УДК 004.31
ББК 3 973.20-047я7
А 817

Рецензенти:
Н.І.Алішов, д-р техн. наук, с.н.с.
(Інститут кібернетики ім. В.М. Глушкова НАН України)
І.А.Дичка, д-р техн. наук, проф.
(Національний технічний університет України "КПІ")
С.Ф.Теленик, д-р техн. наук, проф.
(Національний технічний університет України "КПІ")

Гриф надано Міністерством освіти і науки України
(Лист № 1.4/18-Г-2426 від 29.12.2007)

Видання друкується за рішенням Вченої ради
Інституту комп'ютерних технологій НАУ
(Протокол № 10 від 14.11.2007)

Жабін В.І., Жуков І.А., Клименко І.А., Стіренко С.Г. –
А 817 Арифметичні та управляючі пристрої цифрових ЕОМ: Навчальний посібник. – К.:ВЕК +, 2008. – 176 с.

ISBN 966–7140–11–3

Навчальний посібник присвячений питанням реалізації арифметичних операцій в цифрових ЕОМ. Розглянуті питання побудови арифметичних пристроїв різних типів та засобів управління виконанням операцій. Запропоновані завдання та надані рекомендації по організації практичних і лабораторних занять, подані приклади побудові функціональних та принципових електричних схем.

Посібник призначений для студентів навчального напряму "Комп'ютерна інженерія". Може бути корисним спеціалістам, що працюють в області проектування цифрових систем.

ISBN 966–7140–11–3

ЗМІСТ

ВСТУП

В навчальному посібнику узагальнені матеріали методичних розробок, які виконувалися авторами в процесі викладання курсів схемотехнічного напрямку на кафедрі комп'ютерних систем та мереж Інституту комп'ютерних технологій Національного авіаційного університету та кафедри обчислювальної техніки Національного технічного університету України "КПІ" для студентів навчального напряму "Комп'ютерна інженерія".

Матеріал навчального посібника присвячений вивченню принципів організації та дослідженню арифметико-логічних і управляючих пристроїв, а також побудові функціональних та принципових електричних схем цифрових ЕОМ.

Крім необхідного теоретичного матеріалу, в посібнику приведені завдання та рекомендації до виконання лабораторних робіт. До лабораторних робіт надані теоретичні відомості, необхідні для виконання кожної роботи, приклади проектування, рекомендації до виконання завдання та саме завдання. До кожної лабораторної роботи надаються контрольні питання що застосовуються для контролю знань за відповідною тематикою.

Виконання лабораторних робіт дозволяє розширити і закріпити теоретичні знання з дисципліни, опанувати навички проектування і дослідження арифметичних та управляючих пристроїв цифрових ЕОМ. Кожній лабораторній роботі повинна передувати самостійна підготовка студентів, в процесі якої вони докладно вивчають опис лабораторної роботи, відповідні розділи посібника, конспекту лекцій та літературні джерела. В процесі підготовки складається звіт про лабораторну роботу, в якому повинні бути відображені всі пункти теоретичного завдання, а також заготовлені для виконання експериментальної частини лабораторної роботи таблиці, осі для часових діаграм і таке інше. Перед початком лабораторної роботи результати підготовки перевіряються викладачем. За цим студент повинен представити заготовлений звіт і

відповісти на контрольні питання. Перед початком наступного заняття в лабораторії студент представляє викладачеві цілком оформлений звіт за попередньою роботою. Звіт повинен містити короткі теоретичні відомості, необхідні для виконання завдання, відповіді на контрольні питання, усі схеми, формули, таблиці, діаграми, графіки, отримані при виконанні завдання та в процесі експериментального дослідження схем, а також висновки за роботою. Залік за виконання лабораторної роботи студент одержує після співбесіди за тематикою виконаної роботи.

У посібнику надані завдання до виконання практичних та розрахунково-графічних робіт, а також задачі до самостійного розв'язування на базі яких складаються тести до модульних контролів.

Теоретичний матеріал, зміст лабораторних робіт, практичних завдань та розрахунково-графічної роботи безпосередньо відповідають навчальному плану дисципліни "Цифрові ЕОМ". Посібник може бути корисним при вивченні курсів "Комп'ютерна схемотехніка", "Архітектура комп'ютерів", "Проектування комп'ютерних систем" та інших курсів схемотехнічного напрямку.

Автори посібника вдячні рецензентам: провідному науковому співробітнику Інституту кібернетики ім. В.М. Глушкова НАН України, доктору технічних наук, старшому науковому співробітнику Алішову Н.І., професору кафедри спеціалізованих комп'ютерних систем Національного технічного університету України "Київський політехнічний інститут", доктору технічних наук, професору Дичці І.А., завідувачу кафедри автоматики та управління в технічних системах Національного технічного університету України "Київський політехнічний інститут", доктору технічних наук, професору Теленику С.Ф. за слушні зауваження.

ПЕРЕЛІК СКОРОЧЕНЬ

АЛБ – Арифметико-логічний блок
АЛП – Арифметико-логічний пристрій
БМУ – Блок мікропрограмного управління
БУ – Блок управління
ВМ – Вертикальне мікропрограмування
ГМ – Горизонтальне мікропрограмування
ДК – Доповнювальний код
ЕОМ – Електронно-обчислювальні машини
ЗК – Зворотний код
МА – Мікроалгоритм
МК – Мікрокоманда
МО – Мікрооперація
МП – Мікропрограма
НОЗП – Надоперативний запам'ятовуючий пристрій
ОПр – Операційний пристрій
ОП – Основна пам'ять
ОС – Операційна схема
ОТ – Обчислювальна техніка
ПЗП – Постійний запам'ятовуючий пристрій
ПК – Прямий код
ПМК – Пам'ять мікрокоманд
ПЛМ – Програмовані логічні матриці
УГП – Умовне графічне позначення
УП – Управляючий пристрій
УС – Управляючий сигнал

РОЗДІЛ 1

Основи цифрової обробки інформації в аріфметичних пристроях ЕОМ

1.1. Кодування чисел у ЕОМ

В ЕОМ доцільно подавати знаки чисел за допомогою тих самих символів, що застосовуються для запису самого числа в k-й системі числення. Для цього використовують додатковий розряд, названий знаковим, який розташовується ліворуч від старшого розряду числа.

В ЕОМ для виконання операцій з числами, що мають знаки, використовують спеціальні коди:

- прямий код,
- зворотний код,
- доповнювальний код.

Розглянемо прямий, зворотний та доповнювальний коди для двійкової системи числення з цифрами {0,1}, саме яка має найбільше поширення в ОТ.

При запису числа вважаємо, що число має n-розрядну цілу частину і k-розрядну дробову частину. Крім цього, до числа додається ще знаковий розряд. Відповідна *розрядна сітка* зображена на рис. 1.1.

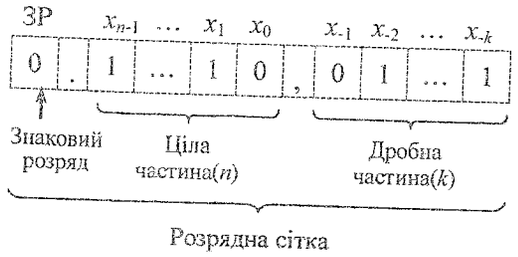


Рис. 1.1. Приклад запису двійкового числа зі знаком

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

Подання числа X у *прямому коді* визначається виразом:

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

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 A=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

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

Прямий код застосовується для зберігання чисел в пам'яті комп'ютера. Для операцій додавання і віднімання ПК не використовується.

Під час перетворення від'ємного числа в *зворотний 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 ЗК збігається із числом у ПК, тобто основні розряди не інвертуються, у знаковий розряд записується 0.

Формула перетворення чисел у зворотний код має вигляд:

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

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

Виконання завдання

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

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

Виконання завдання

$$[A]_{зк} = 0.101011; [B]_{зк} = 1.1000101.$$

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

Виконання завдання

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

Під час перетворення від'ємного числа в *доповнювальний 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:

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

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

Виконання завдання

$[A]_{пк}$=1 . 011 , 100	$[B]_{пк}$=0 . 010 , 010
$[A]_{зк}$=1 . 100 , 011	$[B]_{зк}$=0 . 010 , 010
+ 1	$[B]_{дк}$=0 . 010 , 010
$[A]_{дк}$=1 . 100 , 100	

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

Виконання завдання

$[A]_{пк}$=1 , 011100	$[B]_{пк}$=0 , 010010
$[A]_{зк}$=1 , 100011	$[B]_{зк}$=0 , 010010
+ 1	$[B]_{дк}$=0 , 010010
$[A]_{дк}$=1 , 100100	

1.2. Форми подання чисел у ЕОМ

В ОТ переважно використовуються дві форми подання чисел:

- подання чисел із фіксованою комою;
- подання чисел із плаваючою комою.

Будь-яке число X у позиційній системі числення з основою k можна записати як

$$X = k^P M,$$

де M – *мантиса* числа X, а P – *порядок* числа X.

