37	31	Роздільник юнітів	_	137	95	Я	237	159	▀	337	223
SP	40	32	Пропуск	`	140	96	а	240	160	р	340	224
!	41	33	Знак оклику	a	141	97	б	241	161	с	341	225
"	42	34	Лапки	b	142	98	в	242	162	т	342	226
#	43	35	Знак решётки	c	143	99	г	243	163	у	343	227
$	44	36	Долар	d	144	100	д	244	164	ф	344	228
%	45	37	Відсоток	e	145	101	е	245	165	х	345	229
&	46	38	Амперсанд	f	146	102	ж	246	166	ц	346	230

Закінчення таблиці 5.1

Символ	Вісімкове кодування	Десяткове кодування	Примітка	Символ	Вісімкове кодування	Десяткове кодування	Символ	Вісімкове кодування	Десяткове кодування	Символ	Вісімкове кодування	Десяткове кодування
'	47	39	Апостроф	g	147	103	з	247	167	ч	347	231
(	50	40		h	150	104	и	250	168	ш	350	232
)	51	41		i	151	105	й	251	169	щ	351	233
*	52	42		j	152	106	к	252	170	ъ	352	234
+	53	43		k	153	107	л	253	171	ы	353	235
,	54	44		l	154	108	м	254	172	ь	354	236
-	55	45		m	155	109	н	255	173	э	355	237
.	56	46		n	156	110	о	256	174	ю	356	238
/	57	47	Правий слеш	o	157	111	п	257	175	я	357	239
0	60	48		p	160	112	▒	260	176	Ё	360	240
1	61	49		q	161	113	▒	261	177	ё	361	241
2	62	50		r	162	114	▓	262	178	Є	362	242
3	63	51		s	163	115	│	263	179	є	363	243
4	64	52		t	164	116	┤	264	180	Ї	364	244
5	65	53		u	165	117	╡	265	181	ї	365	245
6	66	54		v	166	118	╢	266	182	Ў	366	246
7	67	55		w	167	119	╖	267	183	ў	367	247
8	70	56		x	170	120	╕	270	184	°	370	248
9	71	57		y	171	121	╣	271	185	∙	371	249
:	72	58		z	172	122	║	272	186	·	372	250
;	73	59		{	173	123	╗	273	187	√	373	251
<	74	60		‖	174	124	╝	274	188	№	374	252
=	75	61		}	175	125	╜	275	189	¤	375	253
>	76	62		~	176	126	╛	276	190	■	376	254
?	77	63		Δ	177	127	┐	277	191		377	255

Будь-який символ коду ASCII можна ввести в такий спосіб: утримуючи клавіатурну клавішу Alt, набрати на цифровому блоці десятковий код символу і відпустити клавішу Alt. Це корисно при введенні таких символів, наприклад, як ╣.

5.1.3 Криптосистеми на основі гамування

Метод гамування відноситься до симетричних криптосистемам і сутність цього методу полягає в тому, що символи тексту, який шифрується, послідовно складаються з символами деякої спеціальної послідовності, яка

називається гамою. Іноді такий метод подають як накладення гами на вихідний текст, тому він і отримав назву «гамування».

Процедура накладання гами на вихідний текст зазвичай здійснюється у такій спосіб: символи вихідного тексту і гамми представляються у вигляді двійкового коду. Для кодування символів тексту найчастіше використовують розширене кодування ASCII (табл. 5.1). Потім відповідні розряди кожного символа у двійковому коді складаються по модулю два з двійковими розрядами гами. Математично це записується в такий спосіб: $S = R \oplus G$, де R – черговий біт вихідного тексту, G – відповідний біт гами, S – біт шифротексту. Таблиця істинності для цієї булевої операції має вигляд:

R	G	S
0	0	0
0	1	1
1	0	1
1	1	0

Перевагою операції по модулю 2 є те, що знаючи S і G можна відновити вихідне значення за формулою $R = S \oplus G$. Дійсно:

S	G	R
0	0	0
1	1	0
0	1	1
1	0	1

Замість складання по модулю два при гамуванні можна використовувати й інші логічні операції, наприклад перетворення за правилом логічної еквівалентності $(S = R \sim G)$, яка має таку таблицю істинності

R	G	S
0	0	1
0	1	0
1	0	0
1	1	1

За операцією еквівалентності можна відновити вихідне значення R $(R = S \sim G)$. дійсно

S	G	R
1	0	0
0	1	0
0	0	1
1	1	1

Така заміна тотожня введенню ще одного ключа, яким є вибір правила формування символів зашифрованого повідомлення із символів вихідного тексту і гами. У таблиці 5.2 наведено приклад формування символів зашифрованого повідомлення із символів вихідного тексту і гамми (використовується додавання за модулем два).

Таблиця 5.2 – Процедура гамування

Текст, що шифрують	м	о	д	а
	10101100	10101110	10100100	10100000
Знаки гами	7	1	8	2
	00110111	00110001	00111000	00110010
Шифрований текст	10011011	10011111	10011100	10010010
	Ы	Я	Ь	Т

Стійкість шифрування методом гамування визначається переважно властивостями гами – тривалістю періоду і рівномірністю статистичних характеристик. Остання властивість забезпечує відсутність закономірностей у появі різних символів у межах періоду.

Зазвичай поділяють два різновиди гамування – із кінцевої і нескінченної гамами. При гарних статистичних властивостях гами якість шифрування визначається тільки довжиною періоду гами. Водночас, якщо довжина періоду гами перевищує довжину тексту, що шифрують, то такий шифр теоретично є абсолютно стійким, оскільки його не можна розкрити за допомогою статистичної обробки зашифрованого тексту. Це, однак, не означає, що дешифрування такого тексту взагалі неможливо: за наявності деякої додаткової інформації вихідний текст може бути частково або повністю відновлений навіть за використання нескінченної гами.

Як гама може бути використана будь-яка послідовність випадкових символів, наприклад послідовність цифр числа e = 2,718281828 .. (основа натурального логарифма), числа (π = 3,141592654 ..) і т. п. При шифруванні за допомогою ЕОМ послідовність гами може формуватися за допомогою датчика псевдовипадкових чисел. Наразі розроблено декілька алгоритмів роботи таких датчиків, які забезпечують задовільні характеристики гами.

5.2 Завдання на практичне заняття, порядок його виконання і оформлення

1. По таблиці варіантів (табл. 5.3) вибрати вихідний текст і зашифрувати його відповідної гамою. Порядок виконання наведено в п. 5.2. Для кодування символів використовувати код ASCII, наведений в таблиці 5.1.

