МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки

Комп'ютерна логіка - 2. Комп'ютерна арифметика

Методичні вказівки до виконання лабораторних робіт
для студентів напряму підготовки
«123 – Комп'ютерна інженерія»
кафедри обчислювальної техніки
Форма навчання: заочна

Рекомендовано кафедрою
обчисленної техніки
Протокол № ___
від _29_._06_.2017 р.
Завідувач кафедри

_____Стіренко С.Г.

Київ
НТУУ «КПІ імені Ігоря Сікорського»
2017

Комп'ютерна логіка -2. Комп'ютерна арифметика. Методичні вказівки до виконання лабораторних робіт. [Електронне видання.Текст] / Уклад.: В.І.Жабін, О.А.Верба. НТУУ «КПІ імені Ігоря Сікорського», 2017. – 55 с.

Методичні вказівки призначено для виконання лабораторних робіт для студентів напряму підготовки „123 – Комп’ютерна інженерія” заочної форми навчання.

Укладачі: д.т.н., проф. Жабін В.І.

к.т.н., доц. Верба О.А.

За редакцією укладачів

ЗМІСТ

1. Лабораторна робота №1. Дослідження методів подання даних та виконання однотактних операцій в комп’ютерах.

2. Лабораторна робота №2. Проектування та дослідження пристроїв для множення чисел.

3. Лабораторна робота №3. Проектування та дослідження пристроїв для ділення чисел.

4. Лабораторна робота №4. Дослідження операцій додавання та віднімання в двійково-кодованих системах числення.

Додаток А. Програмний комплекс для виконання лабораторних робіт.

1. Лабораторна робота №1
ДОСЛІДЖЕННЯ МЕТОДІВ ПОДАННЯ ДАНИХ ТА ВИКОНАННЯ ОДНОТАКТНИХ ОПЕРАЦІЙ В КОМП'ЮТЕРАХ

Мета роботи: вивчення методів та засобів подання чисел в комп'ютерах з використанням машинних кодів, одержати навички побудови та опису операційних схем для виконання однотактних операцій, оволодіти програмним комплексом моделювання та дослідження цифрових пристроїв.

Теоретичні відомості

Кодування чисел в комп'ютерах

Найбільше поширення в обчислювальній техниці (ОТ) має двійкова однорідна позиційна система числення з цифрами {0, 1} та природним порядком вагів розрядів. Для подання чисел та виконання операцій з числами, що мають знаки, використовують спеціальні машинні коди:

- прямий код (ПК),
- обернений код (ОК),
- доповняльний код (ДК).

Будемо вважати, що розрядна сітка містить число, яке має *n*-розрядну цілу частину, *k*-розрядну дробову частину і знаковий розряд зліва від основних розрядів (рис. 2.1).

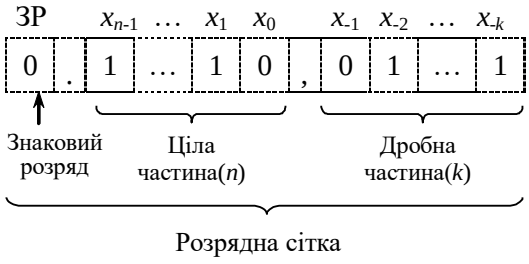


Рис. 1.1. Приклад запису числа зі знаком, що має цілу та дробову частину

Старший розряд цілої частини має вагу 2^{n-1}, а молодший розряд дробової частини – вагу 2^{-k}.

Подання числа *X* у *прямому коді* визначається виразом:

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

Під час утворення прямого коду знаковий розряд дорівнює 0, якщо число додатне і 1, якщо число від'ємне.

Під час запису числа знаковий розряд відокремлюється від основних розрядів крапкою, а ціла частина числа від дробової – комою. У випадку, коли числа не мають цілої частини, то знаковий розряд відокремлюється від основних розрядів комою.

Приклад 1.1. Записати числа А=10,101011; *B*= –10,0111010 у ПК.

Виконання завдання:

$[A]_{\text{ПК}}$ = 0.10, 101011; $[B]_{\text{ПК}}$ =1.10, 0111010

*Приклад 1.2***.** Записати числа *A*=0,101011; *B*= – 0,0111010 у ПК.

Виконання завдання:

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

ПК застосовується для зберігання чисел в пам'яті комп'ютера та виконання деяких операцій (наприклад, множення, ділення та ін.). Для операцій додавання і віднімання ПК не використовується.

Під час перетворення від'ємного числа в *обернений код*, у знаковий розряд записується 1, а значення основних розрядів інвертуються, тобто, у кожнім розряді 0 замінюється на 1, а 1 замінюється на 0. Додатне число у ОК збігається із числом у ПК, тобто основні розряди не інвертуються, у знаковий розряд записується 0.

Формула перетворення чисел у ОК код має вигляд:

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

Приклад 1.3. Записати числа *A*=10,1011; *B*= – 01,11010 в ОК.

Виконання завдання:

$[A]_{\text{ОК}}$ = 0.10,1011; $[B]_{\text{ОК}}$ = 1.10,00101.

Приклад 1.4. Записати числа *A*=101011; *B*= – 0111010 в ЗК.

Виконання завдання:

$[A]_{\text{ОК}}$ = 0.101011; $[B]_{\text{ОК}}$ = 1.1000101.

Приклад 1.5. Записати числа *A*=0,101011; *B*= – 0,0111010 в ЗК.

Виконання завдання:

$[A]_{\text{ОК}}$ = 0,101011; $[B]_{\text{ОК}}$ = 1,1000101.

Під час перетворення від'ємного числа в *доповняльний код*, у знаковий розряд записується 1, а значення основних розрядів інвертуються, після чого до молодшого розряду додається 1 (з поширенням переносів між розрядами). Додатне число в ДК збігається з числами у ПК і ЗК, тобто в знаковий розряд записується 0, а основні розряди не змінюються.

Формула перетворення чисел у доповнювальний код має вигляд:

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

Приклад 1.6. Записати числа *A*= – 011,100; *B*= 010,010 у ПК, ОК і ДК.

Виконання завдання:

$[A]_{ПК}$=1.010,010;
$[A]_{ОК}$=1.101,101;
+_____1
$[A]_{ДК}$=1.101,110.

$[B]_{ПК}$=0.010,010;
$[B]_{ОК}$=0.010,010;
$[B]_{ДК}$=0.010,010.

Приклад 1.7. Записати числа A= – 0,011100; B= 0,010010 у ПК, ОК і ДК.

Виконання завдання:

$[A]_{ПК}$ =1 , 010010;
$[A]_{ОК}$ =1 , 101101;
+_____1
$[A]_{ДК}$ =1 , 101110.

$[B]_{ПК}$ =0 , 010010;
$[B]_{ОК}$ =0 , 010010;
$[B]_{ДК}$ =0 , 010010.

Додавання чисел із знаками у машинних кодах

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

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

Для виявлення переповнення використовують *модифікомані машинні коди*, в яких вводиться додатковий (другий) знаковий розряд $ЗР_2$ ліворуч основного (першого) розряда $ЗР_1$. У разі переповнення сітки старший знаковий розряд завжди зберігає знак результату. Ознака переповнення визначається функцією OVR=$ЗР_2 \oplus ЗР_1$.

Формули подання модифікованих кодів мають вигляд:

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

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

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

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

Приклад 1.9. Задано $A = 0{,}10101$; $B = -0{,}01001$. Знайти суму C з використанням ОК.

Виконання завдання:

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

Приклад 1.10. Задано $A = 0.01001$; $B = -0.10101$. Знайти суму C з використанням ОК.

Виконання завдання:

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

(Перенос відсутній)

Приклад1.11. Задано обернені коди чисел $A = -0.10101$ і $B = -0.01001$. Знайти обернений код суми C.

Виконання завдання:

$$
\begin{array}{rl}
 & [A]_{\text{зк}} = 11,01010 \\
+ & \underline{[B]_{\text{зк}} = 11,10110} \\
 & 11,00000 \\
 & \underline{+\qquad\quad 1} \text{ (перенос)} \\
 & [C]_{\text{зк}} = 11,00001
\end{array}
$$

Приклад 1.12. Задано $A = 0{,}10101$ і $B = 0{,}01110$. Знайти суму C з використанням ОК.

.Виконання завдання:

$$
\begin{array}{rl}
 & [A]_{\text{зк}} = 00,10101 \\
+ & \underline{[B]_{\text{зк}} = 00,01110} \\
 & [C]_{\text{зк}} = 01,00011 \text{ – додатне переповнення}
\end{array}
$$

Приклад 1.13. Задано $A = -0.10101$; $B = -0.01110$. Знайти суму C.

Виконання завдання:

$$
\begin{array}{rl}
 & [A]_{\text{зк}} = 11,01010 \\
+ & \underline{[B]_{\text{зк}} = 11,10001} \\
 & 10,11011 \\
 & \underline{+\qquad\quad 1} \text{ (перенос)} \\
 & [C] = 10,01100 \text{ – від'ємне переповнення}
\end{array}
$$

Функціональна схема пристрою, що реалізує операцію додавання і віднімання в ОК наведена на рис. 1.2.

Операнди надходять на вхід суматора *SM* з регістрів *RGx* і *RGy*. Операція віднімання виконується шляхом додавання зменшуваного до від'ємника, який інвертується за допомогою елемента ВИКЛЮЧНЕ АБО. Для цього на вхід *COM* подається одиничний сигнал. Під час додавання на цей вхід потрібно подати сигнал 0. Суматор формує основні розряди суми (ОР) і два знакових розряди $ЗР_2$ і $ЗР_1$.

Корекція суми здійснюється за допомогою ланцюга циклічного переносу зі знакового розряду в молодший розряд суматора *(CO→ CI)*. Цей ланцюг завжди замкнутий. Перенос автоматично виникає тільки тоді, коли

потрібно реалізувати корекцію суми. При значенні *OVR* = 0, переповнення розрядної сітки відсутнє, у випадку *OVR* = 1 наявне переповнення розрядної сітки.

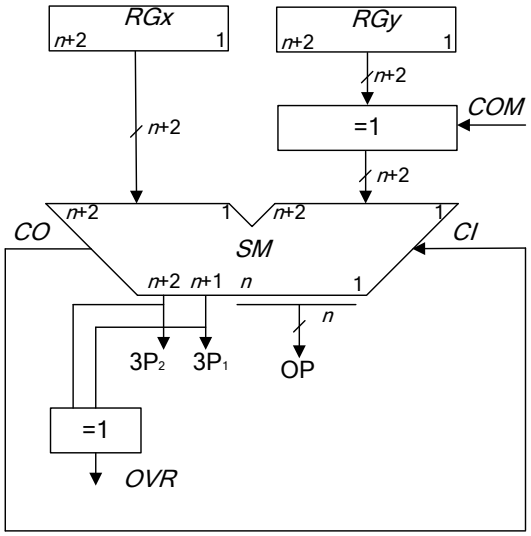


Рис. 1.2. Схема виконання операцій додавання і віднімання в оберненних кодах

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

Приклад 1.14. Задано A= 0,10101; B= 0,01001. Знайти суму C в ДК.

Виконання завдання:

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

Приклад 1.15. Задано A= 0.10101; B= – 0.01001. Знайти суму C в ДК.

Виконання завдання:

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

Приклад 1.16. Для A= 0.01001 і B= – 0.10101 знайти суму C в ДК.

Виконання завдання:

$$\begin{array}{r} [A]_{\text{ДК}} = 00,01001 \\ +\underline{[B]_{\text{ДК}} = 11,01011} \\ [C]_{\text{ДК}} = 11,10100 \end{array}$$

Приклад 1.17. Задано $A = -0.10101$; $B = -0.01001$. Знайти C в ДК.

Виконання завдання:

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

Приклад 1.18. Задано $A = 0.10101$; $B = 0.01110$. Знайти суму C в ДК. Визначити знак результату.

Виконання завдання:

$$\begin{array}{r} [A]_{\text{ДК}} = 00,10101 \\ +\underline{[B]_{\text{ДК}} = 00,01110} \\ [C]_{\text{ДК}} = 01,00011 \end{array}$$ – додатне переповнення

У цьому випадку наявне додатне переповнення розрядної сітки, та за застосування модифікованого коду старший розряд зберігає знак результату. Отже отримане в результаті обчислень число є додатним.

Приклад 1.19. Задано $A = -0.10101$ і $B = -0.01110$. Знайти суму C в ДК. Визначити знак результату.