Якщо порядок фіксований для всіх чисел, то говорять, що число подане у *формі з фіксованою комою*. В цьому випадку числа задаються тільки одним об'єктом – мантисою.

За подання чисел у формі з фіксованою комою положення коми залишається незмінним для всіх чисел, з якими оперує машина, тобто спеціальні розряди для коми не виділяються. З метою спрощення обчислення масштабних коефіцієнтів кому звичайно фіксують перед старшим розрядом мантиси або після молодшого розряду.

За фіксації коми перед старшим розрядом мантиси ЕОМ оперує з числами, які менше одиниці. Якщо число розрядів для запису мантис становить n, то мінімальне (за абсолютною величиною) число, що може бути подане в машині, дорівнює k^{-n}, а максимальне дорівнює $(1–k^{-n})$.

У другому випадку, за фіксації коми після молодшого розряду, в машині виконуються операції над цілими числами, абсолютні величини яких перебувають у межах від 0 до $(k^{n}–1)$.

Розрядна сітка для подання чисел у ЕОМ із фіксованою комою складається з двох частин: один розряд для подання знака, інші розряди для подання мантиси.

Якщо кожне число задається парою P і M, то таке подання називають формою з *плаваючої комою*.

При поданні чисел у формі з плаваючою комою порядок P може бути додатним або від'ємним цілим числом. Мантиса ж у більшості випадків є додатним або від'ємним правильним дробом, причому

$$k^{-1} \leq M \leq 1. \qquad (1.1)$$

Це означає, що старший розряд модуля мантиси завжди дорівнює 1 (мантиса нормалізована). Якщо в ЕОМ для запису порядку використається m розрядів, а для запису мантиси – n розрядів, то в цій машині може бути подане наступне максимальне (за абсолютною величиною) число

$$k^{k^m-1}(1-k^{-n}) = k^{k^m-1} - k^{k^m-n-1}.$$

У свою чергу, мінімальне за абсолютною величиною число, що відрізняється від нуля, у такій машині дорівнює

$$k^{-k^m+1}k^{-1} = k^{-k^m}.$$

Крім зазначених вище розрядів, для подання порядку й мантиси, у розрядній сітці ЕОМ із плаваючою комою є також розряди для подання знаків порядку й мантиси (рис. 1.2).

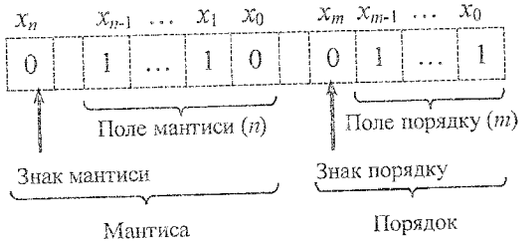


Рис. 1.2. Приклад запису числа у формі з плаваючою комою

Приклад 1.8. Записати у формі із плаваючою комою (m=n=4) двійкові доповнювальні коди чисел:

$$X = 9_{(10)},\ Y = -9_{(10)},\ Z = -9/256_{(10)}.$$

Виконання завдання

$$P_{(X)} = 0\ ,\ 0100 \qquad M_{(X)} = 0\ ,\ 1001$$

$$P_{(Y)} = 0\ ,\ 0100 \qquad M_{(Y)} = 1\ ,\ 0111$$

$$P_{(Z)} = 1\ ,\ 1100 \qquad M_{(Z)} = 1\ ,\ 0111$$

У ЕОМ, де використається подання чисел у формі з плаваючою комою, арифметичні операції виконуються як над мантисами чисел, так і над їхніми порядками. Часто також необхідно виконувати операцію нормалізації чисел, сутність якої складається у виконанні умови (1.1). Тому машини з плаваючою комою є більш складним і менш швидкодіючим, ніж машини з фіксованою комою. З іншого боку, у машинах з фіксованою комою через масштабування виникають труднощі при програмуванні.

Для універсальних комп'ютерів розроблений стандарт *ANSI/IEEE* 754-1985 на подання чисел із плаваючою комою. В цьому стандарті визначені два основні базові формати: короткий та довгий. У

кожному форматі використовують схований старший розряд мантиси з вагою 2^{-1}, який завжди дорівнює 1 для додатних чисел.

Короткий формат (рис. 1.3, *а*) має 32 розряди, з яких 8 розрядів призначені для подання порядку, 23 розряди – для подання мантис і один розряд – для знака мантиси. Спеціального розряду для знаку порядку немає. Вводиться *зміщений порядок:* $P_{зм}=P-128$. Таким чином, від'ємні порядки P будуть подані зміщеними порядками $P_{зм}$, меншими за 128, а додатні порядки – більшими за 128. Введення зміщеного порядку дозволяє звести операції над порядками до арифметичної дії над цілими додатними числами.

Отже, максимальний додатний порядок числа в короткому форматі дорівнює

$$11111111_{(2)} - 10000000_{(2)} = 01111111_{(2)} = 127_{(10)},$$

а максимальний від'ємний порядок

$$00000000_{(2)} - 1000000_{(2)} = -128_{(10)}.$$

ЗМ	2^7	2^6	…	2^0	2^{-2}	2^{-3}	…	2^{-24}
Знак мантиси	Зміщений порядок (8 розрядів)				Мантиса (23 розряди)			

а

ЗМ	2^{10}	2^9	…	2^0	2^{-2}	2^{-3}	…	2^{-53}
Знак мантиси	Зміщений порядок (11 розрядів)				Мантиса (52 розряди)			

б

Рис. 1.3. Формати чисел із плаваючою комою:
а – короткий формат; *б* – довгий формат.

Довгий формат, що використовується для обчислень із підвищеною точністю, має 64 розряди. Для порядку виділяється 11 розрядів, а для мантиси – 52 розряди.

Крім розглянутих вище найбільш поширених форм подання чисел у спеціалізованих ЕОМ можуть використатися також форми, що одержані шляхом певного функціонального перетворення чисел. Прикладом такої форми є логарифмічна, коли числа представляються їхніми

логарифмами по деякій основі. Це дає можливість замінити операції множення й ділення чисел операціями додавання й вирахування їхніх логарифмів.

1.3. 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 порізному. Окрім того, операція віднімання повинна здійснюватися не на суматорі, а на спеціальній схемі.

З використанням зворотних та доповнювальних кодів операції додавання і віднімання можна виконувати за допомогою тільки суматорів, на яких оброблюються як основні, так й знакові розряди. В цьому випадку операція віднімання заміняється операцією додавання з числом, що має протилежний знак. Наприклад, операція $S = A - B$ виконується як $S = A + (-B)$.

Для виконання операції віднімання чисел в зворотних та доповнювальних кодах використовують багаторозрядні суматори. Схеми та принцип функціонування суматорів розглянуті у [6].

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

Ознакою переповнення є розбіжність значень вхідного переносу P_{in} у знаковий розряд і вихідного переносу P_{out} із знакового розряду. Отже, ознаку переповнення можна сформувати перемикальною функцією $OVR=P_{in}\oplus P_{out}$.

Інший підхід для виявлення переносу складається у використанні другого допоміжного знакового розряду ліворуч від основного (першого) знакового розряду. Таке подання чисел називають *модифікованим машинним кодом*. Старший знаковий розряд при цьому завжди зберігає знак результату.

Формули кодування для модифікованих кодів мають вигляд:

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

$$[A]_{ДК} = \begin{cases} A, якщо\ A \geq 0; \\ 2^{n+2} + A, якщо\ A < 0. \end{cases} \qquad (1.2)$$

1.4. Додавання і віднімання чисел в зворотних кодах

В ЗК операція віднімання замінюється операцією додавання, при цьому знаковий розряд 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 ЗК є циклічний перенос зі знакового розряду в молодший розряд суми, завдяки якому здійснюється корекція результату.

Розглянемо чотири можливих випадки додавання чисел у модифікованих кодах.

Випадок 1. $A>0$, $B>0$, $A+B>0$.

Під час підсумовування кодів відповідно до функції кодування (1.2) ми повинні одержати

$$[A]_{ЗК}+[B]_{ЗК} = [A+B]_{ЗК}=A+B,$$

тому що доданки додатні, тобто

$$[A]_{ЗК} =A,\ [B]_{ЗК}=B \text{ і } [A]_{ЗК}+[B]_{ЗК} = A+B.$$

У даному випадку шуканий результат збігається з отриманим. Отже, корекція не потрібна. Помітимо, що при підсумовуванні нульових знакових розрядів перенос не формується, тобто корекція за циклічним ланцюгом справді буде відсутня.

Випадок 2. $A>0$, $B<0$, $|A|>|B|$, $A+B>0$.

Сума додатна, отже, результат повинен мати вигляд

$$[A]_{ЗК}+[B]_{ЗК}= [A+B]_{ЗК}=A+B.$$

У підсумовуванні беруть участь коди $[A]_{ЗК}=A$ і $[B]_{ЗК}=2^{n+2}-2^k+B$. Сума буде мати вигляд

$$[A]_{ЗК}+[B]_{ЗК}=A+2^{n+2}-2^k+B= 2^{n+2}-2^k+(A+B).$$

У результаті операції додавання виникла помилка $(2^{n+2}-2^k)$. Тут 2^{n+2} – перенос зі знакового розряду за межі розрядної сітки, тобто на цю величину корекція не потрібна. Таким чином, результат отриманий з недоліком на величину 2^k. Корекція на $(+2^k)$ здійснюється додаванням у молодший розряд результату переносу за циклічним ланцюгом, що завжди виникає при зазначених значеннях операндів.