Таблиця 5.3 – Варіанти виконання практичної роботи

1	Текст	Ученье-свет
	Гамма	biwmtrdospa
2	Текст	Неученье-тьма
	Гамма	шеутрылкцтнв
3	Текст	Дорогу осилит
	Гамма	etysgforkcysmr
4	Текст	Ударный батальон
	Гамма	ndijkecklswetyudhc
5	Текст	Звездное небо
	Гамма	hgfdioytrnhjkd
6	Текст	Сбор возле дуба
	Гамма	ldurhnwiofnslvre
7	Текст	Компьютеризация
	Гамма	jlrlsivmrkcjwkctxb
8	Текст	Индустриализация
	Гамма	bwjydluaofjkceudka
9	Текст	Коммуникация
	Гамма	ldjtwiocmdusbw
10	Текст	Диверсификация
	Гамма	lwhysjcimepaxznc
11	Текст	Батальоны вперед!
	Гамма	rdhjkxbdralivjnexks
12	Текст	Шашки обнажить!
	Гамма	poekudfhznceqjdbd
13	Текст	Терпенье и труд
	Гамма	wkaldjtvbsnxyase
14	Текст	Страна Мозамбик
	Гамма	ojdhrycbjkdyekm
15	Текст	Созвездие Псов
	Гамма	wmxycrenbkahyd
16	Текст	Экономическая ниша
	Гамма	jayqzxcgsnmfjrvbwkct
17	Текст	Благотворительство
	Гамма	vstyuwkyfuksohelkga
18	Текст	Кафедра радиофизики
	Гамма	ertbgdjkalisxcvbehtdjkd
19	Текст	Институт хирургии
	Гамма	xfkhgspyueknrychek
20	Текст	Инженер-педагог
	Гамма	cetajdlinekdyvpejq

2. Для отримання найвищого балу по практичній роботі необхідно скласти програму шифрування тексту на С++ і перевірити правильність виконання завдання.

3. Оформити письмово звіт по практичному заняттю і захистити його викладачеві. Звіт повинен містити титульний аркуш із відповідними атрибутами, вихідні дані, результат виконання практичного заняття, висновки.

Завдання виконується і оформлюється в такій послідовності.

1. На титульному аркуші вказати назву роботи:

Освоєння криптосистеми на основі «гамування»,

прізвище та ініціали виконавця, номер групи, номер варіанта і дату виконання.

2. Починаючи з другої сторінки, написати мету роботи, вихідні дані.

3. Скласти таблицю (табл. 5.4), куди спочатку занести символи вихідного тексту, їх восьмеричний код ASCII, символи гами, їх восьмеричний код ASCII.

Таблиця 5.4 – Заготовка для виконання завдання

Текст, що шифрується									
Вісімковий код тексту, що шифрується									
Двійковий код тексту, що шифрується									
Знаки гами									
Вісімковий код гами									
Двійковий код гами									
Двійковий код шифрованого тексту									
Вісімковий код шифрованого тексту									
Шифрований текст у символьному вигляді									

4. Вісімковий код ASCII перевести в двійковий код. Для цього кожну цифру вісімкового коду замінити тризначним двійковим еквівалентом. Наприклад 3 ~ 011.

5. Виконати підсумовування по модулю два кожного двійкового розряду двійкового коду кожного символу вихідного тексту з відповідним двійковим розрядом відповідного символу гами. Результат занести в сьомий рядок таблиці 5.4.

6. Замінити двійковий код кожного символу зашифрованого тексту вісімковим кодом. Для цього двійковий код умовно розбити справа наліво на тріади і кожну тріаду замінити відповідною вісімковою цифрою.

7. За таблицею кодувань ASCII зіставити кожному вісімковому числу відповідний символ і результат занести в останній рядок таблиці 5.4.

8. Під таблицею 5.4 виписати вихідний текст, а нижче текст в зашифрованому вигляді.

9. Сформулювати висновки по роботі і записати їх у звіт.

5.3 Приклад виконання завдання

Нехай необхідно зашифрувати текст «Знання» гамою knowle.

Заносимо посимвольно вихідний текст і гаму в осередки першої та четвертої рядків таблиці 5.4. Отримуємо таблицю 5.5.

Таблиця 5.5 – Перший крок виконання завдання

Текст, що шифрується	З	н	а	н	н	я
Вісімковий код тексту, що шифрується						
Двійковий код тексту, що шифрується						
Знаки гами	k	n	o	w	l	e
Вісімковий код гами						
Двійковий код гами						
Двійковий код шифрованого тексту						
Вісімковий код шифрованого тексту						
Шифрований текст у символьному вигляді						

За таблицею кодів ASCII знаходимо вісімкові коди символів вихідного тексту і гами і заносимо їх у відповідні комірки другого та четвертого рядків таблиці 5.5. Отримуємо таблицю 5.6.

Таблиця 5.6 – Другий крок виконання завдання

Текст, що шифрується	З	н	а	н	н	я	
Вісімковий код тексту, що шифрується	207	255	240	255	255	357	
Двійковий код тексту, що шифрується							
Знаки гами	k	n	o	w	l	e	
Вісімковий код гами	153	156	157	167	154	145	
Двійковий код гами							
Двійковий код шифрованого тексту							
Вісімковий код шифрованого тексту							
Шифрований текст у символьному вигляді							

Переводимо вісімковий код чисел вихідного тексту і гами в двійковий код. Отримаємо таблицю 5.7.

Таблиця 5.7 – Третій крок виконання завдання

Текст, що шифрується	З	н	а	н	н	я
Вісімковий код тексту, що шифрується	207	255	240	255	255	357
Двійковий код тексту, що шифрується	10000111	10101101	10100000	10101101	10101101	10101111
Знаки гами	k	n	o	w	l	e
Вісімковий код гами	153	156	157	167	154	145
Двійковий код гами	01101011	01101110	01101111	01110111	01101100	01100101
Двійковий код шифрованого тексту						
Вісімковий код шифрованого тексту						
Шифрований текст у символьному вигляді						

Здійснюємо порозрядне додавання по модулю два двійкових кодів символів вихідного тексту і гами. Результат заносимо в сьомий рядок таблиці 5.7. Отримуємо таблицю 5.8.