Виконання завдання:

$$\begin{array}{r} [A]_{\text{ДК}} = 11,0\ 1011 \\ +\underline{[B]_{\text{ДК}} = 11,1\ 0010} \\ [C]_{\text{ДК}} = 10,1\ 1101 \end{array}$$ – від'ємне переповнення

У цьому випадку наявне від'ємне переповнення розрядної сітки, та за застосування модифікованого коду старший розряд зберігає знак результату. Отже отримане в результаті обчислень число є від'ємним.

Схема, що реалізує операцію додавання і віднімання в ДК, аналогічна схемі для зворотних кодів (рис. 1.2), але в цьому випадку відсутній циклічний ланцюг корекції результату (*CO→ CI)*. За віднімання разом з одиничним сигналом на вхід *COM* подається одиниця на вхідний перенос *CI* су-

матора. Це необхідно для зміни знака від'ємника, яке здійснюється інвертуванням усіх розрядів від'ємника і додаванням одиниці до його молодшого розряду. Під час додавання на входи *COM* і *CI* потрібно подати значення 0.

Зсуви машинних кодів.

Існують два різновиди машинних зсувів:

- логічний зсув;
- арифметичний зсув.

Логічний зсув це зміщення розрядів машинного слова у просторі із втратою розрядів, що виходять за межі розрядної сітки. Розряди, що звільняються заповнюються нулями. Схема логічного зсуву чисел зображена на рис. 1.3.

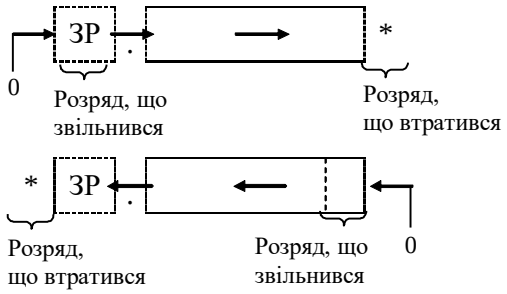


Рис. 1.3. Операційна схема логічного зсуву двійкових чисел

Приклад 1.20. Виконати логічний зсув двійкового числа вліво і вправо на один розряд.

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

Виконання завдання:

$$\begin{array}{ll} A & |1.0101,1101| \\ A\rightarrow & |0.1010,1110|* \end{array}$$

$$\begin{array}{ll} A & |1.0101,1101| \\ \leftarrow A & *|0.1011\ 1010| \end{array}$$

Арифметичний зсув виконується з урахуванням знакового розряду. Правила зсуву чисел поданих у ПК, ЗК та ДК відрізняються. Арифметич-

ний зсув ліворуч означає множення числа на 2 (тобто на основу системи числення), а зсув праворуч – ділення числа на 2.

Арифметичний зсув чисел поданих у ПК

При цьому типі зсуву знаковий розряд не зсувається. Основні розряди зсуваються ліворуч або праворуч із заповненням нулями розрядів, що звільнилися при зсуві. Розряди, що вийшли за межі розрядної сітки втрачаються. За арифметичного зсуву вліво можлива втрата значимості числа. За зсуву праворуч виникає похибка. Схема арифметичного зсуву чисел поданих у ПК зображена на рис. 1.4

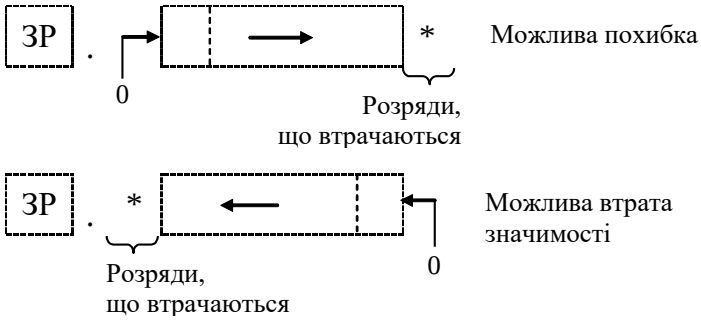


Рис. 1.4. Операційна схема арифметичного зсуву чисел у ПК

Приклад 1.21. Виконати арифметичний зсув вліво і вправо на один розряд двійкових чисел, поданих у ПК.

$A_{(2)} = 1.\ 10101$; $B_{(2)} = 0.\ 11011$.

Виконання завдання:

A	1. 1 0 101	
$A\rightarrow$	1. 0 1 010*	Похибка
$\leftarrow A$	1.*0 1 010	Втрата значимості
B	0. 1 1 011	
$B\rightarrow$	0. 0 1 101*	Похибка
$\leftarrow B$	0.*1 0 110	Втрата значимості

Арифметичний зсув ліворуч чисел, поданих у ОК.

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

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

На рис. 1.5 зображена операційна схема реалізації арифметичного зсуву ліворуч від'ємних чисел поданих у ОК.

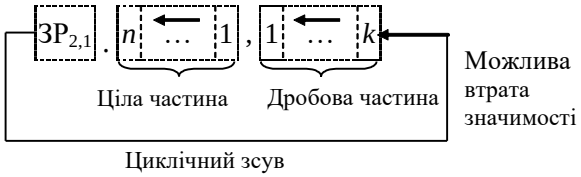
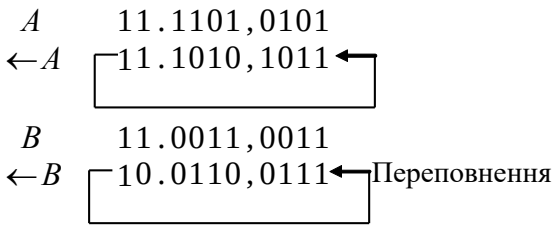


Рис. 1.5. Операційна схема арифметичного зсуву ліворуч від'ємних чисел у ОК

Приклад 1.22. Виконати арифметичний зсув ліворуч на один розряд двійкових чисел, поданих у ОК.

$A_{(2)} = 11.\ 1101,\ 1010$; $B_{(2)} = 11.\ 0011,\ 0011$.

Виконання завдання:



Арифметичний зсув праворуч чисел, поданих у ОК.

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

На рис. 1.6 зображена операційна схема реалізації арифметичного зсуву праворуч від'ємних чисел поданих у ЗК.

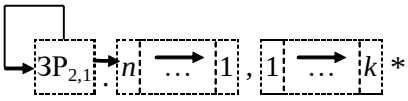


Рис. 1.6. Операційна схема арифметичного зсуву праворуч від'ємних чисел у ОК

Приклад 1.23. Виконати арифметичний зсув праворуч на один розряд двійкових чисел, поданих у ЗК.

$A_{(2)}$ = 00. 1011, 1001; $B_{(2)}$ = 11. 1010, 0011.

Виконання завдання:

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

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

Арифметичний зсув ліворуч чисел, поданих у ДК

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

На рис. 1.7 зображена операційна схема реалізації арифметичного зсуву ліворуч чисел поданих у ДК.

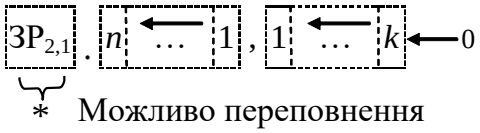


Рис. 1.7. Операційна схема арифметичного зсуву ліворуч від'ємних чисел у ДК

Арифметичний зсув праворуч чисел, поданих у ДК

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

Операційна схема реалізації арифметичного зсуву праворуч чисел поданих у ДК зображена на рис. 1.8.

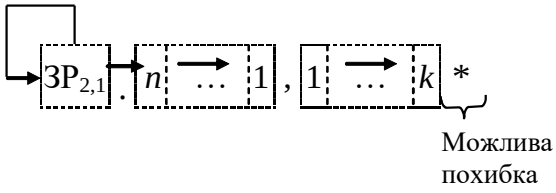


Рис. 1.8. Операційна схема арифметичного зсуву праворуч від'ємних чисел у ДК

Приклад 1.24. Виконати арифметичний зсув праворуч і ліворуч на один розряд двійкових чисел, поданих у ДК.

$A_{(2)} = 0.\ 1011,\ 0111$; $B_{(2)} = 1.\ 1011,\ 1001$.

Виконання завдання:

$$\begin{array}{ll} A & 00.1011,0111 \\ \leftarrow A & 01.0110,1110 \\ A \rightarrow & 00.0101,1011^* \end{array}$$

$$\begin{array}{ll} B & 11.1011,1001 \\ \leftarrow B & 11.0111,0010 \\ \rightarrow B & 11.1101,1100^* \end{array}$$

Операційні схеми та мікроалгоритми.

Перетворення інформації в арифметико-логічних пристроях комп'ютерів провадиться шляхом послідовного виконання мікрооперацій над машинними словами (кодами чисел, символами та іншими об'єктами).

Під *мікрооперацією* (МО) розуміють елементарну дію, в результаті виконання якої можуть змінюватися значення машинних слів.

Машинне слово можна задати перерахуванням розрядів чи за допомогою ідентифікаторів. Наприклад, 01011001 – машинне слово, яке завдано переліком всіх 8-ми розрядів. Ідентифікатори можуть подаватися рядком символів (букв та цифр), починаючи з букви. Доцільно у якості ідентифікаторів використовувати позначення вузлів, на яких виконуються мікрооперації, наприклад: *RG1, CT*. Для визначення довжини слів та впорядкування номерів розрядів використовують додаткові дані у дужках, наприклад *RG*1[0…31], *CT*[7…0].

Алгебраїчні перетворення даних у цифрових пристроях виконуються за допомогою таких МО як *пересилання*, *підсумовування*, *зсув*, *підрахування* (*декремент, інкремент*), *інвертування*.

Для позначення мікрооперації будемо використовувати оператор присвоювання (: =). Інформаційний зв'язок між вузлами цифрових пристроїв пояснюється операційною схемою, на який визначається напрямок передачі даних. Управляючі сигнали не показують, бо це визначається особливостю елементної бази. Такі сигнали показують на функціональних і принципових схемах.

Приклади операційних схем для виконання МО пересилання *RG*1:=*DATA*, *RG*1:=*RG*2, *RG*1[31…24]:=*RG*2[7…0] відповідно показані на рис. 1.9.

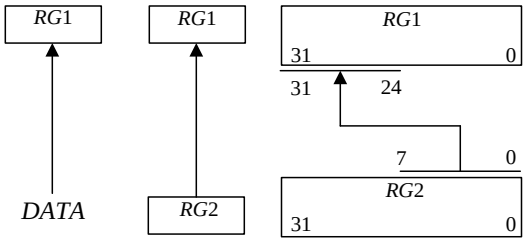
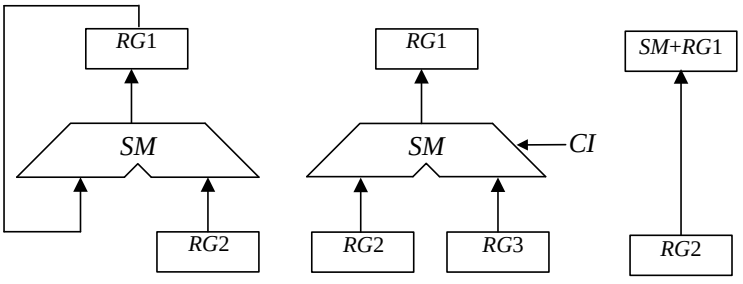


Рис. 1.9. Операційні схеми для виконання мікрооперацій пересилання даних

Для підсумовування слів в схемі використовують суматор. Приклади операційних схем, що відповідають мікрооперацям *RG*1:=*RG*1+*RG*2 і *RG*1:=*RG*2+*RG*3+*CI* (де *CI* – перенос у молодший розряд суматора) відповідно показані на рис. 1.10. В схемах рис. 1.10, *а* і 1.10, *б* використовуються комбінаційні суматори, а в схемі рис. 1.12, *в* – накопичуючий суматор, який є композицією суматора і регістра.

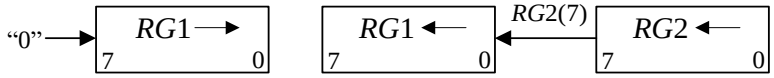
Мікрооперації інкременту (*CT* := *CT*+1) та декременту $(CT := CT - 1)$ виконуються на лічильнику.



а *б* *в*

Рис. 1.10. Операційні схеми для виконання мікрооперацій підсумовування чисел: *а, б* – із застосуванням комбінаційних суматорів; *в* – з використанням накопичуючого суматора.