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

Виконання завдання

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

Випадок 3. *A>0, B<0, |A|<|B|, A+B<0.*

Сума від'ємна, виходить, результат відповідно до виразу (1.2) повинен мати вигляд

$$[A]_{зк}+[B]_{зк}=[A+B]_{зк}=2^{n+2}-2^k+(A+B).$$

Підсумовуються коди $[A]_{зк}$=A і $[B]_{зк}=2^{n+2}-2^k+B$. В результаті одержимо

$$[A]_{зк}+[B]_{зк}=A+2^{n+2}-2^k+B=2^{n+2}-2^k+(A+B).$$

Шуканий результат збігається з отриманим, тобто корекція не потрібна.

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

Виконання завдання

$$\begin{array}{r} [A]_{зк}=00,01001 \\ +\underline{[B]_{зк}=11,01010} \\ [C]_{зк}=11,10011 \\ [C]_{пк}=11,01100 \end{array}$$

Випадок 4. *A<0, B<0, A+B<0.*

Сума від'ємна, відповідно до цього, результат повинен мати вигляд

$$[A]_{зк}+[B]_{зк}=[A+B]_{зк}=2^{n+2}-2^k+(A+B).$$

Підсумовуються коди $[A]_{зк}=2^{n+2}-2^k+A$ і $[B]_{зк}=2^{n+2}-2^k+B$.

В результаті на суматорі одержимо

$$[A]_{зк}+[B]_{зк}=2^{n+2}-2^k+A+2^{n+2}-2^k+B=2^{n+2}-2^k+(A+B)+2^{n+2}-2^k.$$

Потрібна корекція результату, що здійснюється аналогічно другому випадкові.

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

Виконання завдання

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

Під час додавання чисел з однаковими знаками може виникнути переповнення розрядної сітки. Ознакою переповнення при використанні модифікованих кодів є різні цифри в знакових розрядах. Функцію переповнення можна записати у вигляді $OVR=3P_2 \oplus 3P_1$. Старший знаковий розряд завжди зберігає правильний знак результату.

Приклад 1.12. Задано $A=0,10101$ і $B=0.01110$. Знайти суму C.

Виконання завдання

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

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

Виконання завдання

$$\begin{array}{r} [A]_{зк}=11,01010 \\ +\underline{[B]_{зк}=11,10001} \\ 10,11011 \\ \underline{+\qquad\quad 1}\text{ (перенос)} \\ [C]_{зк}=00,01100 \text{ – від'ємне переповнення} \end{array}$$

Функціональна схема пристрою, що реалізує операцію додавання і віднімання в зворотних кодах наведена на рис. 1.4.

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 $3P_2$ і $3P_1$.

Корекція суми здійснюється за допомогою ланцюга циклічного переносу зі знакового розряду в молодший розряд суматора $CO \rightarrow CI$. Цей ланцюг завжди замкнутий, тому що перенос виникає тільки тоді,

коли потрібно реалізувати корекцію суми. Щоб знайти переповнення розрядної сітки використовується ознака переповнення OVR ($OVR = 3P_2 \oplus 3P_1$). При значенні $OVR = 0$, переповнення розрядної сітки відсутнє, у випадку, коли $OVR = 1$ наявне переповнення розрядної сітки.

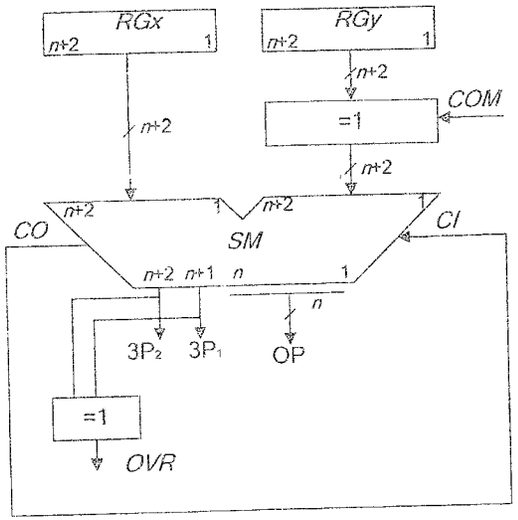


Рис. 1.4. Схема виконання операцій додавання і віднімання в зворотних кодах

1.5. Додавання і віднімання чисел в доповнювальних кодах

У ДК операція віднімання, як і в ЗК, замінюється операцією додавання зменшуваного до від'ємного від'ємника. За підсумовування операндів у ДК автоматично отримуємо 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

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

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

Виконання завдання

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

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

Виконання завдання

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

$$\begin{array}{r} [A]_{\text{ДК}} = 11,0\ 1011 \\ +\,[B]_{\text{ДК}} = 11,1\ 0010 \\ \hline [C]_{\text{ДК}} = 10,1\ 1101 \end{array}$$ – 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.

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. 1.4), 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* суматора. 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.

1.6. 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. 1.5.

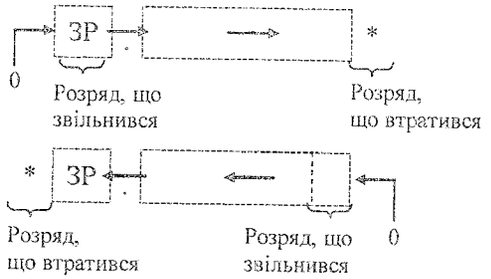


Рис. 1.5. Операційна схема логічного зсуву двійкових чисел

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

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

Виконання завдання

A	1.0101,1101
$A\rightarrow$	0.1010,1110 *

A	1.0101,1101
$\leftarrow A$	* 0.1011 1010

Арифметичний зсув виконується з урахуванням знакового розряду. Правила зсуву чисел поданих у ПК, ЗК та ДК відрізняються. Арифметичний 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. 1.6

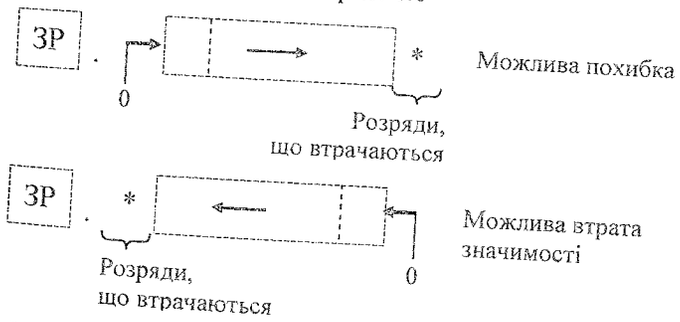


Рис. 1.6. Операційна схема арифметичного зсуву чисел у ПК

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 ЗК.

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. 1.7 Зображена операційна схема реалізації арифметичного зсуву ліворуч від'ємних чисел поданих у ЗК.

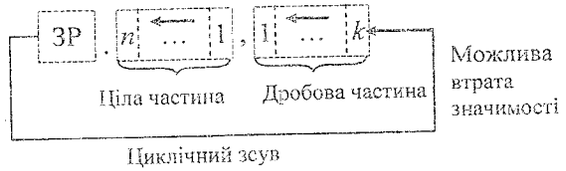


Рис. 1.7. Операційна схема арифметичного зсуву ліворуч від'ємних чисел у ЗК

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

$A_{(2)} = 1.\ 1101,\ 1010;\ B_{(2)} = 1.\ 0011,\ 0011.$

Виконання завдання

A	1.1101,0101
$\leftarrow A$	1.1010,1011
B	1.0011,0011
$\leftarrow B$	0.0110,0111 ← Втрата значимості

Арифметичний зсув праворуч чисел, поданих у ЗК.

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

На рис. 1.8 зображена операційна схема реалізації арифметичного зсуву праворуч від'ємних чисел поданих у ЗК.

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

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

$A_{(2)} = 0.\ 1011,\ 1001;\ B_{(2)} = 1.\ 1010,\ 0011.$

Виконання завдання

A	0.1011,1001
$\rightarrow A$	0.0101,1100*
B	1.1010,0011
$\rightarrow B$	1.1101,0001*

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. 1.9 зображена операційна схема реалізації арифметичного зсуву ліворуч чисел поданих у ДК.

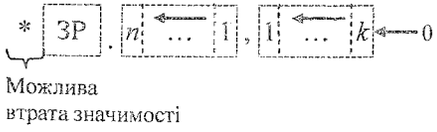


Рис. 1.9. Операційна схема арифметичного зсуву ліворуч від'ємних чисел у ДК

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. 1.10.