Таблиця 5.8 – Четвертий крок виконання завдання

Текст, що шифрується	З	н	а	н	н	я
Вісімковий код тексту, що шифрується	207	255	240	255	255	357
Двійковий код тексту, що шифрується	10000111	10101101	10100000	10101101	10101101	10101111
Знаки гами	k	n	o	w	l	e
Вісімковий код гами	153	156	157	167	154	145
Двійковий код гами	01101011	01101110	01101111	01110111	01101100	01100101
Двійковий код шифрованого тексту	11101100	11000011	11001111	11011010	11000001	10001010
Вісімковий код шифрованого тексту						
Шифрований текст у символьному вигляді						

Розбивши двійковий код кожного символу зашифрованого тексту справа наліво на тріади і замінивши кожну тріаду вісімковою цифрою, отримаємо вісімковий код зашифрованого тексту (табл. 5.9).

Таблиця 5.9 – П’ятий крок виконання завдання

Текст, що шифрується	З	н	а	н	н	я
Вісімковий код тексту, що шифрується	207	255	240	255	255	357
Двійковий код тексту, що шифрується	10000111	10101101	10100000	10101101	10101101	10101111
Знаки гами	k	n	o	w	l	e
Вісімковий код гами	153	156	157	167	154	145
Двійковий код гами	01101011	01101110	01101111	01110111	01101100	01100101
Двійковий код шифрованого тексту	11101100	11000011	11001111	11011010	11000001	10001010
Вісімковий код шифрованого тексту	354	303	317	332	301	212
Шифрований текст у символьному вигляді						

За таблицею кодів ASCII, наведеною в таблиці 5.1, знаходимо символи, відповідні восьмеричним кодам зашифрованого тексту, і заносимо їх в останній рядок табл. 5.9. Отримуємо підсумкову таблицю (табл. 5.10).

Таблиця 5.10 – Підсумкова таблиця завдання

Текст, що шифрується	З	н	а	н	н	я
Вісімковий код тексту, що шифрується	207	255	240	255	255	357
Двійковий код тексту, що шифрується	10000111	10101101	10100000	10101101	10101101	10101111
Знаки гами	k	n	o	w	l	e
Вісімковий код гами	153	156	157	167	154	145
Двійковий код гами	01101011	01101110	01101111	01110111	01101100	01100101
Двійковий код шифрованого тексту	11101100	11000011	11001111	11011010	11000001	10001010
Вісімковий код шифрованого тексту	354	303	317	332	301	212
Шифрований текст у символьному вигляді	ь	├	╧	┌	┴	К

Таким чином, вихідний і зашифрований тексти такі:

З н а н н я

ь ├ ╧ ┌ ┴ К

(Тут між символами для наочності додані проміжки).

Контрольні запитання і завдання

1. Назвіть базові булеві функції двох змінних
2. Яка особливість булевої функції «Додавання по модулю два» ?
3. Як кодуються текстові символи в комп’ютері ?
4. Що таке алфавіт? Назвіть алфавіти, які використовуються в інформаційних системах.
5. Що таке криптографія, криптосистема?
6. Що таке ключ шифру?
7. Поясніть терміни шифрування, дешифрування.
8. Які розділи містить криптографія? Подайте їхню коротку характеристику
9. На які види поділяються криптосистеми і в чому між ними різниця?
10. Опишіть криптосистему на основі методу гамування.
11. Порозрядно підсумуйте два числа по модулю два

111010001

010110011

Результат подайте в шістнадцятковій і десятковій системах числення.

6 ОПТИМАЛЬНЕ ПРОЄКТУВАННЯ СТРУКТУРИ ЛОКАЛЬНОЇ КОМП'ЮТЕРНОЇ МЕРЕЖІ

Мета заняття: вивчення завдання проєктування структури локальної комп'ютерної мережі (ЛКМ); освоєння технології побудови математичної моделі структури ЛКМ і вибір її оптимальної структури на ПЕОМ

6.1 Теоретична частина

Важлива частина проєктування ЛКМ системи – розроблення її інформаційної структури. Удосконалювання інформаційної структури полягає в раціоналізації форм носіїв інформації, скороченні кількості джерел інформації й усуненні дублювальних потоків інформації, зміні маршрутів руху інформації й алгоритмів її передачі. Подібне завдання виникає, зокрема, під час проєктуванні інформаційного забезпечення ЛКМ підприємством. Інформація в ЛКМ передається у вигляді повідомлень, кожне з яких є інформаційним відображенням деяких властивостей об'єкта або групи об'єктів. Джерелами інформації можуть бути різні засоби збору й реєстрації інформації, автоматизовані робочі місця користувачів. До споживачів інформації варто зарахувати різні підсистеми ЛКМ.

Для передачі інформації від джерел до споживачів використовуються технічні засоби, що містять у собі лінії зв'язку, модеми для перетворення переданої інформації в допоміжний сигнал, а потім прийнятого сигналу в зручну для видачі форму, пристрої стискання інформації й т.д. Найвідомішою серед подібних систем є мережа Інтернет. Технічні засоби, призначені для передачі інформації, зазвичай називають каналами зв'язку. Будемо вважати, що можливі структури організації руху інформації відомі. Потрібно вибрати таку підструктуру інформаційних потоків, що мінімізує довжину шляху проходження інформації від джерела до споживача. Отримана інформаційна структура системи може бути реалізована за допомогою технічних засобів збору, передачі й видачі інформації за мінімуму економічних витрат.

6.1.1 Модель дискретного програмування інформаційної структури ЛКМ, її алгоритмічна і програмна реалізація

Джерелам і споживачам інформації поставимо у відповідність вершини графа, а кожну пару вершин з'єднаємо дугою, що йде від вершини x_i до вершини x_j, якщо елемент x_i є входом елемента x_j. Вийде інформаційний граф, приклад якого подано на рисунку 6.1.

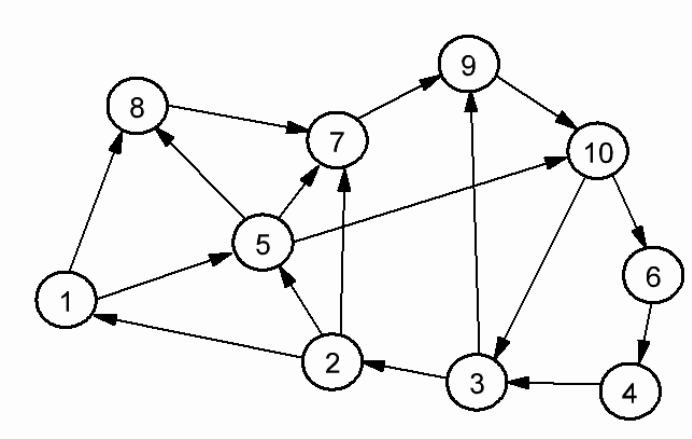

Рисунок 6.1 – Граф структури локальної комп'ютерної мережі

Кожна дуга графа має певну вагу c_{ij}, що характеризує довжину шляху, по якому проходить інформаційний потік між вершинами *i* й *j*. Найкоротшим називається шлях, у якому сума ваг дуг мінімальна порівняно з сумою будь-якого іншого шляху, що з'єднує розглянуті вершини графа. Потрібно з декількох шляхів, що ведуть від однієї вершини до іншої, вибрати однієї мінімальної довжини. У такий спосіб поставлена задача зводиться до дискретного програмування.