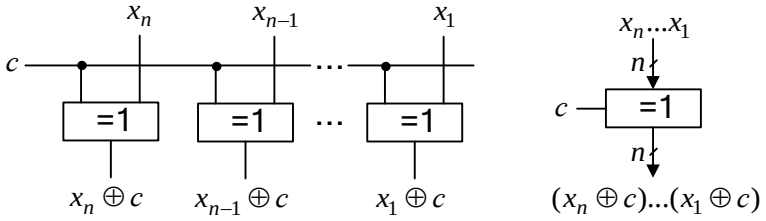
Мікрооперації зсуву слів можуть бути виконані на регістрі зсуву праворуч або ліворуч. Для означення напрямку зсуву використовують оператори зсуву *r* (*right*) та *l* (*left*). За допомогою складеного слова визначають значення розряду, який заповнюється внаслідок зсуву. Приклад операційної схеми, що відповідає мікрооперації зсуву *RG*1:=0.*r*(*RG*1) зображений на рис 1.11 *а*. Для реалізації зсуву, що відповідає операційній схемі на рис. 1.11, *б* необхідно виконати дві МО в одному такті *RG*2:=*l*(*RG*2).0 та *RG*1:=*l*(*RG*1).*RG*2[7].



а *б*

Рис. 1.11. Операційні схеми для виконання мікрооперацій зсуву

Інвертування розрядів двійкових чисел можна забезпечити інвертуванням розрядів регістру, що зберігає це число, або використанням лінійки логічних елементів при пересиланні числа, наприклад, елементів ВИКЛЮЧНЕ АБО (рис. 1.12).



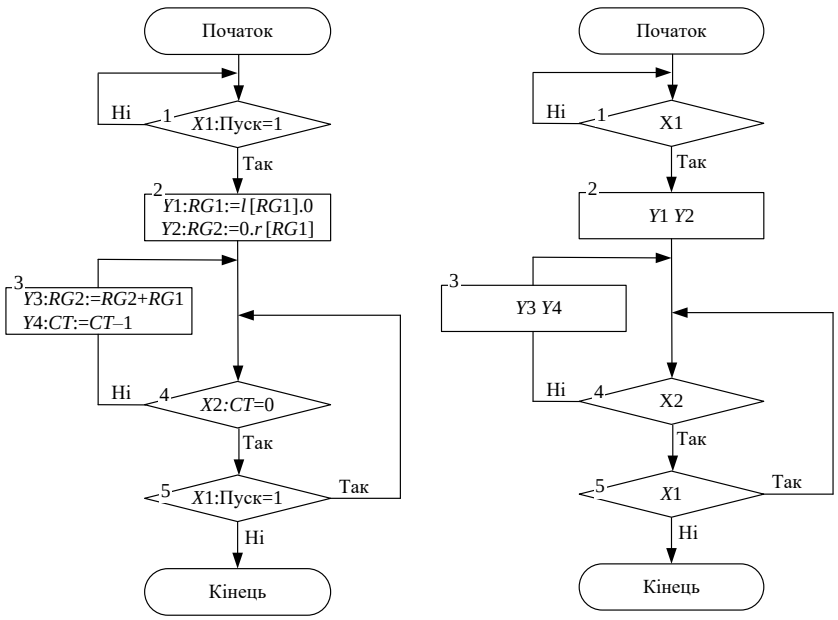
а *б*

Рис. 1.12. Інвертування розрядів x_i двійкового числа: *а* – логічна схема; *б* – умовне позначення (*с* – сигнал управління інвертуванням)

Послідовність мікрооперацій, що забезпечує задане перетворення інформації, називається *мікроалгоритмом* (МА).

Для опису МА використовують *схеми мікроалгоритмів*, яки можуть бути описані мовами ГСА (графічна схема алгоритмів) і ЛСА (логічна схема алгоритмів).

Мікроалгоритми можуть складатися як у змістовній, так і в закодованій формі. Для складання змістовного мікроалгоритму мікрооперації записують у змістовній формі, наприклад, через оператори присвоювання або їх ідентифікаторів (рис. 1.13). Для розробки такого мікроалгоритму достатньо скласти операційну схему пристрою, який виконує задані МО.



а *б*

Рис. 1.13. Приклад запису мікроалгоритму: *а* – з розширеним змістовним описом мікрооперацій; *б* – із скороченим описом мікрооперацій у вигляді ідентифікаторів

Для складання закодованого мікроалгоритму необхідна більш докладніша схема (наприклад, функціональна), яка пояснює спосіб управління

кожною мікрооперацією, включаючи означення управляючих сигналів. Змістовні МО у закодованому мікроалгоритмі замінюються на сукупність управляючих сигналів, які забезпечують виконання мікрооперацій.

Підготовка до лабораторної роботи

1. Вивчити теоретичні відомості, інформацію в Додатку А та, при необхідності, відповідні розділи в літературі, що подається в списку.

2. Визначити свій варіант завдання. Для цього перевести десятковий номер залікової книжки студента в двійкову систему числення і виділити молодші розряди $a_7, a_6, a_5, a_4, a_3, a_2, a_1$ двійкового числа.

3. Визначити два двійкових числа: $F = 1a_7a_6a_5a_41$ і $G = 1011a_3a_2a_11$. Записати F і G через кому, надати числу знак «-». Одержане двійкове від'ємне число має вигляд $X = -F, G = -1a_7a_6a_5a_41,1011a_3a_2a_11$.

4. Одержане двійкове число X з природною фіксованою комою записати у 15-розрядну сітку в машинних кодах: прямому, доповняльному і оберненому.

5. Подати модифіковані коди (доповняльний і обернений) у 16-розрядній сітці.

6. Виконати арифметичний зсув одержаних модифікованих кодів числа X на один розряд ліворуч і на один розряд праворуч. Перевірити переповнення розрядної сітки.

7. Одержати доповняльний та обернений коди числа $Y = X + 101a_4a_3,1a_2a_110$.

8. Виконати підсумування $Z = X + Y$ в доповняльних і обернених кодах.

9. Виконати підсумування $N = X + (-Y)$ в доповняльних і обернених кодах.

10. Розробити функціональні схеми перетворення 15-розрядного числа, поданого в ПК, в 16-розрядні модифіковані ОК і ДК.

11. У відповідності з операційною схемою (рис.1.14) розробити функціональну схему, що виконує мікрооперації:

- перетворює 15-розрядні операнди *E* і *H*, подані ПК, в модифіковані 16-розрядні коди згідно з варіантом (ДК при a_1=0) і (ОК при a_1 =1);

- виконує мікрооперації додавання і віднімання модифікованих ДК або ОК (за варіантом);

- записує результат додавання (віднімання) в регістр зсуву;

- виконує арифметичний зсув модифікованих кодів на один розряд ліворуч і проворуч.

12. Описати мікроалгоритми, що виконуються пристроєм, за допомогою ГСА в змістовних мікрооперaціях (як на рис. 1.13). Одержати закодований мікроалгоритм, в якому змістовні мікрооперації замінені на управляючі сигнали, що забезпечують їх виконання, наприклад: W – запис кода в регістр, SL – зсув кода в регістре ліворуч, SR – зсув кода в регістре праворуч.

Рис. 1.14. Операційна схема пристрою

Порядок виконання роботи

1. За допомогою моделюючего комплексу ПРОГМОЛС-2 вівчити спосіб управління всіма логічними елементами і вузлами (регістрами, суматорами), що необхідні для побудови розроблених функціональниї схем.

2. Побудувати і відлагодити всі розроблені функціональні схеми.

3. Промоделювати роботу пристроїв для різних наборів операндів.

4. Зробити висновки по роботі.

Контрольні питання

1. Яким чином представляються числа із знаками в компютерах?

2. Які машинні коди використовують для виконання операцій додавання і віднімання?

3. Поясніть правила подання чисел із знаками в різних машинних колах.

4. Поясніть правила зсуву чисел в ПК, ОК і ДК.

5. Поясніть правила додавання та віднімання чисел в ОК і ДК.

6. Як можна виявити переповнення розрядної сітки при віконанні операцій з машинними кодами?

7. Яким чином можна подати мікрооперації і мікроалгоритми?

Список літератури

1. Жабін В.І., Жуков І.А., Клименко І.А., Стиренко С.Г. Арифметичні та управляючи пристрої цифрових ЕОМ: Навчальний посібник. – К.: ВЕК+, 2008. – 176 с.

3. Жабін В.І., Жуков І.А., Клименко І.А., Ткаченко В.В. Прикладна теорія цифрових автоматів: Навч. посібник. – К.: Вид-во НАУ, 2009. – 364 с.

4. Майоров С.А. и др. Проектирование цифровых вычислительных машин. – М.: Высшая школа, 1972 . – 347 с.

5. Карцев М.А. Арифметика цифрових машин. – М.: Наука, 1969. – 578 с.

6. Самофалов К.Г., Корнейчук В.И., Тарасенко В.П. Цифровые ЭВМ. - Киев: Вища школа, 1989. – 424 с.

2. Лабораторна робота №2

ПРОЕКТУВАННЯ ТА ДОСЛІДЖЕННЯ ПРИСТРОЇВ ДЛЯ МНОЖЕННЯ ЧИСЕЛ

Ціль роботи – вивчити методи реалізації операції множення чисел в прямих кодах, одержати навики в дослідженні операційних пристроїв.

Теоретичні відомості

При множенні чисел у прямих кодах знакові та основні розряди обробляються роздільно. Для визначення знака добутку здійснюють підсумовування за модулем 2 цифр, записаних в знакових розрядах співмножників. Будемо вважати, що множене Y і множник X – правильні двійкові дроби виду $X = x_1 x_2 ... x_n$, $Y = y_1 y_2 ... y_n$, де двійкові розряди x_i, $y_i \in \{0,1\}$. Тоді добуток Z модулів чисел дорівнює

$$Z = YX = Yx_1 2^{-1} + Yx_2 2^{-2} + ... + Yx_i 2^{-i} + ... + Yx_n 2^{-n}. \qquad (2.1)$$

Множення Y і X може бути реалізоване шляхом виконання циклічного процесу, характер якого залежить від конкретної форми виразу (2.1). Один цикл множення складається з додавання чергового часткового добутку, що представляє собою добуток множеного на одну цифру множника, до суми часткових добутків. Розрізняють чотири способи множення.

Перший спосіб множення

Вираз (2.1) можна представити у вигляді

$$Z = YX = ((...((\,0 + Yx_n)2^{-1} + Yx_{n-1})2^{-1} + ... + Yx_i)2^{-1} + ... + Yx_1)2^{-1}\ .$$

Звідси випливає, що отримані суми часткових добутків в i-му циклі ($i=\overline{1,n}$) зводиться до обчислення

$$Z_i = (Z_{i-1} + Yx_{n\text{-}i+1})2^{-1}$$

з початковими значеннями i=1, Z_0=0, причому Z_n=Z=YX.

Множення здійснюється з молодших розрядів множника, сума часткових добутків зсувається вправо, а множене залишається нерухомим.

Другий спосіб множення.

Запишемо (2.1) у вигляді

$$Z = ((...((0 + Y2^{-n}x_n) + Y2^{-n+1}x_{n\text{-}1}) + ... + Y2^{-1}x_1.$$

Очевидно, що процес множення може бути зведений до n-кратного виконання циклу

$$Z_i = Z_{i-1} + Y_i x_{n\text{-}i+1},\ Y_i = 2Y_{i-1},$$

з початковими значеннями i=1, Y_0=$Y2^{-n}$, Z_0=0. Множення здійснюється з молодших розрядів, множене зсувається вліво, а сума часткових добутків залишається нерухомою.

Третій спосіб множення.

Представимо (10.1) у виді

$$Z = ((...((0 + Y2^{-n}x_1)2 + Y2^{-n}x_2)2 + ... + Y2^{-n}x_i)2 + ... + Y2^{-n}x_n.$$

Отже, суму часткових добутків у і-м циклі (i=$\overline{1,n}$) можна одержати по формулі

$$Z_i = 2Z_{i-1} + Y2^{-n}x_i.$$

Початковими значеннями є i=1, Z0=0. Множення здійснюється зі старших розрядів множника, сума часткових добутків зсувається вліво, а множене нерухоме.