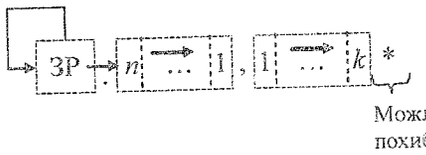


Рис. 1.10. Операційна схема арифметичного зсуву праворуч від'ємних чисел у ДК

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

$$\begin{array}{ll} A & 0.1011,0111 \\ \leftarrow A & *1.0110,1110 \\ A \rightarrow & 0.0101,1011* \end{array}$$

$$\begin{array}{ll} B & 1.1011,1001 \\ \leftarrow B & *1.0111,0010 \\ \rightarrow B & 1.1101,1100* \end{array}$$

1.7. 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 *мікроопераціями* 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 (: =).

Пересилання слів здійснюється, як правило, між двома регістрами. Результатом пересилання є запис слова в регістр, що є приймачем слова. Джерелом інформації, окрім регістрів, можуть бути зовнішні входи пристрою, на які надходять дані, наприклад, з пам'яті, загальної шини системи, тощо.

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. 1.11.

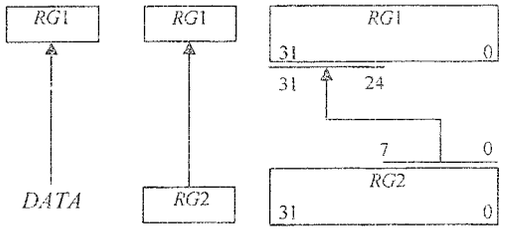
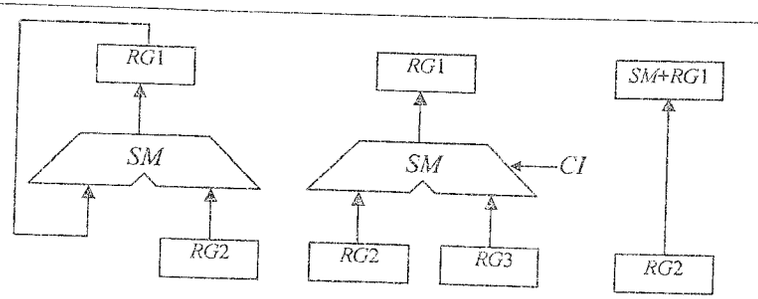


Рис. 1.11. Операційні схеми для виконання мікрооперацій пересилання даних

Для підсумовування слів в схемі використовують суматор. Приклади операційних схем, що відповідають мікроопераціям *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. 1.12, *а*, *б*. В схемах рис. 1.12, *а* і 1.12, *б* використовуються комбінаційні суматори, а в схемі рис. 1.12, *в* – накопичуючий суматор, який є композицією суматора і регістра.

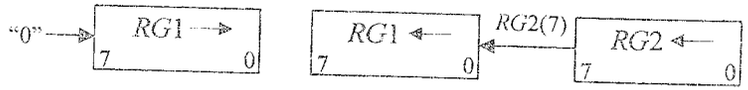
Мікрооперації інкременту ($CT := CT+1$) та декременту ($CT := CT-1$) виконуються на лічильнику.



а *б* *в*

Рис. 1.12. Операційні схеми для виконання мікрооперацій підсумовування чисел:
а, *б* – із застосуванням комбінаційних суматорів; *в* – з використанням накопичуючого суматора.

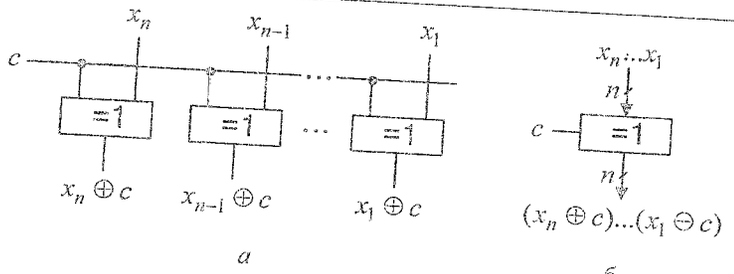
Мікрооперації зсуву слів можуть бути виконані на регістрі зсуву праворуч або ліворуч. Для означення напрямку зсуву використовують оператори зсуву *r* (*right*) та *l* (*left*). За допомогою складеного слова визначають значення розряду, який заповнюється внаслідок зсуву. Приклад операційної схеми, що відповідає мікрооперації зсуву *RG*1:=0.*r*[*RG*1] зображений на рис 1.13, *а*. Для реалізації зсуву, що відповідає операційній схемі на рис. 1.13, *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. 1.13. 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. 1.14).



а *б*

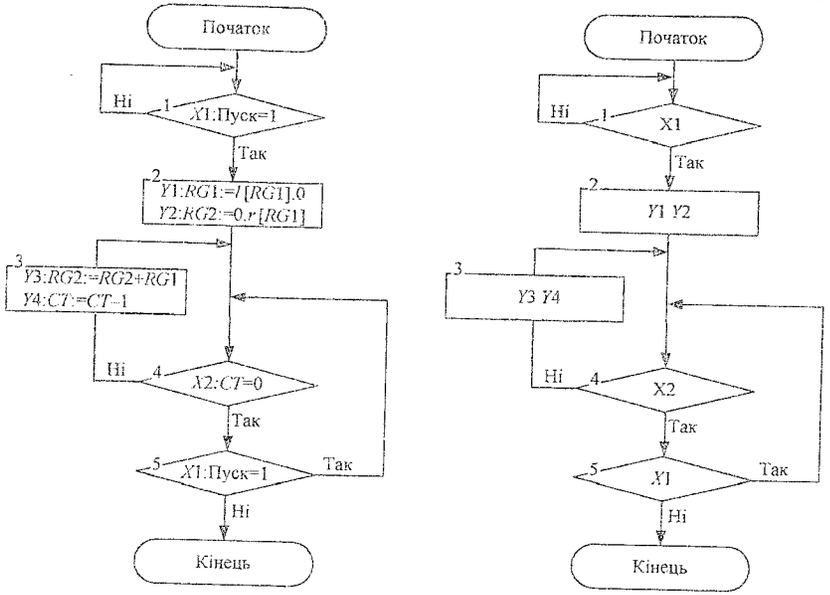
Рис. 1.14. Інвертування розрядів x_i двійкового числа:
а – логічна схема; *б* – умовне позначення (*c* – сигнал управління інвертуванням)

Послідовність мікрооперацій, що забезпечує задане перетворення інформації, називається мікроалгоритмом.

Для опису мікроалгоритмів використовують *графічні схеми мікроалгоритмів*, яки можуть бути описані мовами ГСА і ЛСА [6].

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. 1.15). Для розробки такого мікроалгоритму достатньо скласти операційну схему пристрою, який виконує задані МО.

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.



а *б*

Рис. 1.15. Приклад запису мікроалгоритму:
а – з розширеним змістовним описом мікрооперацій; *б* – із скороченим описом мікрооперацій у вигляді ідентифікаторів

РОЗДІЛ 2

Арифметико-логічні пристрої

Арифметико-логічні пристрої призначені для виконання арифметичних та логічних операцій над машинними словами (числами, командами та машинними кодами інших об'єктів).

За структурою розрізняють АЛП з розподіленою та зосередженою логікою (інакше АЛП із закріпленими та загальними мікроопераціями). В АЛП першого типу апаратура для реалізації мікрооперацій розподілена між регістрами та закріплена за ними, тобто кожен регістр використовує власну логіку для виконання мікрооперацій. У пристроях другого типу всі логічні ланцюги об'єднані в арифметико-логічному блоці, а всі регістри реалізовані у вигляді надоперативного запам'ятовуючого пристрою.

За способом обміну між регістрами та арифметичним блоком АЛП із зосередженою логікою поділяються на послідовні, паралельні та послідовно-паралельні.

За формою подання чисел розрізняють АЛП із плаваючою комою, АЛП із фіксованою комою, та АЛП, що працюють як із плаваючою так і з фіксованою комою. В деяких ЕОМ передбачається режим цілих чисел, за яким кома фіксується після останнього розряду.

В залежності від основи системи числення розрізняють двійкові та десяткові АЛП. Відомі АЛП, що працюють в системах числення з основою 2^k, де k – додатне ціле число.

В залежності від часу, що відводиться на виконання окремих операцій, розрізняють АЛП синхронного, асинхронного та комбінованого типу. На виконання різних операцій в синхронних АЛП відводиться один і той самий час. В асинхронних АЛП на виконання кожної операції відводиться стільки тактів машинного часу, скільки потрібно, а виконання наступної операції розпочинається тільки за наявністю сигналу завершення поточної операції. Комбіновані АЛП поєднують простоту синхронних та швидкодію асинхронних АЛП. За цим всі операції поділяються на декілька групи, як правило на дві – однотактні та бага-

тотактні операції. Однотактні операції реалізуються за синхронним способом, багатотактні – за асинхронним.

Набір операцій, що виконується в АЛП, є дуже важливою його характеристикою. Набір операцій повинен бути функціонально повним, для реалізації будь якого обчислювального алгоритму. З ціллю підвищення швидкодії і спрощення програмування указаний набір як правило має значну надлишковість. Кількість операцій коливається від декількох десятків до декількох сотень. За великим розмаїттям наборів операцій в їх складі звичайно є чотири основні арифметичні та найбільш важливі логічні операції, такі як порівняння, порозрядна кон'юнкція, тощо.