Математично вона записується в такий спосіб:

Мінімізувати $\sum_{(i,j)\in cemu}\sum c_{ij}x_{ij}$

при обмеженнях

$$\sum_{(k,j)\in cemu} x_{kj} - \sum_{(i,k)\in cemu} x_{ij} = \begin{cases} 1,\ k = s\ (\text{початковий вузол}) \\ 0,\ \text{для всіх інших k} \\ -1,\ k = r\ (\text{кінцевий вузол}) \end{cases},$$

де x_{ij} – булева змінна, що дорівнює одиниці, якщо дуга, що зв'язує вершини *i* й *j*, входить в оптимальний маршрут, і нулю в протилежному випадку.

Запропоновано кілька алгоритмів розв'язання цієї задачі. Один з них модифікований алгоритм Уоршелла [7, с. 198].

В алгоритмі використовуються дві вихідні квадратні матриці *X* й *V,* розміром $n \times n$, де *n* – число вершин графа, *X* – матриця ваг дуг, у якій кожній дузі, яка з'єднує вершини *i* і *j*, зіставляється елемент x_{ij}, що дорівнює довжині цієї дуги. Якщо вершини *i* й *j* прямо не зв'язані, то елемент x_{ij} дорівнює ∞. *V* – матриця суміжності, у якій кожній дузі зіставлене певне

ім’я. Після виходу з алгоритму будуть сформовані два масиви: X^* – масив найкоротших відстаней і V^* – масив найкоротших шляхів. Послідовність кроків алгоритму така:

1. Покласти $X^* = X$ й $V^* = V$.
2. Покласти $j = 1$.
3. Покласти $i = 1$.
4. Якщо $x_{ij}^* = \infty$, перейти до 8.
5. Покласти $k = 1$.
6. Якщо $(x_{ij}^* + x_{jk}^*) < x_{ik}^*$, то

   а) покласти $x_{ik}^* = x_{ij}^* + x_{jk}^*$,

   б) покласти $v_{ik}^* = v_{ij}^* \cdot v_{jk}^*$.
7. Покласти $k = k + 1$. Якщо $k \le n$, перейти до 6.
8. Покласти $i = i + 1$. Якщо $i \le n$, перейти до 4.
9. Покласти $j = j + 1$. Якщо $j \le n$, перейти до 3; інакше *кінець.*

Операція $v_{ik}^* = v_{ij}^* \cdot v_{jk}^*$ над елементами матриці V^* на кроці 6 є зчеплення.

Програмна реалізація цього алгоритму мовою програмування символьної системи комп'ютерної математики Maple при довільному *n* має вигляд:

```
>n:=**:
> m:=1000:
> for j from 1 to n do↵
    for i from 1 to n do↵
      if X[i,j]<m then ↵
        for k from 1 to n do↵
        if X[i,j]+X[j,k]<X[i,k] then X[i,k]:=X[i,j]+X[j,k]: ↵
                                     V[i,k]:=V[i,j]*V[j,k] ↵
       fi:                  od
      fi:                od
                       od:
>for s from 1 to n do V[s,s]:=`-`:X[s,s]:=`-` od:
> evalm(V);
> evalm(X);
```

X[i,j] – матриця, елементи якої спочатку містять значення дуг для безпосередньо зв’язаних вершин (кореспонденції), а на виході алгоритму – найкоротші відстані між вузлами мережі. *V[i,j]* – матриця, елементи якої

спочатку містять назву дуг для безпосередньо зв'язаних вершин, а на виході алгоритму – найменування кореспонденцій, через які проходить найкоротші відстані між вузлами мережі.

У цій реалізації замість нескінченності використовується значення, свідомо більше максимальної відстані, у цьому випадку 1000.

6.1.2 Метод потенціалів розрахунку найкоротших відстаней між вузлами мережі

Для розрахунку найкоротших відстаней між вузлами мережі ручним способом можна використати метод потенціалів. Під час його застосування будемо використовувати наступні позначення:

– P_i – абсолютний потенціал i-ї вершини відносно базової, тобто найкоротша відстань між базовою вершиною мережі, відносно якою обчислюються найкоротші шляхи і відстані, і вершиною з номером i;

– p_{ij} – відносний потенціал вершини i відносно вершини j, тобто найкоротша відстань між базовою вершиною мережі і вершиною з номером i, а шлях до вершини i, проходить через вершину j;

– l_{ij} – довжина дуги $[i, j]$.

Алгоритм методу потенціалів такий:

1. Вершині, від якої потрібно визначити найкоротші відстані до всіх інших, привласнюємо потенціал, що дорівнює 0.

2. Переглядаємо всі дуги, потенціал однієї з вершин яких (початкової вершини) визначений, а іншої (кінцевої) – не визначений. За формулою $p_{ij} = P_i + l_{ij}$ визначаємо відносні потенціали кінцевих вершин по відповідних дугах.

3. Із всіх потенціалів по дугах (знайдених на всіх кроках і залишених для розгляду) знаходимо найменший (або якогось з них, якщо таких декілька) і привласнюємо його як абсолютний потенціал відповідної кінцевої вершини. Відповідну дугу відзначаємо стрілкою, що веде в кінцеву вершину. Усі раніше обчислені потенціали по дугах, що ведуть у цю вершину, надалі не розглядаємо.

Переходимо до п.2.

Розрахунок продовжуємо до визначення потенціалів всіх вершин. Величина потенціалу вершини пункту показує найкоротшу відстань від обраного початкового пункту до потрібного пункту. Дуги зі стрілками утворять найкоротший маршрут руху від початкового пункту до всіх інших.

Приймаючи за початок мережі послідовно кожну її вершину, по описаному методі одержуємо таблицю найкоротших відстаней між всіма вершинами мережі.

Результати розрахунків зводимо в таблицю.