Четвертий спосіб множення.

$$Z = ((...((0 + Y2^{-1} x_1) + Y2^{-2} x_2) + ... + Y2^{-i} x_i) + ... + Y2^{-n} x_n.$$

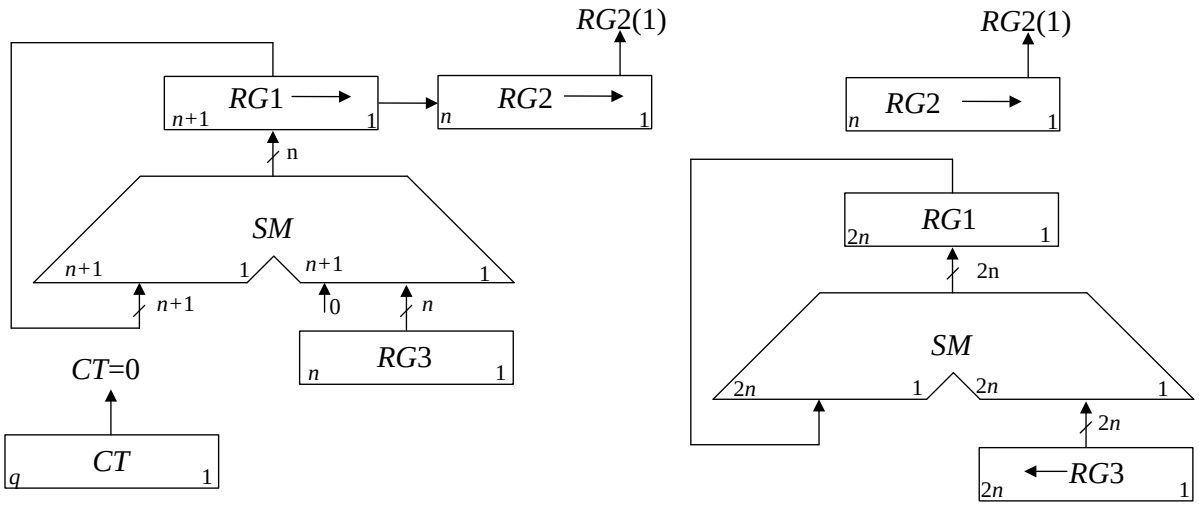
Процес множення може бути зведений до *n*-кратного виконання циклу

$$Z_i = Z_{i-1} + Y_{i\text{-}1} x_i, \;\; Y_i = Y_{i-1} 2^{-1}$$

с початковими значеннями i=1, $Y_0 = Y2^{-1}$, Z_0=0.

Множення виконується зі старших розрядів множника, сума часткових добутків залишається нерухомою, а множене зсувається вправо.

Принцип побудови пристроїв, що реалізують різні способи множення, показаний на рис. 2.1, де *RG*3 – регістр множеного, *RG*1 – регістр добутку, *RG*2 – регістр множника. Цифрами зазначені номери розрядів *SM* і регістрів, а стрілками показаний напрямок зсуву кодів у регістрах.



а *б*

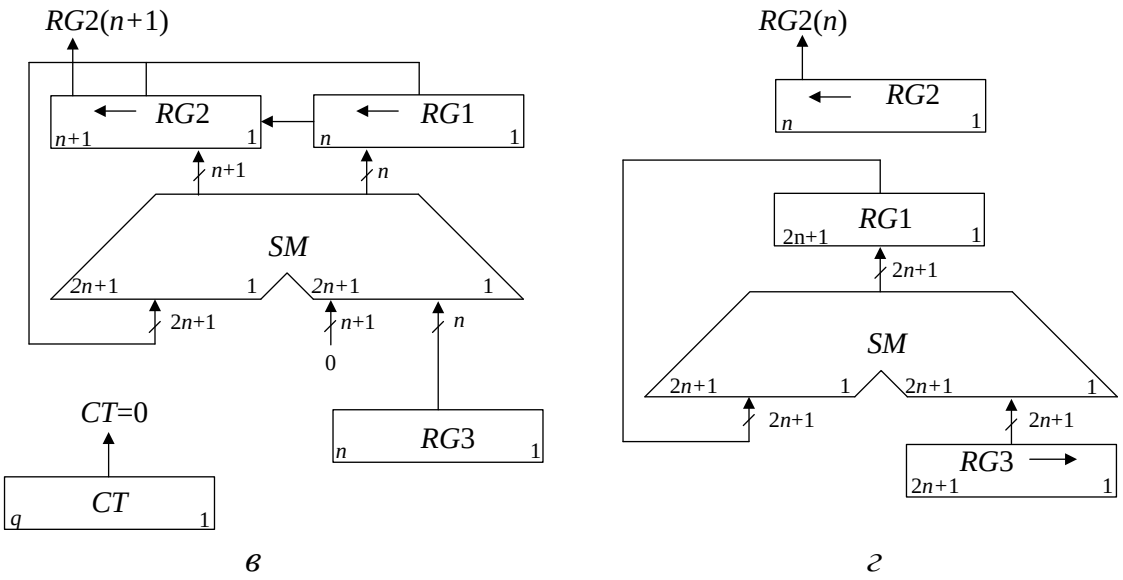


Рис. 2.1. Операційні схеми пристроїв для множення чисел: *а* – перший спосіб; *б* – другий спосіб; *в* – третій спосіб; *г* – четвертий спосіб

Під час множення *першим способом* (рис. 2.1, *а*) в першому такті *i*-го циклу аналізується значення *RG*2[1] – молодшого (*n*-го) розряду регістру *RG*2, в якому знаходиться чергова цифра множника. Вміст *RG*3 додається до суми часткових добутків, що знаходяться в регістрі *RG*1, якщо *RG*2[1]=1, або не додається, якщо *RG*2[1]=0. В другому такті здійснюється правий зсув у регістрах *RG*1 і *RG*2, що еквівалентно множенню їхнього вмісту на 2^{-1}. При зсуві цифра молодшого розряду регістру *RG*1 записується у вивільнюваний старший розряд регістру *RG*2. Після виконання *n* циклів молодші розряди 2*n*-розрядного добутку будуть записані в регістр *RG*2, а старші – у *RG*1.

Максимальний час множення, якщо не застосовуються методи прискорення операції, визначається виразом $t_м = n(t_п + t_з)$, де $t_п$ і $t_з$ – тривалості тактів підсумовування та зсуву відповідно.

Перед початком множення *другим способом* (рис. 2.1, *б*) множник *X* записують в регістр *RG*2, а множене *Y* – в молодші розряди регістру *RG*3 (тобто в регістрі *RG*3 установлюють $Y_0 = Y2^{-n}$). В кожному *i*-му циклі мно-

ження додаванням кодів *RG*3 і *RG*1 управляє цифра *RG*2[1], а в регістрі *RG*3 здійснюється зсув вліво на один розряд, в результаті чого формується величина $Y_i = 2Y_{i-1}$. Оскільки сума часткових добутків в процесі множення нерухома, зсув в регістрі *RG3* можна виконати суміщення в часі з підсумовуванням (як правило, $t_п \geq t_з$). В цьому випадку $t_м = nt_п$. Завершення операції множення визначається за нульовим вмістом регістру *RG*2, що також приводить до збільшення швидкодії, якщо множник ненормалізований.

При множенні *третім способом* (рис. 2.1, в) множник *X* записується в старші розряди *RG*2, при цьому *RG*2[1]=0. Вага молодшого розряду *RG*3 дорівнює 2^{-2n}, тому код в регістрі *RG*3 являє собою значення $Y2^{-n}$. В кожному циклі множення підсування виконується при *RG*2[*n*+1]=1. В регістрах *RG*1 і *RG*2 виконується лівий зсув. В результаті підсумовування вмісту *RG*3 і *RG*1 може виникнути перенос в молодший розряд регістру *RG*2, що реалізується на *SM*. Збільшення довжини *RG*2 на один розряд усуває можливість поширення переносу в розряди множника. Після виконання *n* циклів молодші розряди добутку будуть знаходитися в регістрі *RG*1, а старші – в регістрі *RG*2. Час множення третім способом визначається аналогічно першому способу.

Перед множенням *четвертим способом* (рис. 2.1, г) множник записують в регістр *RG*2, а множене – в старші розряди регістру *RG*3 (тобто в *RG*3 установлюють $Y_0=Y2^{-1}$). В кожнім циклі цифра *RG*2[*n*+1], що знаходиться в старшому розряді регістру *RG*2, управляє підсумовуванням, а в *RG*3 здійснюється правий зсув на один розряд, що еквівалентно множенню вмісту цього регістра на 2^{-1}. Час виконання множення четвертим способом складає $t_м=nt_п$, визначається аналогічно другому способу.

В ЕОМ при роботі із дробовими числами часто потрібно обчислювати не 2*n*, а тільки (*n*+1) цифр добутку й округляти його до n розрядів. В цьому випадку при реалізації другого способу можна зменшити довжину *SM* і

*RG*1, а при реалізації четвертого – зменшити довжину *SM*, *RG*1 і *RG*3. Для того щоб похибка від відкидання молодших розрядів не перевищила половини ваги *n*-го розряду результату, в перерахованих вузлах досить мати тільки по *l* додаткових молодших розрядів, де *l* вибирається з умови

$$l \geq 1 + \log 2(n - l - 1).$$

Операція округлення здійснюється звичайно шляхом додавання одиниці до *n*+1-го розряду результату після коми і відкиданням усіх розрядів, розташованих праворуч від *n*-го.

В операційних пристроях, що реалізують другий і четвертий способи множення, можна без пересилань кодів між регістрами обчислювати суму *K* попарних добудків $\sum_j^K X_j Y_j$, для чого досить черговий результат операції залишати в регістрі *RG*2, який в цьому випадку повинен мати додаткові старші розряди.

При реалізації другого, третього та четвертого способу можна обчислювати функйію $F = XY + G$, де $G = 0, g_1 g_2 ... g_n$ – дробове число в такій самій формі, як і *X* та *Y*. Для цього необхідно перед множенням записати число *G* у регістр *RG*1 з урахуванням ваги розрядів. Наприклад, для третього способу – в молодші розряди регістра *RG*1. Для можливості підключення до входів *RG*1 в різних циклах різні коди (спочатку операнд *G*, а потім в кожному циклі виходи *SM*) необхідно мати мультиплексор.

Найбільш простими є пристрої, що реалізують перший спосіб, а найбільш швидкодіючими – другий і четвертий. Однак другий спосіб не має особливих переваг порівняно з четвертим і, крім того, вимагає більших апаратурних витрат при реалізації множення до *n* розрядів після коми.

Етапи побудови операційних пристроїв для множення чисел.

1. Вивчити алгоритм множення чисел заданим методом.

2. Побудувати операційну схему пристрою.

3. Розробити змістовний (функціональний) мікроалгоритм з використанням операторів присвоєння та зсуву.

4. Виконати логічне моделювання роботи пристрою за допомогою таблиці станів вузлів (регістрів, лічильника) у кожному такті. Перевірити правильність вибору розрядності вузлів на операційній схемі.

5. Побудувати функціональну схему з відображенням управляючих сигналів для всіх вузлів.

6. Розробити структурний мікроалгоритм, в якому змістовні мікрооперації замінюються на сукупність управляючих сигналів, що забезпечують виконання мікрооперацій. Сигнали, що формуються завжди разом, можна подати одним символом. Це зменшує кількість функцій вихідних сигналів при синтезі пристроїв управління.

7. Побудувати і відлагодити схему в системі ПРОГМОЛС-2 (AFDK).

Розглянемо побудову пристрою множення третім способом по наведеній вище схемі.

Операційна схема множення представлена на рис. 2.1, *в*, а функціональний мікроалгоритм подано на рис. 2.2, *а*.