Виконання будь якої операції в АЛП зводиться до виконання послідовності мікроопераций на регістрах, суматорах і таке інше. Послідовність МО, виконання яких приводить до виконання операції називають мікроалгоритмом цієї операції. В залежності від способу реалізації мікроалгоритмів розрізняють АЛП із схемним та мікропрограмним способом управління операціями (дивись розділ 3).

Виконання любих операцій в АЛП розпочинається з підготовчих МО добування операндів із основної пам'яті та фіксації їх у визначеному порядку в регістрах АЛП. Ділі, з ціллю спрощення викладення матеріалу, робота схем, що виконують окремі операції, зазвичай, розглядається з того моменту, коли всі підготовчі дії стосовно добування операндів та фіксації їх у ОП завершені.

2.1. Арифметико-логічні пристрої з розподіленою логікою

2.1.1. Основні способи множення чисел у прямих кодах

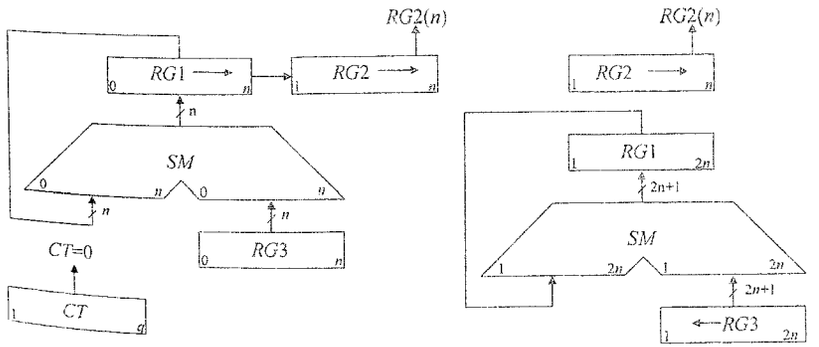
Принцип побудови пристроїв, що реалізують різні способи множення, показаний на рис. 2.1, де *RG3* – регістр множеного, *RG1* – регістр добутку, *RG2* – регістр множника. Цифрами зазначені номери розрядів *SM* і регістрів, а стрілками показаний напрямок зсуву кодів у регістрах.

Цифри, що записані в молодших розрядах регістрів *RG3* і *RG1* при реалізації першого способу мають вагу 2^{-n}, а при реалізації інших способів – 2^{-2n}. Перед початком множення будь-яким способом регістр *RG1* встановлюється в нульовий стан. Підрахунок кількості циклів множення забезпечують лічильники *CT*, відповідно з чим обирається й його розрядність q.

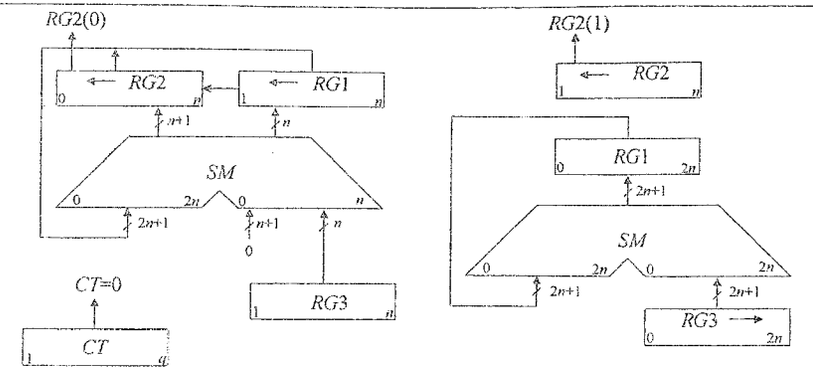
Під час множення *першим способом* (рис. 2.1, *а*) в першому такті i-го циклу аналізується значення *RG2(n)* – молодшого (n-го) розряду регістру *RG2*, в якому знаходиться чергова цифра множника. Вміст *RG3* додається до суми часткових добутків, що знаходяться в регістрі RG1, якщо *RG2(n)*=1, або не додається, якщо *RG2(n)*=0. В другому такті здійснюється правий зсув в регістрах *RG1* і *RG2*, що еквівалентно множенню їх вмісту на 2^{-1}. При зсуві цифра молодшого розряду регістру *RG1* записується у вивільнюваний старший розряд регістру *RG2*. Після виконання n циклів молодші розряди $2n$-розрядного добутку будуть записані в регістр *RG2*, а старші – у *RG1*.

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 і зсуву відповідно.

Перед початком множення *другим способом* (рис. 2.1, *б*) множник X записують в регістр *RG2*, а множене Y – в молодші розряди регістру *RG3* (тобто в регістрі *RG3* установлюють $Y_0 = Y2^{-n}$). В кожному i-му циклі множення додаванням кодів *RG3* і *RG1* управляє цифра *RG2*(n), а в регістрі *RG3* здійснюється зсув вліво на один розряд, в результаті чого формується величина $Y_i = 2Y_{i-1}$. Оскільки сума часткових добутків в процесі множення нерухома, зсув в регістрі *RG3* можна сполучити 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 *RG2*, що також приводить до збільшення швидкодії, якщо множник ненормалізований.



а *б*



в *г*

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

Під час множення *третім способом* (рис. 2.1, *в*) вага молодшого розряду *RG*3 дорівнює 2^{-2n}, тому код в регістрі *RG*3 являє собою значення $Y2^{-n}$. На початку кожного циклу множення здійснюється лівий зсув в регістрах *RG*1 і *RG*2, а потім виконується додавання, яким управляє *RG*2(1). В результаті підсумовування вмісту *RG*3 і *RG*1 може виникнути перенос в молодший розряд регістру *RG*2. У старшій частині суматора, на якому здійснюється підсумовування коду *RG*2 з нулями, відбувається поширення переносу. 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 і дорівнює $t_м = n(t_п + t_з)$.

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(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 з дробовими числами часто потрібно обчислювати не $2n$, 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 і відкидання всіх розрядів, розташованих правіше n-го. При цьому похибка стає знакозмінною, а максимальне абсолютне її значення не перевищує половини ваги молодшого розряду. Додаткового такту підсумовування для округлення не потрібно. Досить записати одиницю перед початком множення в той розряд регістру *RG*1, що після виконання множення залишається старшим розрядом, який відкидається.

У процесі формування суми часткових добутків код з регістру *RG*1 видається на суматор *SM*, а з виходів *SM* знову записується в регістр *RG*1. У зв'язку з цим при використанні потенційних елементів регістр *RG*1 будують на тригерах із внутрішньою затримкою. Характер управляючих сигналів і ланцюга, на який вони впливають, визначається конкретною теоретичною реалізацією вузлів і використовуваною елементною базою.

У операційних пристроях, що реалізують другий і четвертий способи множення, можна без пересилань кодів між регістрами обчислювати вирази вигляду $\sum X_i Y_i$, де $(i = \overline{1,n})$ для чого досить черговий результат операції залишати в регістрі *RG*2, який в цьому випадку повинен мати додаткові старші розряди.

У операційному пристрої, що реалізує третій спосіб, можна без пересилань обчислювати, наприклад, функції вигляду X_i. Для цього множник X перед початком обчислення записується в регістр *RG*3 і в молодші розряди регістру *RG*2, а потім $(i-1)$ раз виконується операція множення з округленням проміжних результатів до n розрядів. Після кожної чергової операції регістр *RG*1 встановлюється в нульовий стан. Остаточний результат буде знаходитися в n молодших розрядах регістру *RG*2. Найбільш простими є пристрої, що реалізують перший спосіб,

а найбільш швидкодіючими – другий і четвертий. Однак другий спосі
не має особливих переваг порівняно з четвертим і, крім того, вимага
великих апаратурних витрат при реалізації.

2.1.2. Синтез арифметико-логічних пристроїв з розподіленою логікою

АЛП з розподіленою логікою застосовуються в спеціалізованих т
проблемно-орієнтованих ЕОМ. Відрізняються від АЛП інших типів вис
кою швидкодією, але мають досить обмежені функціональні можливост
Структура таких АЛП залежить від операцій, що вони виконують, причом
для кожної системи операцій необхідно будувати окремий АЛП.

АЛП з розподіленою логікою складаються з двох функціональн
частин (рис. 2.2):

- управляючий пристрій, що забезпечує формування всіх упра
ляючих сигналів;
- операційний пристрій, забезпечує перетворення інформації
виконує мікрооперації над машинними словами.

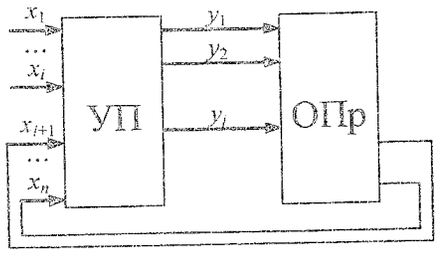


Рис. 2.2. Загальна структура АЛП

Побудова таких АЛП відбувається за наступними етапами:

1. Для кожної операції будується операційна схема та функціо
льний мікроалгоритм (Ф-микроалгоритм). Рекомендується обирати т
мікроалгоритми виконання операцій, що краще сполучаються, тоб
вимагають однакового напрямку зсувів в регістрах, однакову розр
ність регістрів, одні й ті самі джерела операндів суматорів і таке інш

2. Обирається розрядність регістрів, лічильників. Виконується
гічне моделювання роботи ОПр, наприклад, із застосуванням діагра
стану регістрів при виконанні МА з критичними значеннями операн

3. Розробляється функціональна та принципова схеми ОПр із за
значенням управляючих сигналів для кожного вузла пристрою.