Приклад. Розрахувати найкоротші відстані між вузлами інформаційної мережі, заданої графом на рисунку 6.2.

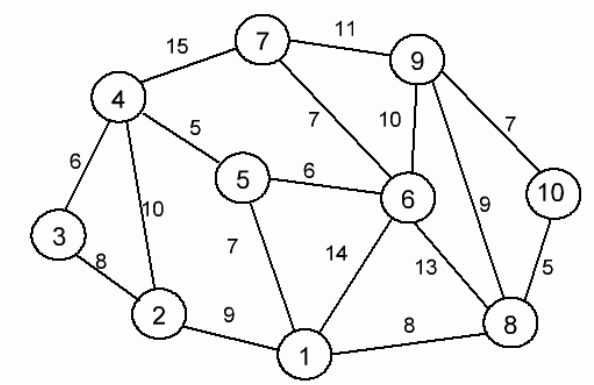

Рисунок 6.2 – Граф мережі до прикладу

Розрахуємо найкоротші відстані від вершини ***1*** до всіх інших.

Вершині ***1*** привласнюємо потенціал, що дорівнює 0: $P_1 = 0$.

Вершина ***1*** є початковою для дуг $[1, 2]$, $[1, 5]$, $[1, 6]$, $[1,8]$. Знаходимо потенціали кінцевих вершин за цими дугами відносно вершини ***1***:

$$p_{12} = P_1 + 9 = 0 + 9 = 9;\ p_{15} = P_1 + 7 = 7;\quad p_{16} = P_1 + 14 = 14;\ p_{18} = P_1 + 8 = 8.$$

Беремо найменший з цих потенціалів – $p_{15} = 7$ (надалі цей потенціал не розглядаємо). Вершині ***5*** привласнюємо абсолютний потенціал $P_5 = 7$ (потенціали будемо проставляти біля відповідних вершин). Дугу $[1, 5]$ відзначаємо стрілкою.

Приймаємо вершину ***5*** за початкову. Вона є початковою для дуг $[5, 4]$, $[5, 6]$. Знаходимо потенціали кінцевих вершин за цими дугами:

$$p_{54} = P_5 + 5 = 7 + 5 = 12,\quad p_{56} = P_5 + 6 = 13.$$

У наявності є такі відносні потенціали:

$$p_{12} = 9,\ \ p_{16} = 14,\ \ p_{54} = 12,\ \ p_{56} = 13,\ \ p_{18} = 8.$$

З усіх відносних потенціалів по дугах знаходимо найменший – $p_{18} = 8$. Вершині ***8*** присвоюємо абсолютний потенціал $P_8 = 8$. Дугу $[1, 8]$ відзначаємо стрілкою. Відносний потенціал $p_{18} = 8$ надалі не розглядається.

Вершину ***8*** приймаємо за початкову. Вона є початковою для дуг $[8, 6]$, $[8, 9]$, $[8, 10]$. Знаходимо потенціали кінцевих вершин за цими дугами відносно вершини ***8***:

$$p_{86} = P_8 + 13 = 8 + 13 = 21; \quad p_{89} = P_8 + 9 = 17; \quad p_{8,10} = P_8 + 5 = 13.$$

У наявності є такі відносні потенціали:

$$p_{12} = 9,\ p_{16} = 14, \quad p_{54} = 12,\ p_{56} = 13,\ p_{86} = 21,\ p_{89} = 17,\ p_{8,10} = 13.$$

Вибираємо найменший – $p_{12} = 9$. Вершині ***2*** присвоюємо абсолютний потенціал $P_2 = 9$. Дугу $[1, 2]$ відзначаємо стрілкою. Відносний потенціал $p_{12} = 9$ надалі не розглядається.

Вершину ***2*** приймаємо за початкову. Вона є початковою для дуг $[2, 3]$, $[2, 4]$. Знаходимо потенціали кінцевих вершин за цими дугами відносно вершини ***2***:

$$p_{23} = P_2 + 8 = 17; \quad p_{24} = P_2 + 10 = 19.$$

У наявності є такі відносні потенціали:

$$p_{16} = 14,\ p_{23} = 17; \quad p_{24} = 19.\ p_{54} = 12,\ p_{56} = 13,\ p_{86} = 21,\ p_{89} = 17,\ p_{8,10} = 13.$$

Вибираємо найменший – $p_{54} = 12$. Вершині ***4*** присвоюємо абсолютний потенціал $P_4 = 12$. Дугу $[5, 4]$ відзначаємо стрілкою. Отримали проміжний граф (рис. 6.3).

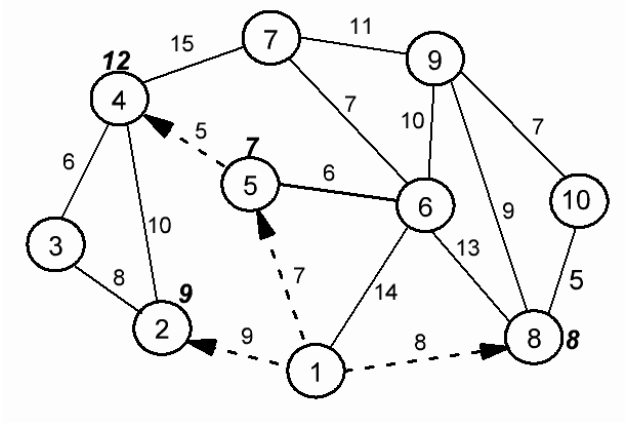

Рисунок 6.3 – Проміжний граф розрахунків

Потенціал $p_{54} = 12$, а також $p_{24} = 19$ надалі не розглядаються, оскільки $p_{24} > p_{54}$. Відповідно і дугу $[2, 4]$ не будемо використовувати у розрахунках.

Вершину ***4*** приймаємо за початкову. Вона є початковою для дуг $[4, 3]$, $[4, 7]$. Знаходимо потенціали кінцевих вершин за цими дугами відносно вершини ***4***:

$p_{43} = P_4 + 6 = 12 + 6 = 18; \quad p_{47} = 12 + 15 = 27.$

У наявності є такі відносні потенціали:

$p_{23} = 17,\ p_{43} = 18,\ p_{47} = 27,\ p_{56} = 13,\ p_{86} = 21,\ p_{89} = 17,\ p_{8,10} = 13.$

Вибираємо найменший – $p_{56} = 13$ (один з двох). Вершині ***6*** присвоюємо абсолютний потенціал $P_6 = 13$. Дугу $[5, 6]$ відзначаємо стрілкою. Відносний потенціал $p_{56} = 13$, а також $p_{86} = 21$ надалі не розглядається. Дугу $[8, 6]$ також виключаємо з розгляду.