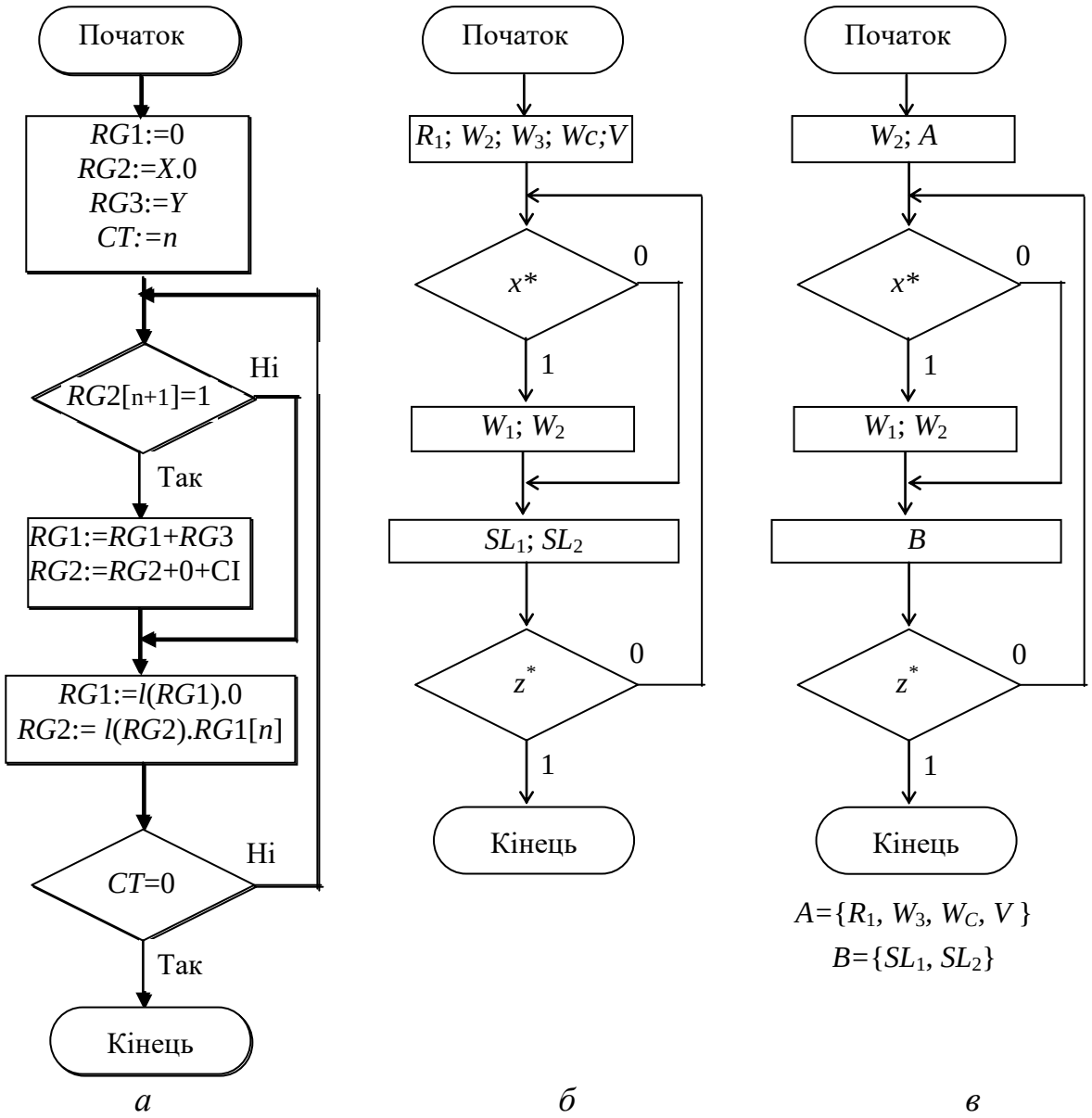


Рис. 2.2. Мікроалгоритми множення чисел за третім способом: **а** – функціональний; б – структурний; в – з обєднаними сигналами

Для додатних операндів $X = 0{,}1011$ і $Y = 0{,}1101$, що мають по чотире основних дробових розрядів, значення станів регістрів і лічильника в кожному такті показано в табл. 2.1. У результаті моделювання перевіряється розрядності вузлів при $n = 4$.

Табл. 2.1. Стани регістрів і лічильника при множенні

№ циклу	*RG*2	*RG*1	*RG*3	*CT*	Мікрооперації
0	**1**011.0	0000	1101	100	*RG*1:=0; *RG*2:=*X*.0; *RG*3:=*Y*; *CT:=n*
1	+00000 10110 **0**1101	+1101 1101 1010	←	011	*RG*1:=*RG*1+*RG*3; *RG*2:=*RG*2+0+CI *RG*1:=*l*(*RG*1).0; *RG*2:= *l*(*RG*2).*RG*1[*n*]
2	**1**1011	0100		010	*RG*1:=*l*(*RG*1).0; *RG*2:= *l*(*RG*2).*RG*1[*n*]
3	+00000 11100 **1**1000	+1101 0001 0010	←	001	*RG*1:=*RG*1+*RG*3; *RG*2:=*RG*2+0+CI *RG*1:=*l*(*RG*1).0; *RG*2:= *l*(*RG*2).*RG*1[*n*]
4	+00000 11000 ***10001***	+1101 1111 ***111***.0	←	000	*RG*1:=*RG*1+*RG*3; *RG*2:=*RG*2+0+CI *RG*1:=*l*(*RG*1).0; *RG*2:= *l*(*RG*2).*RG*1[*n*]

Результат

На функціональній схемі (рис.2.3) відображаються управляючі сигнали і логічні умови x^*, z^*.

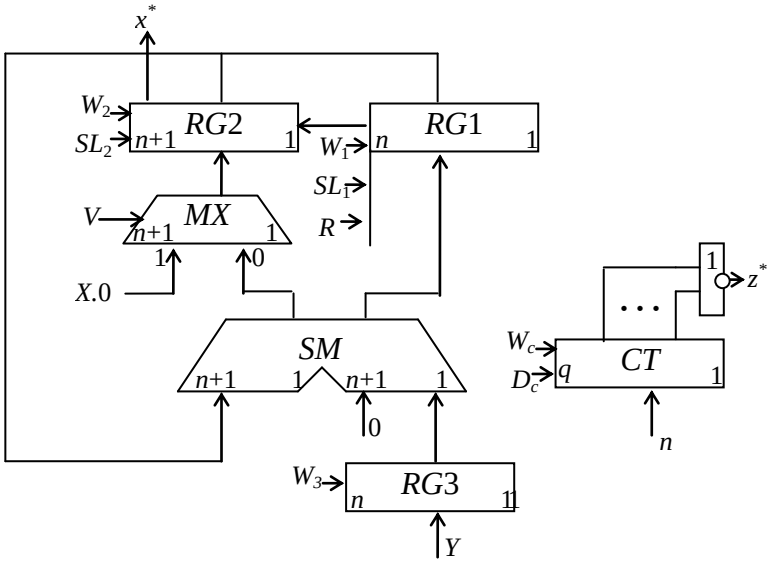


Рис.2.3. Функціональна схема пристрою множення

Занесення інформації в *RG*2 відбувається з двох джерел через мультиплексор *MX* (при V=1 заноситься операнд *X* в старші розряди, а в молодший записується 0. При *V*=0 в *RG*2 заноситься інформація з ви-

ходів *SM*). Інші управляючі сигнали позначаються символами: *W* – занесення даних; *SL* – зсув ліворуч; *D* – декремент; *R* – скид у нуль вмісту регістра.

Враховуючи позначення сигналів на схемі пристрою, будуємо структурний мікроалгоритм та мікроалгоритм із об'єднанням сигналів, що формуються разом (рис. 2.2, *б* і 2.2. ,*в*).

На базі одержаних даних (рис. 2.2, *в* та рис. 2.3) в програмному комплексі ПРОГМОЛС-2 можна набрати та відлагодити схему пристрою.

Підготовка до роботи

1. Варіант завдання визначається молодшими двійковими розрядами $a_6, a_5, a_4, a_3, a_2, a_1$ десяткового номера залікової книжки студента відповідно табл.2.2.

2. Відповідно до запропонованої послідовності етапів синтезу пристроїв множення виконати всі пункти побудови пристрою множення згідно таблиці варіантів (табл. 2.2).

3. Для побудови функціональної схеми можна використовувати комбінаційний суматор, асинхронні регістри і лічильник, що мають окремі входи керування різними мікрооперaціями (*W, SL,SR,D* і т.і.) На схемі повинна бути зазначена розрядність регістрів. Третій спосіб множення використовується для обчислення функції $F = XY + G$, а інші методи – для функції $F = XY$, тобто тільки для множення. Операнд G перед початком циклів множення записується в *RG*1 через *MX*. Для цього виконується окрема мікрооперація, яка повинна бути відображена в мікроалгоритмі.

4. Згідно варіанту завдання (табл. 2.2) треба виконати числовий приклад обчислень у вигляді таблиці станів вузлів в кожному такті. Таблиця станів може використовуватися у якості тесту при налагодженні пристрою.

Табл. 2.2. Таблиця варіантів

a_3	a_2	a_1	**Спосіб множення, розрядність операндів**	**Значення додатних операндів**			**Повна операція**
				X	***Y***	***G***	
0	0	0	1-й, 7	,1 $a_6a_5a_4$ 101	,1001101	-	*F=XY*
0	0	1	2-й, 5	,1 $a_6a_5a_4$ 0	,10011	-	*F=XY*
0	1	0	3-й, 7	,1 $a_6a_5a_4$ 001	,0100111	,1101011	*F=XY+G*
0	1	1	4-й, 6	,1 $a_6a_5a_4$ 01	,110011	-	*F=XY*
1	0	0	1-й, 6	,11 $a_6a_5a_4$ 0	,101111	-	*F=XY*
1	0	1	2-й, 6	,10 $a_6a_5a_4$ 1	,111010	-	*F=XY*
1	1	0	3-й, 6	,10 $a_6a_5a_4$ 1	,101111	,100101	*F=XY+G*
1	1	1	4-й, 5	,10 $a_6a_5a_4$	,11001	-	*F=XY*

Порядок виконання роботи

1. Набрати в комплексі ПРОГМОЛС-2 розроблену функціональну схему пристрою множення. Входами пристрою є інформаційні входи операндів X, Y, G та всі управляючи входи: W, SL, SR і т.і. (рис.2.4). Виходами пристою є логічні умови x^*, z^*.

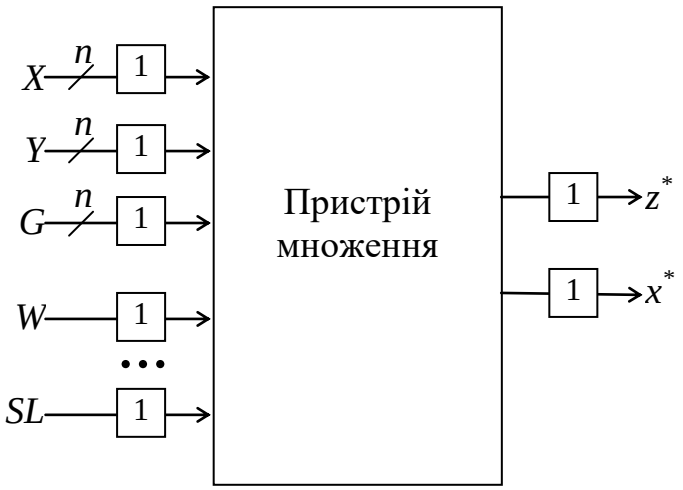


Рис. 2.4. Підготовка схеми для відлагодження

2. Відлагодити схему і виконати підготовлений цифровий приклад. Сигнали подаються мишею вручну безпосередньо на входи пристрою. Послідовність сигналів залежить від логічних умов відповідно схемі мікроалгоритму.

4. Дослідити зазначені викладачем часові параметри схеми.

Зміст звіту

Звіт повинний включати короткі теоретичні відомості, необхідні для виконання лабораторної роботи, всі схеми, формули і таблиці, отримані при виконанні теоретичного завдання і в процесі моделювання схем. Сформулювати висновки по роботі.

Контрольні питання

1. Охарактеризуйте чотири основні методи множення чисел.

2. Як розрахувати розрядність вузлів операційного пристрою?

3. Що таке мікрооперація і мікроалгоритм операції?

4. Охарактеризуйте основні етапи проектування пристроїв множення.

5. Як перейти від функціонального (змістовного) мікроалгоритму до структурного мікроалгоритму?

6. Як визначити необхідну тривалість керуючих сигналів?

7. Як визначити тривалість циклу множення?

8. Порівняйте за часом виконання операції різні способи множення.

9. Які додаткові обчислювальні функції мають різні методи множення.

10. Як перейти від операційної схеми до функціональної?

Література

1. Жабін В.І., Жуков І.А., Клименко І.А., Стіренко С.Г. Арифметичні та управляючі пристрої цифрових ЕОМ: Навч. посібник. К.: ВЕК+, 2008. – 176 с.

2. Жабін В.І., Жуков І.А., Клименко І.А., Ткаченко В.В. Прикладна теорія цифрових автоматів: Навч. посібник. – К.: Вид-во НАУ, 2009. – 364 с.

3. Самофалов К.Г., Корнейчук В.І., Тарасенко В.П., Жабін В.І. Цифрові ЕОМ. Практикум – К.: Вища шк., 1990. – 215 с.