4. Складається закодований структурний мікро алгоритм (С-ми
кроалгоритм) виконання заданих операцій.

5. Виконується синтез управляючого пристрою.

6. Складається функціональна та принципова схеми АЛП.

Приклад 2.1. Побудувати схему АЛП для реалізації операції множення чисел за першим способом.

Синтезувати схему, що дозволяє обчислити добуток $Z=Y\times X$ двох правильних дробів $Y= 0, y_1, y_2...y_n$ та $X= 0, x_1, x_2...x_n$. Вважати, що розрядність дробів $n = 16$.

Виконання завдання

Операційна схема, що реалізує перший спосіб множення, подана на рис. 2.3, де *RG*1 – регістр накопичення суми часткових добутків, *RG*2 – регістр множника, RG3 – регістр множеного, *RG*4 (*CT*) – лічильник циклів, *TC* – тригер переносу, *SM* – комбінаційний суматор. Регістри *RG*1 та *RG*2 реалізують мікрооперації зсуву, лічильник *RG*4 дозволяє формувати ознаку нуля – що визначає закінчення обчислення добутку. За нульовим вмістом регістру *RG*4 результат обчислення формується в регістрах *RG*1 та *RG*2.

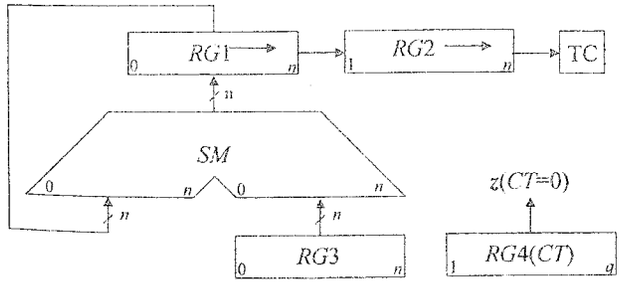


Рис. 2.3. Операційна схема множення

Зауваження. *Операційні схеми* застосовують для відображення апаратури, що застосовується для виконання послідовності заданих мікрооперацій. ОС містить всі функціональні частини операційного пристрою із зазначенням зв'язків між ними. За ОС виконання операції будують *структурну схему* ОПр.

Для розробленої операційної схеми побудуємо Ф-мікроалгоритм. Припустимо, що ОПр входить до складу АЛП із централізованим управлінням, отже робота цього блоку розпочинається із надходження сигналу "Пуск" від центрального блоку управління. Функціональний мікроалгоритм зображений на рис. 2.4, де *TC* – стан тригера переносу, *z* – значення ознаки нуля в лічильнику циклів *RG*4.

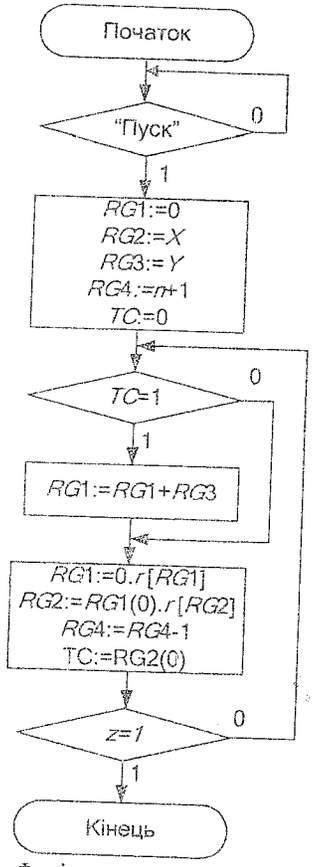


Рис. 2.4. Ф-мікроалгоритм множення чисел

Зауваження. Мікроалгоритми можна розглядати на функціональному та структурному рівнях. На *функціональному* рівні розглядають узагальнені МО, які не суперечать операційній схемі пристрою. При цьому можна не враховувати кількість тактів, необхідних для виконання МО. На *структурному* рівні операційна вершина відповідає одному такту перетворення інформації. С-мікроалгоритми повністю відповідають схемі пристрою з урахуванням елементної бази та тривалості управляючих сигналів. Для побудови С-мікроалгоритму необхідно отримати перелік МО в АЛП, що розробляється.

Логічне моделювання потактової роботи ОПр приведене в табл. 2.1

Значення операндів:

$Y = 5_{10} = 0101_2$;

$X = 7_{10} = 0111_2$;

$Z = 35_{10} = 00100011_2$.

Розрядність дробів $n = 4$.

Таблиця 2.1. Логічне моделювання роботи ОПр

№ такту	*RG*1	*RG*2	*TC*	*RG*3	*RG*4	*z*	МО
ПС	0000	0101	0	0111	0101	0	Початковий стан
1	0000	0010	1	0111	0100	0	*RG*1→, *RG*2→, *RG*4 – 1; *z* = 0
2	0000 +0111 0111 0011	0010 1001	1 0	0111 0111	0100 0011	0 0	*RG*1+ *RG*3 *RG*1→, *RG*2→, *RG*4 – 1; *z* = 0
3	0001	1100	1	0111	0010	0	*RG*1→, *RG*2→, *RG*4 – 1; *z* = 0
4	0001 +0111 1000 0100	1100 0110	1 0	0111 0111	0010 0001	0 0	*RG*1+ *RG*3 *RG*1→, *RG*2→, *RG*4 – 1; *z* = 0
5	0010	0011	0	0111	0000	1	*RG*1→, *RG*2→, *RG*4 – 1; *z* = 1

На підставі ОС множення та Ф-мікроалгоритму складемо перелік управляючих сигналів для всіх функціональних частин ОПр та побудуємо функціональну схему.

Перелік управляючих сигналів наведений в табл. 2.2, функціональна схема ОПр зображена на рис. 2.5.

Таблиця 2.2. Таблиця управляючих сигналів

Елемент	Мікрооперація	Управляючий сигнал
RG1	Скидання	*R*
	Запис	*W*
	Зсув вправо	*SR*
	Заповнення старшого розряду при зсуві вправо	*DR*
RG2	Запис	*W*
	Зсув вправо	*SR*
	Старший розряд при зсуві вправо	*DR*
RG3	Запис	*W*
RG4	Запис	*W*
	Декремент лічильника	*dec*
TC	Скидання	*R*
	Запис молодшого розряду множника у тригер переносу	*C*

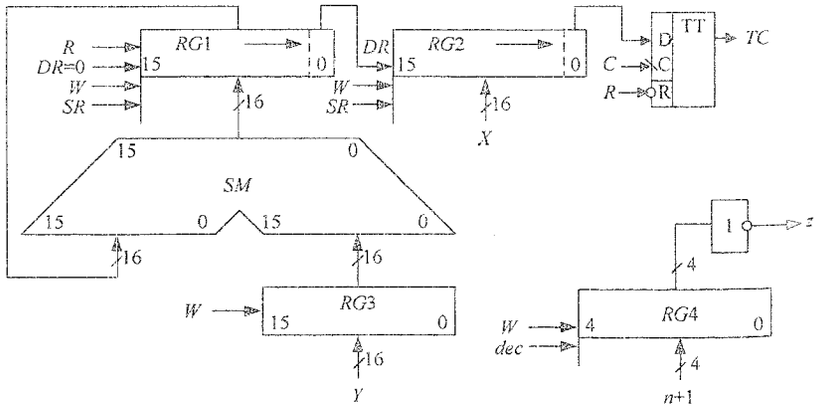


Рис. 2.5. Функціональна схема операційного пристрою

За побудованою функціональною схемою будуємо функціональ структурний мікроалгоритм (ФС-мікроалгоритм), що зображений

рис 2.6. Індекс указує до якої з функціональних частин пристрою множення належить управляючий сигнал.

Кодування сигналів управління та логічних умов. наведене в табл. 2.3 – 2.4.

Для забезпечення перепаду сигналів управління SR_1, SR_2, *dec*, C_{TC} (вершину з цими сигналами охоплює петля рис. 2.6) необхідно ввести порожню додаткову вершину.

Закодований ФС-мікроалгоритм зображений на рис. 2.7, де управляючі сигнали та сигнали логічних умов відповідають рис. 2.6 та табл. 2.2 – 2.4.