Вершину ***6*** приймаємо за початкову. Вона є початковою для дуг $[6, 7]$, $[6, 9]$. Знаходимо потенціали кінцевих вершин за цими дугами відносно вершини ***6***:

$p_{67} = P_6 + 7 = 13 + 7 = 20; \quad p_{69} = 13 + 10 = 23.$

У наявності є такі відносні потенціали:

$p_{23} = 17,\ p_{43} = 18,\ p_{47} = 27,\ p_{67} = 20,\ p_{69} = 23,\ p_{89} = 17,\ p_{8,10} = 13.$

Вибираємо найменший – $p_{8,10} = 13$. Вершині ***10*** присвоюємо абсолютний потенціал $P_{10} = 13$. Дугу $[8, 10]$ відзначаємо стрілкою. Відносний потенціал $p_{8,10} = 13$ надалі не розглядається. Вершину ***10*** приймаємо за початкову. Вона є початковою для дуги $[10, 9]$. Знаходимо потенціал кінцевою вершиною за цією дугою відносно вершини: $p_{10,9} = P_{10} + 7 = 13 + 7 = 20$.

У наявності є такі відносні потенціали:

$p_{23} = 17,\ p_{43} = 18,\ p_{47} = 27,\ p_{67} = 20,\ p_{69} = 23,\ p_{89} = 17,\ p_{10,9} = 20.$

Вибираємо один з двох найменших – $p_{23} = 17$. Вершині ***3*** присвоюємо абсолютний потенціал $P_3 = 17$. Дугу $[2, 3]$ відзначаємо стрілкою. Відносний потенціал $p_{23} = 17$, а також $p_{43} = 18$ надалі не розглядається.

У наявності є такі відносні потенціали:

$p_{47} = 27,\ p_{67} = 20,\ p_{69} = 23,\ p_{89} = 17,\ p_{10,9} = 20$

Вибираємо найменший – $p_{89} = 17$. Вершині ***9*** присвоюємо абсолютний потенціал $P_9 = 17$. Дугу $[8, 9]$ відзначаємо стрілкою. Відносний потенціал $p_{89} = 17$, а також $p_{10,9} = 20$ і $p_{69} = 23$ надалі не розглядаються. Вершину ***9*** приймаємо за початкову. Вона є початковою для дуги $[9, 7]$. Знаходимо потенціал кінцевою вершиною за цією дугою відносно вершини: $p_{97} = P_9 + 11 = 17 + 11 = 28.$

У наявності є такі відносні потенціали: $p_{47} = 27,\ p_{67} = 20,\ p_{97} = 28.$ Вибираємо найменший – $p_{67} = 20.$ Вершині *7* присвоюємо абсолютний потенціал $P_9 = 17.$ Дугу $[6,\ 7]$ відзначаємо стрілкою. Обчислення закінчено.

У підсумку за методом потенціалів отримали підсумковий граф (рис. 6.4) із вказівкою найкоротших відстаней від вершини ***1*** до всіх інших (числа над вершинами графа) і пунктирними стрілками, що позначають найкоротші шляхи до них.

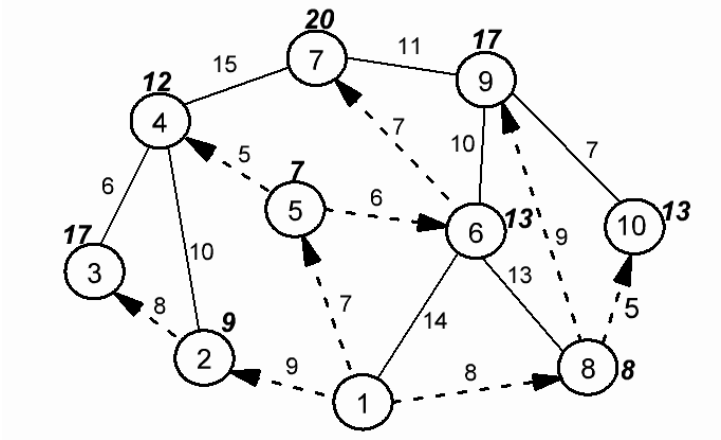

Рисунок 6.4 – Граф з найкоротшими шляхами між вузлом ***1*** та іншими

Найкоротші маршрути від пункту 1 до пунктів відповідно:

(1 – 2); (1 – 2 – 3); (1 – 5 – 4); (1 – 5); (1 – 5 – 6); (1 –5 – 6 – 7); (1 – 8); (1 – 8 – 9); (1 – 8 – 10).

Аналогічно знаходимо найкоротші відстані від всіх інших вершин. Заповнимо таблицю 6.1.

Таблиця 6.1 – Найкоротші відстані між вузлами мережі

Вершини мережі	1	2	3	4	5	6	7	8	9	10
1	-	9	17	12	7	13	20	8	17	13
2	9	-	8	10	15	21	25	17	31	22
3	17	8	-	6	11	17	21	25	27	30
4	12	10	6	-	5	11	20	20	21	25
5	7	15	11	5	-	6	13	15	16	20
6	13	21	17	11	6	-	7	13	10	17
7	20	26	21	15	13	7	-	20	II	18
8	8	17	25	20	15	13	20	-	9	5
9	17	31	27	21	16	10	11	9	-	7
10	13	22	30	25	20	17	18	5	7	-

6.1.3 Комп’ютерна реалізація моделі розрахунку найкоротших відстаней між вузлами мережі

Найбільш ефективно розрахунки по найкоротших відстанях у мережі можна виконати в системі комп’ютерної математики Maple за алгоритмом Уоршелла. Як приклад використовуємо граф на рисунку 6.2. Послідовність дій така.

Починаємо робочий аркуш із команди

> restart;   (кутову дужку > не набирати !)

Викликаємо пакет лінійної алгебри командою
> with(linalg):

Вводимо кількість вузлів мережі та описуємо змінні X і V як матриці командами

> n:=10: X:=matrix(n, n): V:= matrix(n, n):

Оскільки значна частина елементів матриці X не зв'язані прямо один з одним, то заповните всі елементи матриці X максимальним значенням, наприклад 1000 :

```
> for i from 1 to n do ↵
    for j from 1 to n do ↵
      X[i,j]:=1000 ↵
              od ↵
            od;
```

Символ ↵ набирається одночасним натисканням клавіш Shift+Enter.