4. Карцев М.А. Арифметика цифрових машин. – М.: Наука, 1969. – 578 с.

3. Лабораторна робота №3
ПРОЕКТУВАННЯ ТА ДОСЛІДЖЕННЯ ПРИСТРОЇВ ДЛЯ ДІЛЕННЯ ЧИСЕЛ

Ціль роботи – вивчити методи ділення чисел в прямих кодах і способи їх апаратурної реалізації, придбати навики в налагодженні та дослідженні операційних пристроїв.

Теоретичні відомості

Існують два основних методи ділення чисел: ділення з відновленням і без відновлення від'ємної остачі. Реалізація цих методів вимагає приблизно однакового обсягу устаткування, але при діленні першим методом потрібно більше часу для виконання операції. Тому метод ділення чисел без відновлення залишку є кращим.

Нехай ділене X и дільник Y є n-розрядними правильними дробами, представленими в прямому коді. В цьому випадку знакові й основні розряди операндів обробляються окремо. Знак результату визначається шляхом підсумовування за модулем 2 цифр, записаних в знакових розрядах.

Алгоритм ділення чисел без відновлення залишку зводиться до виконання наступних дій.

1. Одержати різницю $R_0=X-Y$. Якщо $R_0\geq 0$, то цифра Z_0 частки, що має вагу 1, а при $R_0< 0$ – дорівнює 0. Різниця R_0 є залишком.

2. Подвоїти залишок (одержати $2R_i$).

3. Якщо $2R_i<0$, то додати, а якщо $2R_0\geq 0$, то відняти Y. Якщо знову отриманий залишок $R_{i+1}\geq 0$, то $Z_{i+1}=1$, інакше $Z_{i+1}=0$.

4. Повторити пп. 2 і 3 n-1 раз.

Пункт 2 алгоритму можна замінити пунктом “зменшити в два рази дільник”. Наявність двох інтерпретацій другого пункту дає два основних варіанти реалізації ділення.

Перший спосіб ділення.

При реалізації ділення за першим варіантом здійснюється зсув вліво залишку при нерухомому дільнику. На рис. 3.1 показана можлива побудова пристрою ділення. Чергова остача формується в регістрі *RG*2 (у вихідному стані в цьому регістрі записаний *X*). Виходи *RG*2 підключені до входів суматора *SM* безпосередньо, тобто ланцюги видачі коду з *RG*2 не потрібні. Дільник *Y* знаходиться в регістрі *RG*1. Результат формується в регістрі *RG*3 за (n + 1) циклів. Знак остачі визначається розрядом *RG*2[n+2]. Розряд *RG*3[n+1] використовується для визначення кінця операції, ознакою цього є маркерний нуль на виході розряду. Максимальний час одержання цифри результату визначається виразом $t_Ц = t_Д + t_З$, де $t_Д$ – тривалість виконання мікрооперації додавання/віднімання; $t_З$ – тривалість виконання мікрооперації зсуву. Час для одержання n+1 цифри частки визначається виразом t=(n+1) $t_Ц$.

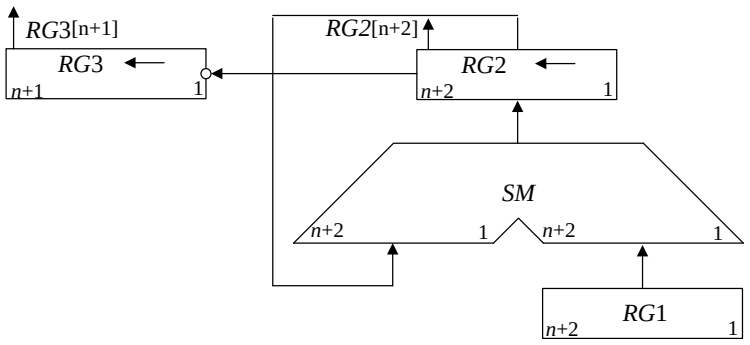


Рис. 3.1. Операційна схема ділення за першим способом із зсувом остачі

Другий спосіб ділення.

При реалізації ділення другим способом (із зсувом дільника) збільшується розрядність регістрів *RG*1, *RG*3 і суматора *SM* (рис. 3.2). В даному випадку процеси додавання/віднімання і зсуву можуть бути суміщені у

часі. Отже, для ділення за другим способом час одержання цифри результату дорівнює $t_Ц = t_Д$. Цифра результату формується на виході переносу суматора $SM(p)$. Загальний час ділення визначається як $t = (n+1)t_Д$.

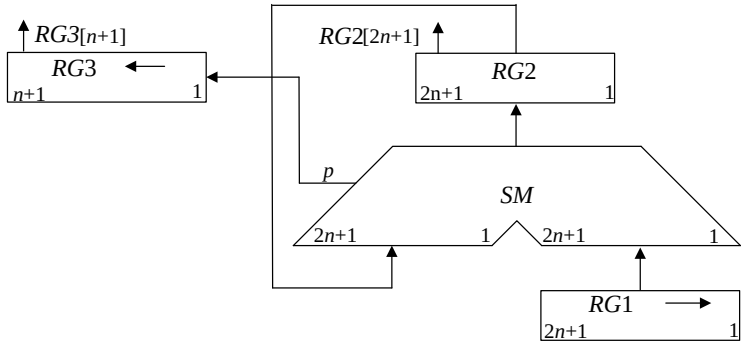


Рис. 3.2. Операційна схема пристрою ділення другим способом із зсувом дільника

При діленні чисел з фіксованою комою повинна бути передбачена можливість фіксації переповнення розрядної сітки. Ознака переповнення формується, якщо в першому циклі цифра результату дорівнює одиниці.

Етапи розробки операційного пристрою для ділення чисел.

1. Вивчити алгоритм ділення чисел заданим методом.
2. Побудувати операційну схему пристрою.
3. Розробити змістовний (функціональний) мікроалгоритм з використанням операторів присвоєння, зсуву тощо.
4. Виконати логічне моделювання роботи пристрою за допомогою таблиці станів регістрів у кожному такті. Перевірити правильність вибору розрядності вузлів на операційній схемі.
5. Побудувати функціональну схему з відображенням управляючих сигналів виконання мікро операцій для всіх вузлів.
6. Розробити структурний мікроалгоритм, в якому змістовні мікрооперації замінюються на сукупність управляючих сигналів, що забезпечують виконання мікрооперацій. Сигнали, що формуються завжди разом,

можна подати одним символом. Це зменшує кількість функцій вихідних сигналів при синтезі пристроїв управління.

7. Побудувати і відлагодити схему в системі ПРОГМОЛС-2 (AFDK).

Змістовний мікроалгоритм ділення за другим способом подано на рис.3.2. Залежно від знакового розряду *RG*2[2*n*+1] до регістра *RG*2 додається або віднімається код із регістра *RG*1. Віднімання забезпечується інверсією коду із *RG*1 і додаванням одиниці (*D*=1) на вхід переносу суматора *SM*. Чергова цифра результату записується в *RG*3 із виходу переносу суматора при зсуві.

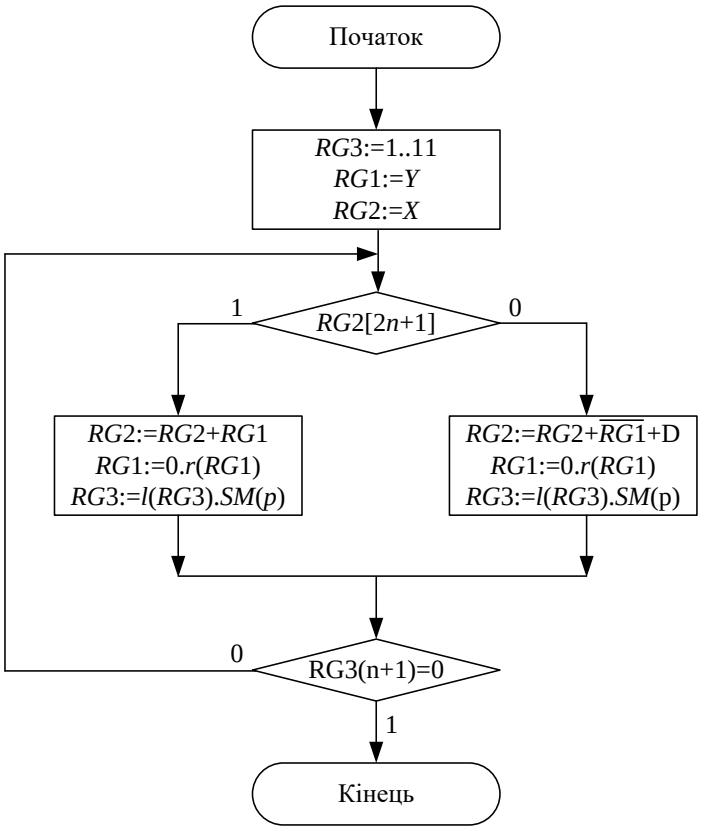


Рис. 3.3. Змістовний мікроалгоритм виконання операції ділення із зсувом дільника

Цифрова діаграма стану вузлів при діленні $Z = Y / X$ двох додатних мантис наведена в табл. 3.1, в якій $Y = 0{,}1001$; $X = 0{,}1100$; $0 < Y, X < 1$; $Y < X$.

При побудові функціональної схеми на базі операційної схеми (рис. 3.1 і рис. 3.2) необхідно мати на виходах *RG*1 перетворювач дільника у

від'ємне число в доповняльному коді для реалізації мікрооперації віднімання.

Перед початком виконання циклів ділення необхідно записати ділене в *RG*2. В процесі ділення в цей регістр передається інформація з виходів суматора *SM*. Забезпечити запис даних в *RG*2 з двох напрямків дозволяє використання мультиплексора.

Табл. 3.1. Стани регістрів при ділення чисел

№ циклу	*RG*3	*RG*2	*RG*1	Мікро-операція
	1 1111	0 10010000	0 11000000	
1	 1 1110	0 10010000 1 01000000 1 11010000	 0 01100000	-Y Зсув
2	1 1110 1 1101	1 11010000 0 01100000 0 00110000	0 01100000 0 00110000	+Y
3	1 1101 1 1011	0 00110000 1 11010000 0 00000000	0 00110000 0 00011000	-Y Зсув
4	1 1011 1 0110	0 00000000 1 11101000 1 11101000	0 00011000 0 00001100	-Y Зсув
5	1 0110 0 1100	1 11101000 0 00001100 1 11110100	0 00001100 0 00000110	+y Зсув

Для визначення кінця операції використовується маркерний нуль. Якщо перед початком обчислень в усі розряди регістру *RG*3 записати одиниці, то перший нуль в цьому розряді після зсуву означає кінець операції. В першому циклі ділення (якщо немає переповнення розрядної сітки) завжди

формується нулева цифра частки. Такий підхід дозволяє спростити пристрій за рахунок усунення лічильника циклів.

Підготовка до роботи

1. Варіант завдання визначається шістьома молодшими двійковими розрядами $a_6, a_5, a_4, a_3, a_2, a_1$ десяткового номера залікової книжки студента відповідно табл. 3.2.

2. Виконати всі запропоновані вище пункти синтезу пристрою ділення згідно таблиці варіантів (табл. 3.2).

3. Для побудови функціональної схеми можна використовувати комбінаційний суматор, асинхронні регістри і лічильник, що мають окремі входи керування різними мікрооперaціями (*W, SL,SR,D* тощо). На схемі повинна бути зазначена розрядність регістрів.

4. Згідно варіанту завдання (табл. 3.2) треба виконати числовий приклад обчислень у вигляді таблиці станів вузлів в кожному такті. Таблиця станів може використовуватися у якості тесту при налагодженні пристрою.

Табл. 3.2. Таблиця варіантів

a_3	a_2	a_1	**Спосіб ділення, розрядність операндів**	**Додатні дробові операнди**	
				X	***Y***
0	0	0	1-й, 7	$,1\,a_6a_5a_4\,101$	,1111101
0	0	1	2-й, 5	$,10\,a_6a_5a_4$	,11011
0	1	0	1-й, 6	$,10\,a_6a_5a_4\,1$	,110111
0	1	1	2-й, 6	$,10\,a_6a_5a_4\,0$	,110011
1	0	0	1-й, 5	$,10\,a_6a_5a_4$	,11011
1	0	1	2-й, 7	$,10\,a_6a_5a_4\,01$	,1110101
1	1	0	1-й, 4	$,1\,a_6a_5a_4$	,1111
1	1	1	2-й, 4	$,0\,a_6a_5a_4$	,1110

Порядок виконання роботи

1. Набрати в комплексі ПРОГМОЛС-2 розроблену функціональну схему пристрою ділення. Входами пристрою є інформаційні входи операндів X, Y та всі управляючи входи: *W, SL, SR,* ... (рис.3.4). Виходами пристрою є логічні умови – старші розряди регістрів *RG*2 і *RG*3 (відповідно x^* і z^*).