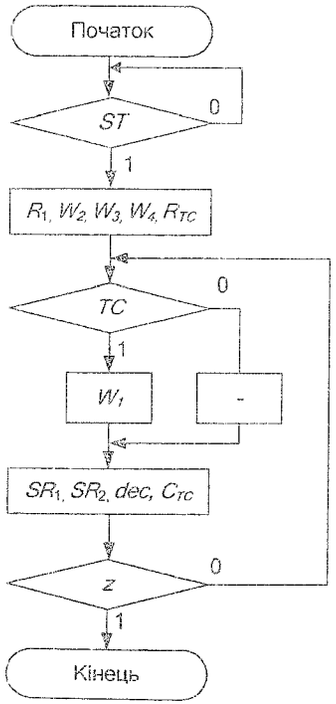


Рис. 2.6. Функціонально-структурний мікроалгоритм

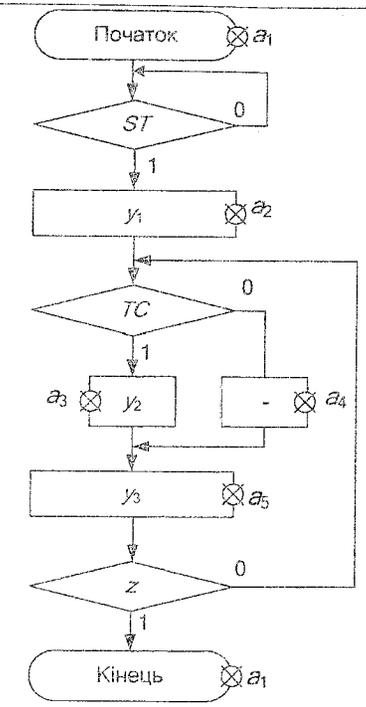


Рис. 2.7. Закодований функціонально-структурний мікроалгоритм

Таблиця 2.3. Кодування сигналів управління

Управляючі сигнали	Код
R_1	y_1
W_2	
W_3	
W_4	
R_{TC}	
W_1	y_2
SR_1	y_3
SR_2	
C_{TC}	
dec	

Отриманий закодований ФС-микроапгоритм є вихідним для здійснення синтезу управляючого пристрою.

Таблиця 2.4. Кодування логічних умов

Логічні умови	Код
Пуск	ST
Аналіз молодшого розряду множника	TC
Нульовий вміст лічильника	z

Для управління роботою ОПр застосуємо *пристрій управління з жорсткою логікою*, який реалізуємо у вигляді цифрового автомата Мура.

Розмітка ФС-микроалгоритма для автомата Мура наведена на рис. 2.7. Стани автомата позначені символами a_i. Часова діаграма роботи управляючого пристрою зображена на рис. 2.8. Часова діаграма відповідає потактовій роботі ОПр для прикладу, виконаного в табл. 2.1.

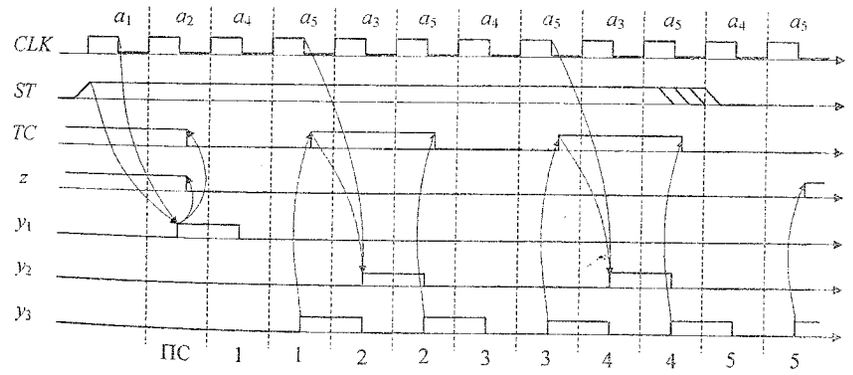


Рис. 2.8. Часова диаграма роботи пристрою управління

На рис. 2.9 зображена узагальнена структурна схема АЛП множення. Управляючі сигнали з виходів пристрою управління підключаються до входів відповідних функціональних частин ОПр.

Схема електрична функціональна АЛП для множення додатних чисел наведена у додатку А. Опис функціональної схеми наведений у прикладі 7.1.

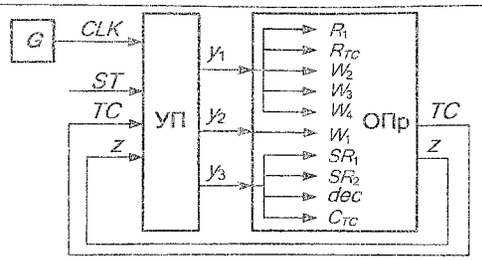


Рис. 2.9. Узагальнена структурна схема АЛП

2.2. Арифметико-логічні пристрої з зосередженою логікою

Можна визначити наступні різновиди АЛП із зосередженою логіко (з загальними мікрооперaціями):

- АЛП із двоспрямованою магістраллю та одноадресним НОЗП;
- АЛП із односпрямованими магістралями та одноадресним НОЗП
- АЛП із односпрямованими магістралями та двоадресним НОЗП.

АЛП із зосередженою логікою мають широкі функціональні можл вості і можуть виконувати велику кількість операцій. При цьому кількіс операцій, що виконується, не впливає на складність пристрою. Зміна алг ритму роботи не потребує змін структури АЛП, а тягне за собою лиш зміни МА, що зберігаються в пам'яті блока управління.

Усі регістри зосереджені у надоперативному запам'ятовуючому пр строї. Мікрооперації виконуються у процесі пересилки слів із одного регі тру в інший через загальну логіку – арифметико-логічний блок.

Перетворення інформації в АЛП управляється словом, що назив ється мікрокомандой. Мікрокоманда має дві частини

F	AD

,

де *AD* – адресна частина, що визначає адресу регістру НОЗП, над як необхідно виконати мікрооперацію;

F – функціональна частина, що задає тип МО в АЛП, спосіб дифікації регістру стану, напрямок зсуву акумулятора, ко тацію зв'язків.

Структура АЛП із двоспрямованою магістраллю та одноадреси НОЗП зображена на рис. 2.10, де:

АЛБ – арифметико-логічний блок;
БРС – блок регістру стану;
РВ – регістр тимчасового зберігання;
РА – регістр акумулятор;
РС – регістр стану;
НОЗП – складається з регістрів (*R0…RN*);
ЛШ – локальна шина;
БД – буфер даних;
СМ – системна магістраль.

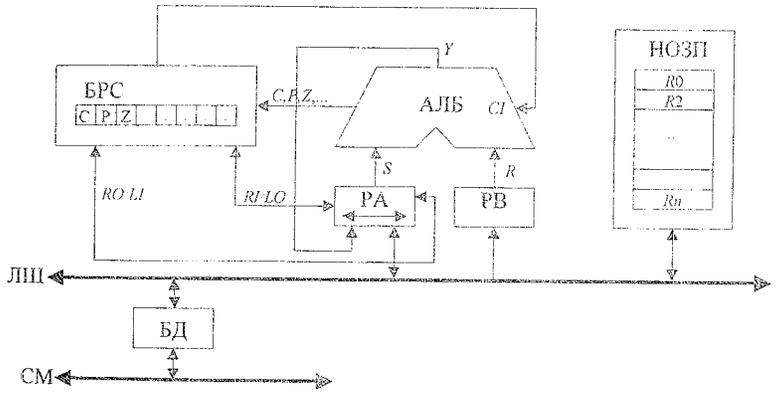


Рис. 2.10. Структура АЛП із двоспрямованою магістраллю та одноадресним НОЗП

АЛП є комбінаційною схемою призначеною для виконання арифметичних та логічних мікрооперацій над операндами *S* і *R*. Операнди *S* і *R* поступають на входи суматора з регістрів РА та РВ, результат виконання мікрооперації *Y* з виходу суматора записується у регістр РА. Регістри РА та РВ зв'язані з регістрами НОЗП по локальній шині. БРС застосовується для зберігання ознак. В якості вхідного переносу *CI* суматора можуть бути застосовані значення розрядів регістру стану (*C*, *Z* та інші). Вихідний перенос видається з АЛБ через вихід *CO* і може біти записаний в один із розрядів РС. Над вмістом акумулятором РА може бути виконана операція зсуву. При зсуві вправо (в сторону молодших розрядів) старший розряд заповнюється значенням, що надходить по ланцюгу *RI/LO*, а молодший (вихідний) розряд по ланцюгу *RO/LI* завантажується у регістр РС. При лівому зсуві вихідний молодший розряд

подається по ланцюгу *RO/LI*, а вихідний поступає в РС по ланцюгу *RI/LO*.

Локальна шина забезпечує обмін інформацією між вузлами НОЗП, причому, у кожному такті передача інформації по ЛШ може відбуватися лише в одному напрямку. Буфер даних БД застосовується для обміну інформацією між АЛП та системною магістраллю.

Мікрооперації, що виконуються в АЛП поділяються на арифметичні та логічні. Приклади МО, що реалізує АЛП даного типу, наведений у табл. 2.5.

Таблиця 2.5. Перелік мікрооперацій АЛП

Мнемоніка	МО
	Арифметичні операції
add	$Y:= S + R + CI$
sub	$Y:= S - R - 1 + CI$
sub	$Y:= R - S - 1 + CI$
	Логічні операції
and	$Y:= S \wedge R$
or	$Y:= S \vee R$
xor	$Y:= S \oplus R$

Структура мікрокоманди для даного типу АЛП має наступний вигляд:

F	*AD*

.