Перевіряємо ввод

> evalm(X);

1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
1000	1000	1000	1000	1000	1000	1000	1000	1000	1000

Оскільки значна частина елементів матриці *V* також не зв'язані прямо один із одним, то заповните всі елементи матриці *V* ознакою відсутності зв'язку, наприклад знаком *

```
> for i from 1 to n do↵
    for j from 1 to n do↵
        V[i,j]:=`*` ↵
            od↵
        od;
```

Привласнюємо елементам матриці *X*[i,j] значення дуг для безпосередньо зв'язаних вершин по запропонованому варіанту. Наприклад:

> X[1,2]:=9: X[1,5]:=7: X[1,6]:=14: X[1,8]:=8↵ *:*

X[2,1]:=9:X[2,3]:=8:X[2,4]:=10: і так далі.

Для контролю виведемо заповнену матрицю *X*

```
>evalm(X);
```

1000	9	1000	1000	7	14	1000	8	1000	1000
9	1000	8	10	1000	1000	1000	1000	1000	1000
1000	8	1000	6	1000	1000	1000	1000	1000	1000
1000	10	6	1000	5	1000	15	1000	1000	1000
7	1000	1000	5	1000	6	1000	1000	1000	1000
14	1000	1000	1000	6	1000	7	13	10	1000
1000	1000	1000	15	1000	7	1000	1000	11	1000
8	1000	1000	1000	1000	13	1000	1000	9	5
1000	1000	1000	1000	1000	10	11	9	1000	7
1000	1000	1000	1000	1000	1000	1000	5	7	1000

Привласнюємо елементам матриці V[i,j] імена дуг для безпосередньо зв'язаних вершин по запропонованому варіанту. Наприклад:

> V[1,2]:=`1_2-`: V[1,5]:=`1_5-`: > V[1,6]:=`1_6-`:V[1,8]:=`1_8-`:↵

V[2,1]:=`2_1-`:V[2,3]:=`2_3-`:V[2,4]:=`2_4-`:↵ і т.д.

Для контролю командою >evalm(V); виводимо заповнену матрицю *V.*

```
>evalm(V);
```

*	1 2-	*	*	1 6-	1 7-	*	1 8-	*	*
2 1-	*	2 3-	2 4-	*	*	*	*	*	*
*	3 2-	*	3 4-	*	*	*	*	*	*
*	4 2-	4 3-	*	4 5-	*	4 7-	*	*	*
5 1-	*	*	5 4-	*	5 6-	*	*	*	*
6 1-	*	*	*	6 5-	*	6 7-	6 8-	6 9-	*
*	*	*	7 4-	*	7 6-	*	*	7 9-	*
8 1-	*	*	*	*	8 6-	*	*	8 9-	8 10-
*	*	*	*	*	9 6-	9 7-	9 8-	*	9 10-
*	*	*	*	*	*	*	10 8-	10 9-	*

Набираємо і відпрацьовуємо алгоритм вибору оптимальних маршрутів передачі інформації по запропонованому варіанту ЛКМ:

```
> m:=1000:
> for  j  from 1 to n do↵
      for i  from 1 to  n do↵
      if  X[i,j]<m then ↵
          for k  from 1 to n do↵
          if  X[i,j]+X[j,k]<X[i,k] then  X[i,k]:=X[i,j]+X[j,k]: ↵
                                         V[i,k]:=V[i,j]*V[j,k] ↵
          fi:                  od↵
      fi:                  od↵
                 od:
```

Викреслюємо циклічні зв’язки

```
>for s  from  1 to n  do V[s,s]:=`-`: X[s,s]:=`-`  od:
```

Введення допоміжних матриць для візуального виводу результатів розрахунків.

```
> Xr:=matrix(n+1,n+1): Vr:=matrix(n+1,n+1):
> Xr[1,1]:=` `:Vr[1,1]:=` `:
```

Введення імен вузлів мережі

```
> for j from 2 to n+1 do↵
    Xr[1,j]:=j-1;Vr[1,j]:=j-1↵
            od:
> for i from 2 to n+1 do↵
    Xr[i,1]:=i-1;Vr[i,1]:=i-1↵
            od:
```

Копіювання матриць

```
>  copyinto(X,Xr,2,2):copyinto(V,Vr,2,2):
```

Виведення найкоротших маршрутів мережі

```
> evalm(Vr);
```

	1	2	3	4	5	6	7	8	9	10
1	-	1_2-	1_2-2_3-	1_5-5_4-	1_5-	1_5-5_6-	1_5-5_6-6_7-	1_8-	1_8-8_9-	1_8-8_10-
2	2_1-	-	2_3-	2_4-	2_4-4_5-	2_4-4_5-5_6-	2_4-4_7-	2_1-1_8-	2_1-1_8-8_9-	2_1-1_8-8_10-
3	3_2-2_1-	3_2-	-	3_4-	3_4-4_5-	3_4-4_5-5_6-	3_4-4_7-	3_2-2_1-1_8-	3_4-4_5-5_6-6_9-	3_2-2_1-1_8-8_10-
4	4_5-5_1-	4_2-	4_3-	-	4_5-	4_5-5_6-	4_7-	4_5-5_1-1_8-	4_5-5_6-6_9-	4_5-5_1-1_8-8_10-
5	5_1-	5_4-4_2-	5_4-4_3-	5_4-	-	5_6-	5_6-6_7-	5_1-1_8-	5_6-6_9-	5_1-1_8-8_10-
6	6_5-5_1-	6_5-5_4-4_2-	6_5-5_4-4_3-	6_5-5_4-	6_5-	-	6_7-	6_8-	6_9-	6_9-9_10-
7	7_6-6_5-5_1-	7_4-4_2-	7_4-4_3-	7_4-	7_6-6_5-	7_6-	-	7_6-6_8-	7_9-	7_9-9_10-
8	8_1-	8_1-1_2-	8_1-1_2-2_3-	8_1-1_5-5_4-	8_1-1_5-	8_6-	8_6-6_7-	-	8_9-	8_10-
9	9_8-8_1-	9_8-8_1-1_2-	9_6-6_5-5_4-4_3-	9_6-6_5-5_4-	9_6-6_5-	9_6-	9_7-	9_8-	-	9_10-
10	10_8-8_1-	10_8-8_1-1_2-	10_8-8_1-1_2-2_3-	10_8-8_1-1_5-5_4-	10_8-8_1-1_5-	10_9-9_6-	10_9-9_7-	10_8-	10_9-	-