2. Відлагодити схему і виконати підготовлений цифровий приклад. Сигнали подаються мишею вручну безпосередньо на входи пристрою. Послідовність сигналів залежить від логічних умов відповідно схемі мікроалгоритму.

3. Дослідити зазначені викладачем часові параметри схеми.

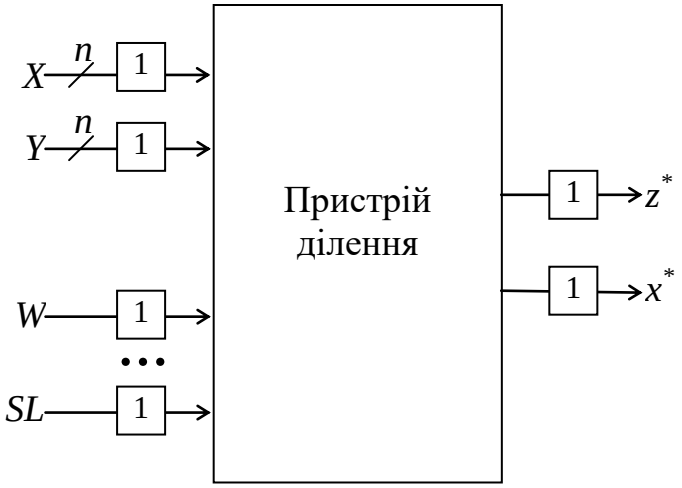


Рис. 3.4. Підготовка схеми для відлагодження

Зміст звіту

Звіт повинний включати короткі теоретичні відомості, необхідні для виконання лабораторної роботи, всі схеми, формули і таблиці, отримані при виконанні теоретичного завдання і в процесі моделювання схеми. Сформулювати висновки по роботі.

Контрольні питання

1. Охарактеризуйте два основні методи ділення чисел.

2. Як розрахувати розрядність вузлів операційного пристрою?

3. Що таке мікрооперація і мікроалгоритм операції?

4. Охарактеризуйте основні етапи проектування пристроїв ділення.

5. Як перейти від функціонального (змістовного) мікроалгоритму до структурного мікроалгоритму?

6. Як визначити необхідну тривалість керуючих сигналів?

7. Як визначити тривалість циклу ділення, часу виконання операції?

8. Порівняйте за часом виконання операції різні способи ділення.

9. В яких пристроях можна суміщувати мікрооперації підсумовування/віднімання і зсуву? Чому це можливо?

10. Чи можна зменшити довжину регістрів операційного пристрою при реалізації ділення чисел за другим варіантом, якщо результат повинний бути представлений q розрядами ($q<n$)?

11. Як перейти від операційної схеми до функціональної?

Література

1. Жабін В.І., Жуков І.А., Клименко І.А., Стіренко С.Г. Арифметичні та управляючі пристрої цифрових ЕОМ: Навч. посібник. К.: ВЕК+, 2008. – 176 с.

2. Жабін В.І., Жуков І.А., Клименко І.А., Ткаченко В.В. Прикладна теорія цифрових автоматів: Навч. посібник. – К.: Вид-во НАУ, 2009. – 364 с.

3. Самофалов К.Г., Корнейчук В.І., Тарасенко В.П., Жабін В.І. Цифрові ЕОМ. Практикум – К.: Вища шк., 1990. – 215 с.

4. Карцев М.А. Арифметика цифрових машин. – М.: Наука, 1969. – 578 с.

4. Лабораторна робота №4

ДОСЛІДЖЕННЯ ОПЕРАЦІЙ ДОДАВАННЯ ТА ВІДНІМАННЯ В ДВІЙКОВО-КОДОВАНИХ СИСТЕМАХ ЧИСЛЕННЯ

Ціль роботи: Оволодіння способами подання десяткових чисел зі знаками. Дослідження операції додавання та віднімання чисел в системах числення з двійково-кодованим поданням цифр.

Теоретичні відомості

Додавання чисел в двійково-кодованих системах числення.

В системах числення з основою $k > 2$ та цифрами $x_i \in \{0,1,...k-1\}$ для подання цифр використовують двійкову систему числення. Для представлення однієї цифри необхідно мати не менше ніж $m =]\log_2(k-1)[$ двійкових розрядів (]..[– функція округлення числа до найближчого цілого). Наприклад, для десяткової системи числення цифри кодуються не менш ніж чотирма двійковими розрядами (тетрадами), хоча двійково-десяткові коди (ДДК) можуть мати і більше розрядів, якщо це дає переваги при виконанні певних операцій (табл. 4.1)

Табл. 4.1. Двійково-десяткові коди

Десяткова цифра	Двійково-десятковий код						
	8432	2421	5421	7421	8421+3	2 із 5	Томсона
0	0000	0000	0000	0000	0011	11000	00000
1	0001	0001	0001	0001	0100	00011	10000
2	0010	0010	0010	0010	0101	00101	11000
3	0011	0011	0011	0011	0110	00110	11100
4	0100	0100	0100	0100	0111	01001	11110
5	0101	1011	1000	0101	1000	01010	11111
6	0110	1100	1001	0110	1001	01100	01111
7	0111	1101	1010	1000	1010	10001	00111
8	1000	1110	1011	1001	1011	10010	00011
9	1001	1111	1100	1010	1100	10100	00001

Наприклад в коді 2 із 5 спрощується дешифрування кодів цифр, а також визначення помилок (зміна будь-якого одного розряду дає заборонену

комбінацію двійкових розрядів). Для коду Томсона лічильник будується на регістрі зсуву з інверсним циклічним ланцюжком, що прискорює мікрооперацію інкремент порівняно з лічильниками, що мають міжрозрядні логічні схеми. Крім того, це також спрощує дешифрування десяткових цифр. Для арифметичних операцій вказані коди не застосовують, бо вони не є адитивними і зваженими.

Адитивними є коди для яких сума кодів двох чисел представляє код суми. В *зважених кодах* кожний розряд в тетраді має постійну вагу.

ДДК з вагами розрядів 8421 називають кодом прямого заміщення. Завдяки адитивності та зваженості зручність цього коду проявляються при машинному переводі з десяткової системи в двійкову і назад, а також при підсумовуванні на звичайних двійкових суматорах завдяки його адитивності (сума кодів двох чисел представляє код суми).

Коди 2421, 5421 і 7421 не є адитивними. Код з вагами 2421 дозволяє простим інвертуванням розрядів одержувати обернений, а при ще й додаванні одиниці – доповняльний код числа. Це зручно для виконання алгебраїчних операцій. В кодах 5421 і 7421 ряд комбінацій має менше одиниць ніж код 8421. Це може зменшити споживання енергії в динамічній елементній базі.

ДДК з надлишком 3 не є зваженим, що ускладнює перетворення чисел в системи з різною основою, але він дозволяє автоматично формувати перенос в старшу тетраду при підсумуванні чисел на звичайних суматорах.

Способи додавання чисел з основою $k=10$ *на базі двійкових суматорів.*

Алгебраїчні операції з десятковими числами, як і з двійковими, виконуються на суматорах з використанням обернених або доповняльних кодів. Для одержання оберненого коду необхідно кожен розряд числа замінити на цифру, що є доповненням до 9. Наприклад, 8 замінюють на 1, 3 – на 6, 5

– на 4 і т.і. Додавання одиниці в молодший розряд переводить обернений код в доповняльний. Додатні числа в знакових розрядах мають 0, а від'ємні – 1.

Наприклад, операції 542+223=765, 542-223=319, 223-542=-319 в доповняльних кодах виконуються наступним чином:

0.542	0.542	0.223
+0.223	+1.777	+1.458
0.765	0.319	1.681
Результат=+765	Результат=+319	Результат=-319

Віднімання $Z = X - Y$ замінюється на додавання $Z = X + (-Y)$, що дозволяє виконувати операції на суматорах. Використання різних адитивних ДДК для подання десяткових цифр потребує корекції при підсумуванні розрядів. В коді 8421 корекція в десяткових розрядах виконується, якщо після підсумування тетрад виникає перенос в старшу тетраду або результат в тетраді стає більше 9. Необхідна корекція в тетрадах «+6», що пояснюється наступним. Одиниця, що залишає тетраду, повинна забрати з неї $k = 10$ одиниць, а фактично забирає 16, тобто подвійну вагу старшого двійкового розряду тетради (для коду 8421 ця вага дорівнює 8). Тому для компенсації до тетради додається число +6 (0110_2) з розповсюдженням переносу в старшу тетраду . На рис. 4.1 показано варіанти корекції при додаванні 2-розрядних двійково-десяткових чисел.

	$38_{2\text{-}10}$=0011	1000		$29_{2\text{-}10}$=0010	1001
+	$16_{2\text{-}10}$=0001	0110	+	$58_{2\text{-}10}$=0101	1000
	0100	1110		1000	0001
	Корекція +0000	0110		Корекція +0000	0110
а)	$54_{2\text{-}10}$=0101		*б)*	$87_{2\text{-}10}$=1000	
	0100			0111	

Рис. 4.1. Корекція результату: а – заборонена цифра 14 (1110_2) в молодшому розряді; б – є перенос в старшу тетраду

Двійково-десяткові суматори.

Двійково-десяткові суматори в кожному десятковому розряді повинні реалізувати п'ять перемикальних функцій (рис. 4.2). Чотири з них відповідають двійково-кодованій десятковій сумі $S = s_4 s_3 s_2 s_1$ і одна - переносу в старший десятковий розряд P_4. Ці функції залежать від десяткових цифр доданків $X = x_4 x_3 x_2 x_1$ і $Y = y_4 y_3 y_2 y_1$, а також переносу з молодшої тетради P_0, тобто від 9 аргументів. Нормальні форми функцій дуже громіздкі і погано мінімізуються. Тому підсумування ДДК доцільно виконувати відповідно до схеми на рис. 4.2.

На першому етапі на двійковому суматорі підсумовують ДДК десяткових цифр за правилами двійковій арифметики. Потім на другому етапі за допомогою ще одного суматора роблять корекцію отриманого результату шляхом додавання або віднімання деякої константи, що визначається комбінаційною схемою КС, а також виділяють десятковий перенос в старшу тетраду. ДДК повинен мати властивості адитивності.

Такою властивістю володіють ДДК 8421 і 8421+Δ, де – Δ ціле число, що назване надлишком. Якщо використовується ДДК, що не володіє властивостями адитивності, то цифри доданків слід перед підсумовуванням перетворити в адитивний ДДК.

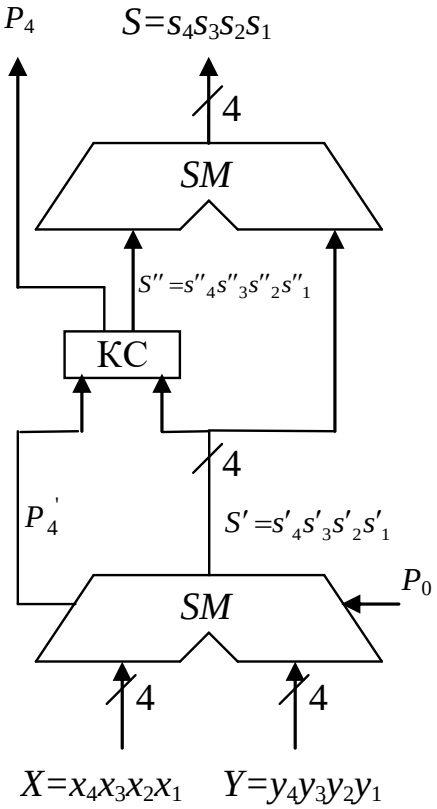


Рис. 4.2. Структура одного розряду десяткового суматора

Схема корекції і виділення переносу може бути визначена шляхом порівняння P'_4 і $S'=s'_4s'_3s'_2s'_1$, отриманих при підсумовуванні цифр доданків, і необхідного результату, тобто P_4 і $S=s_4s_3s_2s_1$. Нехай в якості ДДК використовується код 8421. Тоді стани вихідних сигналів за схемою на рис. 4.2 можна описати за допомогою табл. 4.2, де $\Sigma_Д$ – сума в десятковому вигляді.

З таблиці 4.2 видно, що в залежності від суми, отриманої на першому етапі, корекція результату для ДДК 8421 складається в додаванні 0 або 6. Вважаючи функції $P''_{4,}s''_4,s''_3,s''_2,s''_1$ частково визначеними функціями 5-ти аргументів $P'_{4,}s'_4,s'_3,s'_2,s'_1$, можна після мінімізації знайти: $s''_4=s''_1=0; s''_3=s''_2=P''_4=P'_4\vee s'_4\cdot s'_2\vee s'_4\cdot s'_3$.

Знайдені функції у якості одного з доданків подаються на суматор другого ярусу, на виходах якого формується правильний результат, тобто

десяткова цифра. З десяткових однорозрядних суматорів, показаних на рис. 4.1, складається суматор на необхідну кількість розрядів. Для цього вихідний перенос i -го розряду подається на вхідний перенос $i+1$-го розряду.

Табл. 4.2. Таблиця істинності комбінаційного двійково-десяткового суматора

$\Sigma_д$	Сума до корекції					Сума після корекції					Код корекція			
	P'_4	S'_4	S'_3	S'_2	S'_1	P_4	S_4	S_3	S_1	S_i	S''_4	S''_3	S''_2	S'_1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1	0	0	0	0
2	0	0	0	1	0	0	0	0	1	0	0	0	0	0
3	0	0	0	1	1	0	0	0	1	1	0	0	0	0
4	0	0	1	0	0	0	0	1	0	0	0	0	0	0
5	0	0	1	0	1	0	0	1	0	1	0	0	0	0
6	0	0	1	1	0	0	0	1	1	0	0	0	0	0
7	0	0	1	1	1	0	0	1	1	1	0	0	0	0
8	0	1	0	0	0	0	1	0	0	0	0	0	0	0
9	0	1	0	0	1	0	1	0	0	1	0	0	0	0
10	0	1	0	1	0	1	0	0	0	0	0	1	1	0
11	0	1	0	1	1	1	0	0	0	1	0	1	1	0
12	0	1	1	0	0	1	0	0	1	0	0	1	1	0
13	0	1	1	0	1	1	0	0	1	1	0	1	1	0
14	0	1	1	1	0	1	0	1	0	0	0	1	1	0
15	0	1	1	1	1	1	0	1	0	1	0	1	1	0
16	1	0	0	0	0	1	0	1	1	0	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1	0	1	1	0
18	1	0	0	1	0	1	1	0	0	0	0	1	1	0
19	1	0	0	1	1	1	1	0	0	1	0	1	1	0

Знайдені функції у якості одного з доданків подаються на суматор другого ярусу, на виходах якого формується правильний результат, тобто десяткова цифра. З десяткових однорозрядних суматорів, показаних на рис. 4.2, складається суматор на необхідну кількість розрядів. Для цього вихідний перенос i -го розряду подається на вхідний перенос $i+1$-го розряду.

Підготовка до роботи

1. Для визначення варіанту завдання необхідно перевести повний десятковий номер залікової книжки студента в двійкову систему числення і виділити сім молодших розрядів $a_7,a_6,a_5,a_4,a_3,a_2,a_1$.

2. Відповідно з вихідними даними (табл. 4.3) подати таблицю кодування десяткових цифр 0,1,…,9 в ДДК і таблицю істинності однорозрядного комбінаційного двійково-десяткового суматора з використанням заданого ДДК (по аналогії з табл. 4.2). Якщо ДДК не має властивості адитивності, необхідно синтезувати схеми перетворювачів заданого ДДК в адитивний код (наприклад, код 5421 в код 8421).

3. На основі одержаної таблиці істинності виконати синтез комбінаційної схеми формування корекції (див. КС на рис. 4.2). На двійкових суматорах та заданих логічних елементах (табл. 4.3) побудувати функціональну схему одного розряду двійково-десяткового суматора.

4. Подати один розряд синтезованого суматора у вигляді прямокутника і побудувати на їх основі структурну схему 4-розрядного десяткового суматора. Записати приклади додавання $Z = X + Y$ і віднімання у формі $Z = X + (-Y)$ та $Z = Y + (-X)$ 4-розрядних десяткових чисел на розробленому суматорі. Показати необхідну корекцію в розрядах. Для подання чисел використати доповняльний код. Значення операндів подано в табл. 4.3.

Виконання роботи

1. Використовуючи моделюючу систему ПРОГМОЛС 2.0 побудувати і налагодити розроблений однорозрядний суматор.

2. Виконати по два приклади додавання цифр, що потребують і не потребують корекції результату в тетраді. Відмітити результати у звіті.

3. В режимі синхронного моделювання визначити максимальний час підсумування розрядів.

Табл. 4.3. Вихідні дані для побудови суматора.

$a_7a_6a_5$	ДДК	$a_4a_3a_2$	Логічні елементи	$a_3a_2a_1$	Операнди	
					X	***Y***
000	8421+1	000	2І, 2АБО, НЕ	000	3174	-3442
001	8421+2	001	2І, 3АБО, НЕ	001	-3427	3626
010	8421+3	010	3І, 2АБО, НЕ	010	-5473	3672
011	8421+4	011	3І, 3АБО, НЕ	011	1234	-7333
100	8421+5	100	2АБО-НЕ	100	-5136	4437
101	2421	101	2І-НЕ	101	-4512	1312
110	5421	110	3АБО-НЕ	110	5484	-3445
111	7421	111	3І-НЕ	111	3342	-3912

Зміст звіту

Звіт з лабораторної роботи повинен включати короткі теоретичні відомості, необхідні для виконання лабораторної роботи, схеми, таблиці та діаграми, отримані при виконанні теоретичного завдання, а також у процесі моделювання схем, висновки за результатами досліджень.

Контрольні питання

1. В якій формі у ЕОМ подаються десяткові числа?

2. Як визначити кількість розрядів двійкового і двійково-десяткового числа з однаковим кількісним еквівалентом?

3. Поясніть, коли при додаванні чисел необхідна корекція результату.

4. Як визначити необхідну корекцію при додаванні двійково-десяткових чисел.

5. Наведіть склад апаратури для побудови двійково-десяткового суматора.

6. Яким вимогам повинні задовольняти ДДК, що використовуються у суматорі ?

7. У чому сутність властивості адитивності ДДК і до чого може привести відсутність такої властивості у ДДК?

8. У чому сутність властивості зваженості ДДК і до чого може привести відсутність такої властивості у ДДК?

9. Наведіть приклади ДДК, що володіють і не володіють властивістю адитивності.

10. Наведіть приклади ДДК, що володіють і не володіють властивістю зваженості.

11. Наведіть приклад додавання 4-розрядних двійково-десяткових чисел із знаками в заданому ДДК.

Литература

1. Карцев М.А. Арифметика цифрових машин. – М.: Наука, 1969. – 578 с.

2. Самофалов К.Г., Корнейчук В.И., Тарасенко В.П., Жабин В.И. Цифровые ЭВМ. Практикум – К.: Вища шк., 1990. – 215 с.

3. Жабін В.І., Жуков І.А., Клименко І.А., Стиренко С.Г. Арифметичні та управляючи пристрої цифрових ЕОМ: Навчальний посібник. – К.: ВЕК+, 2008. – 176 с.

4. Жабін В.І., Жуков І.А., Клименко І.А., Ткаченко В.В. Прикладна теорія цифрових автоматів: Навч. посібник. – К.: Вид-во НАУ, 2009. – 364 с.

5. Самофалов К.Г., Корнейчук В.И., Тарасенко В.П. Цифровые ЭВМ. - Киев: Вища школа, 1989. – 424 с.

Додаток А

ПРОГРАМНИЙ КОМПЛЕКС ДЛЯ ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

Програмний комплекс ПРОГМОЛС 2.0 (**ПРОГ**рама **МО**делювання **Л**огічних **С**хем) призначений для моделювання процесів у комбінаційних і послідовнісних схемах. Він дозволяє створювати і редагувати логічні схеми, здійснювати моделювання у синхронному (без урахування затримок сигналів в елементах схеми) і в асинхронному (з урахуванням затримок) режимах, а також зберігати отримані моделі. На базі цього комплексу виконуються лаборкторні роботи 1 і 2. Вигляд моделі показано на рис. А.1.

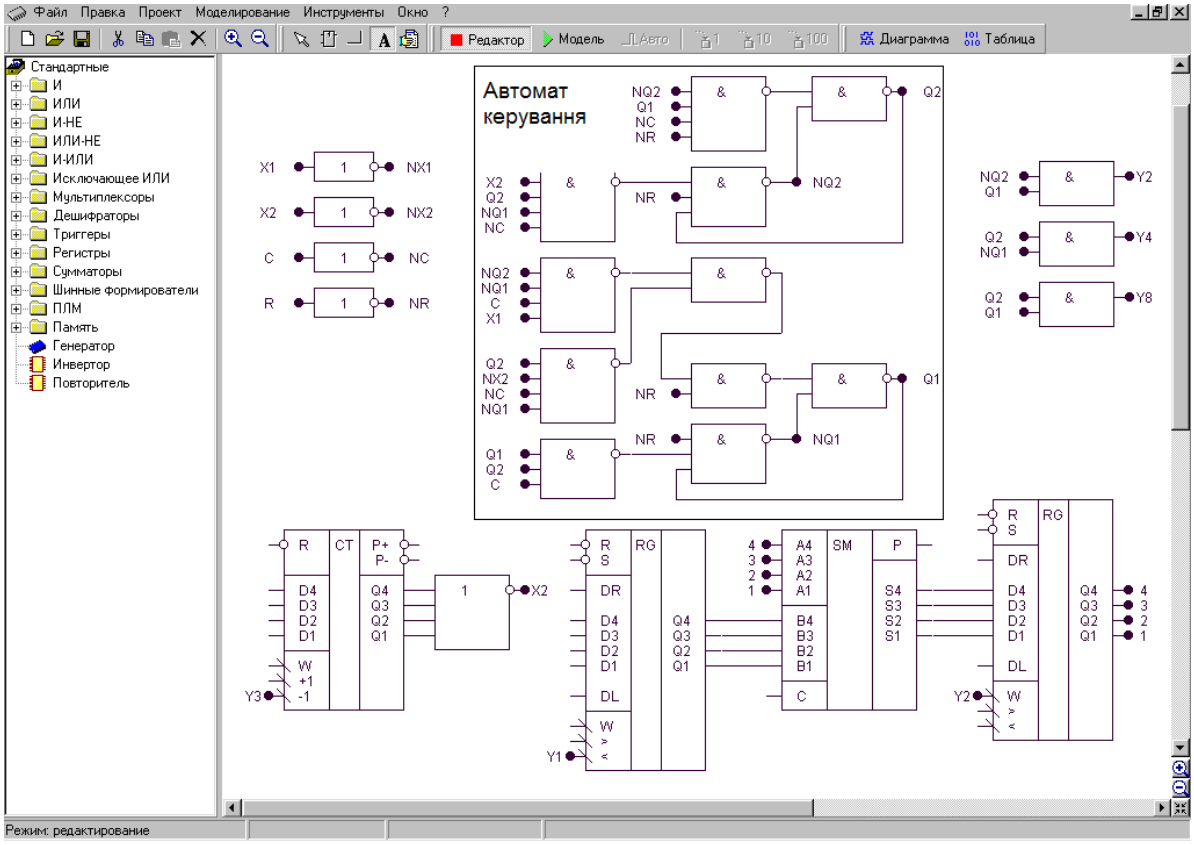


Рис. А.1. Зображення моделі операційного пристрою з автоматом керування

Для роботи з програмою використовують систему ієрархічних меню, що містить наступні розділи:

- ***файл;***
- ***виправлення;***
- ***проект;***
- ***моделювання;***
- ***інструменти;***
- ***вікно;***
- ***допомога.***

Комплекс включає систему підказок, що полегшує роботу в різних режимах моделювання. Побудова та елементи керування для кожного розділу пояснюються нижче.

Для дослідження схем у загальному випадку необхідно виконати послідовність дій.

1. Створити за допомогою редактора логічну схему на екрані дисплея.
2. Позначити на схемі вхідні і вихідні змінні.
3. Створити заголовок (перелічити вхідні і вихідні змінні) і сформувати послідовність вхідних наборів в таблиці істинності.
4. Задати необхідні величини затримок сигналів для елементів схеми.
5. Встановити початковий стан схеми.
6. Перейти до режиму моделювання схеми.