Виводимо матрицю найкоротших відстаней між вузлами мережі

> *evalm(Xr);*

` `	**1,**	**2,**	**3,**	**4,**	**5,**	**6,**	**7,**	**8,**	**9,**	**10**
1,	`-`,	9,	17,	12,	7,	13,	20,	8,	17,	13
2,	9,	`-`,	8,	10,	15,	21,	25,	17,	26,	22
3,	17,	8,	`-`,	6,	11,	17,	21,	25,	27,	30
4,	12,	10,	6,	`-`,	5,	11,	15,	20,	21,	25
5,	7,	15,	11,	5,	`-`,	6,	13,	15,	16,	20
6,	13,	21,	17,	11,	6,	`-`,	7,	13,	10,	17
7,	20,	25,	21,	15,	13,	7,	`-`,	20,	11,	18
8,	8,	17,	25,	20,	15,	13,	20,	`-`,	9,	5
9,	17,	26,	27,	21,	16,	10,	11,	9,	`-`,	7
10,	13,	22,	30,	25,	20,	17,	18,	5,	7,	`-`

Зауважимо, що матриця симетрична, відносно головної діагоналі, оскільки у нашому випадку шлях між двома вершинами мережі рівноцінний в обох напрямках.

6.2 Завдання на практичне заняття

1. Вивчить теоретичний матеріал.

2. Проаналізуйте запропонований варіант практичного заняття (рис. 6.5), перемалюйте у звіт графову структуру ЛКМ із конкретними значеннями дуг суміжних вузлів (*i* – остання цифра студентського білету, *j* - передостання цифра студентського білету).

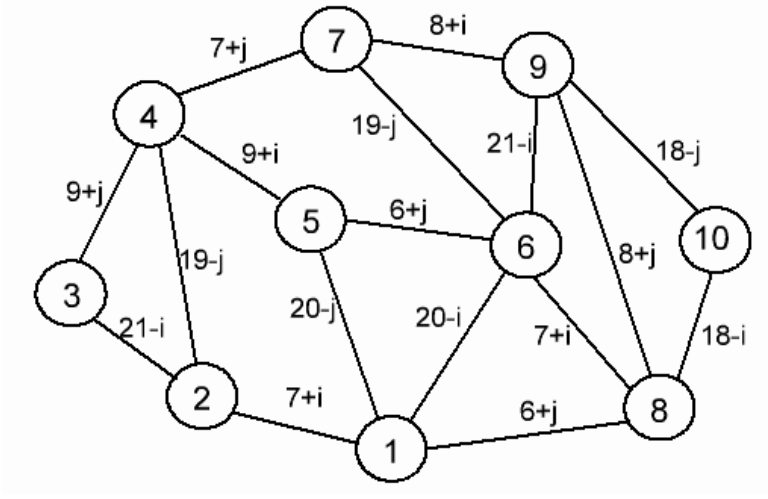

Рисунок 6.5 – Граф ЛКМ до завдання

3. По своєму варіанту мережі розрахуйте методом потенціалів найкоротші відстані між вибраним вузлом інформаційної мережі та іншими, а також на графі виділить відповідні оптимальні маршрути.

4. У системі Maple складіть робочий лист по розрахунку найкоротших відстаней між вузлами мережі й оптимальних маршрутів згідно зі своїм варіантом. За зразок візьміть п.6.1.3. Відпрацюйте його. Випишіть у звіт матрицю найкоротших відстаней і найкоротші маршрути між вузлами ЛКМ. Порівняйте результати обчислень вручну та на комп'ютері.

5. Захистить роботу перед викладачем.

Контрольні запитання і завдання

1. Що таке локальна комп'ютерна мережа і для чого вона створюється?
2. Як передається інформація з одного комп'ютера на інший.
3. Який критерій використовується для оптимізації інформаційних потоків у ЛКМ ?
4. У чому зміст задачі вибору найкоротших маршрутів?
5. Який математичний апарат використовується для оптимізації інформаційних потоків у ЛКМ
6. Поясніть алгоритм обчислення найкоротших маршрутів за методом потенціалів.
7. У чому перевага системи Maple перед традиційними системами програмування при розв'язуванні задачі обчислення найкоротших маршрутів?

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Кравчук С. О. Основи комп’ютерної техніки: Компоненти, системи, мережі : навч. посіб. для студ. вищ. навч. закл. / С. О. Кравчук, В. О. Шонін. – Київ : ІВЦ «Видавництво «Політехніка»: Видавництво «Каравела», 2005. – 344 с.

2. Литвинов А. Л. Математика (линейная алгебра и математический анализ) : Учебно-методический комплекс / А. Л. Литвинов. – Белгород : Изд-во БулГУ, 2005. – 263 с.

3. Каган Б. М. Электронные вычислительные машины и системы : учеб. пособие для вузов / Б. М. Каган. – М. : Энергоатомиздат, 1991. – 592 с.

4. Мельник А. О. Архітектура комп’ютера / А. О. Мельник. – Луцьк : Волинська обласна друкарня, 2008. – 470 с.

5. Таненбаум, Э. Архитектура компьютера / Э. Таненбаум, Т. Остин. – СПб. : Питер, 2013. – 816 с.

6. Яковлев А. В. Криптографическая защита информации : учеб. пособие / А. В. Яковлев, А. А. Безбогов, В. В. Родин, В. Н. Шамкин. – Тамбов: Изд-во Тамб. гос. техн. ун-та, 2006. – 140 с.

7. Берзтисс А. Т. Структури даних / А. Т. Берзтисс. – М. : Статистика, 1974. – 408 с.

Навчальне видання

ЛИТВИНОВ Анатолій Леонідович

ПРАКТИКУМ З АРХІТЕКТУРИ КОМП'ЮТЕРНИХ СИСТЕМ

НАВЧАЛЬНИЙ ПОСІБНИК

Відповідальний за випуск *М. М. Булаєнко*

Редактор *О. В. Михаленко*

Комп'ютерне верстання *А. Л. Литвинов*

Підп. до друку 17.05.2020. Формат 60×84/16.
Друк на ризографі. Ум. друк. арк. 4,0.
Тираж 50 пр. Зам. №

Видавець і виготовлювач:
Харківський національний університет міського
господарства імені О. М. Бекетова,
вул. Маршала Бажанова,17, Харків, 61002.
Електронна адреса: rectorat@kname.edu.ua
Свідоцтво суб'єкта видавничої справи:
ДК № 5328 від 11.04.2017.