Для опису мікрокоманд можна застосовувати як змістовний запис МО так і запис у мнемонічному вигляді. Мнемонічна запис для арифметичної МК має наступний вигляд:

```
{<мнемоніка> [<оператор зсуву>], <приймач результату>,
<операнд 1>, <операнд 2>, CI}.
```

Наприклад, змістовний запис $R0:= r(R0 + R1)$ можна подати у мнемонічному вигляді `{add sr, r0, r0, r1, 0}`.

Формат логічної МК відрізняється від арифметичної відсутністю *CI*.

Приклад 2.2. Для АЛП із двоспрямованою магістраллю та одноадресним НОЗП побудувати мікроалгоритм виконання узагальненої дії $R0:= r(R1 + R2)$, де $R0$, $R1$, $R2$ – регістри НОЗП.

Виконання завдання

Мікроалгоритм виконання заданої операції зображений на рис. 2.11.

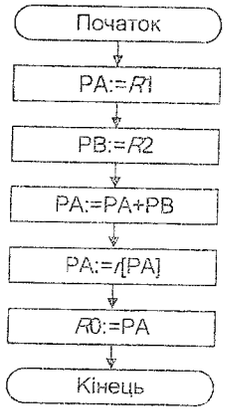


Рис. 2.11. Мікроалгоритм виконання операції в АЛП із двоспрямованою магістраллю

Перевагою АЛП розглянутого типу є простота реалізації, але такі АЛП характеризуються низькою швидкодією.

Для підвищення швидкодії застосовують декілька магістралей для обміну інформацію. До таких АЛП належать АЛП із односпрямованими магістралями із одноадресним і двоадресним НОЗП.

Структура АЛП із односпрямованими магістралями та одноадресним НОЗП наведена на рис. 2.12, де:

S – зсувач даних;
MX – мультиплексор вибору другого операнду.

Зсувач *S* є комбінаційною схемою, що виконує модифікацію результату. Він дозволяє залишити результат виконання МО незмінним, або зсунутим вправо чи вліво. При виконанні зсувів із БРС може бути записана ознака у розряд, що звільняється при зсуві, а вихідний розряд може бути записаний в один із розрядів РС.

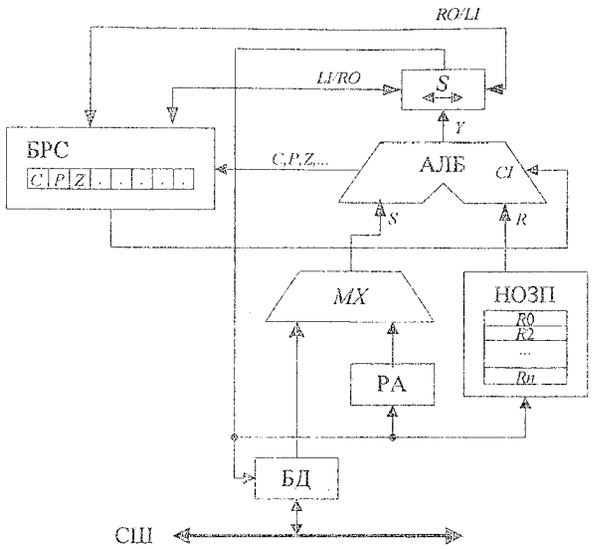


Рис. 2.12. Структура АЛП з односпрямованою внутрішньою магістраллю та одноадресним НОЗП

В АЛП із односпрямованою магістраллю та одноадресним НОЗП виконуються арифметичні та логічні МО (табл. 2.5). Одним з операндів є регістр НОЗП, другим операндом може бути або вміст акумулятору РА, або зовнішні по відношенню до АЛП дані, що подаються з системної магістралі через буфер даних БД.

Структура мікрокоманди має наступний вигляд

F	*AD*

.

Типи зсувів, що, зазвичай, реалізовані в АЛП наведені в табл. 2.6.

Таблиця 2.6. Типи зсувів в АЛП

Мнемоніка	Зсув	
sl	C ← S ←	Зсув вліво логічний
slc	C ← S ←	Зсув вліво циклічний
sr	→ S → C	Зсув вправо логічний
src	→ S → C	Зсув вправо циклічний

Приклад 2.3. Для АЛП із односпрямованою магістраллю та одноадресним НОЗП побудувати мікроалгоритм виконання узагальненої дії $R0 := r(R1 + R2)$, де $R0$, $R1$, $R2$ – регістри НОЗП.

Виконання завдання

Для виконання заданої операції можна застосувати мікроалгоритм зображений на рис. 2.13.

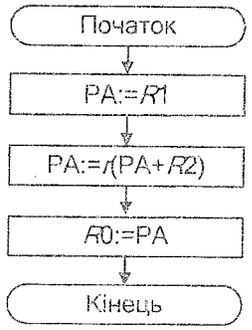


Рис. 2.13. Мікроалгоритм виконання операції в АЛП із односпрямованою магістраллю та одноадресним НОЗП

Як видно застосування даного типу АЛП дозволяє зменшити на два кількість кроків обробки інформації відносно АЛП, що розглядалося у прикладі 2.2.

Структура АЛП із односпрямованою магістраллю та двоадресним НОЗП приведена на рис. 2.14.

Всі регістри в НОЗП мають однакові можливості.

Для управління АЛП застосовуються мікрокоманда наступної структури:

F	*AD*1	*AD*2

,

де *AD*1 і *AD*2 – адреси регістрів НОЗП над вмістом яких виконується мікрооперація. Поле *AD*1 задає адресу першого операнда та адресу приймача результату, а *AD*2 – адресу другого операнда. Результат мікрооперації записується в НОЗП по першій або другій адресі.

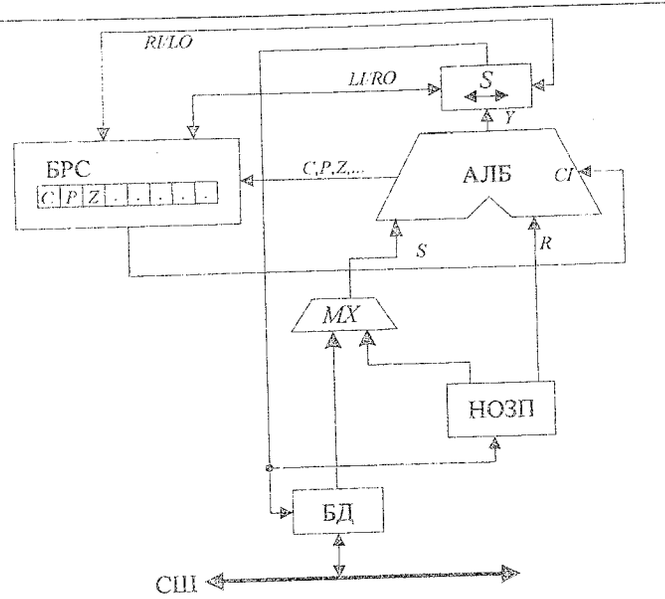


Рис. 2.14. Структура АЛП із однонаправленою магістраллю та двоадресним НОЗП

Приклад 2.4. Для АЛП із односпрямованою магістраллю та двоадресним НОЗП побудувати мікроалгоритм виконання узагальненої дії $R0:=r(R1+R2)$, де $R0$, $R1$, $R2$ – регістри НОЗП.

Виконання завдання

Задану операцію можна виконати за допомогою мікроалгоритму зображеного на рис. 2.15.

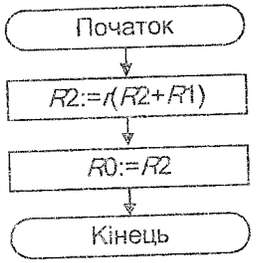


Рис. 2.15. Мікроалгоритм виконання операції в АЛП із односпрямованою магістраллю та двоадресним НОЗП

Таким чином, АЛП з двоадресним НОЗП та односпрямованою магістраллю дозволяє досягти більшої швидкодії по відношенню до АЛП з двоспрямованою магістраллю, але вони мають більш складну структуру.

Загальним недоліком АЛП із зосередженою логікою є неможливість суміщення мікрооперацій, бо для їх виконання застосовується загальний АЛБ. Тому основні напрями підвищення ефективності АЛП із зосередженою логікою направлені на вирішення проблеми суміщення мікрооперацій у часі та обробку слів подвійної довжини. Для цього у схему АЛП вводять додаткові вузли (регістри, лічильники, мультиплексори) на яких виконуються окремі МО. АЛП, що побудовані за такими принципами, називаються комбінованими, тобто вони поєднують у собі ознаки АЛП із розподіленою та зосередженою логікою.

Один із способів підвищення функціональних можливостей АЛП пов'язаний із введенням додаткового регістру розширювача RQ (рис. 2.16). Це дозволяє виконувати одночасно МО зсуву RQ, одночасно з будь-якою іншою МО в АЛБ, а також виконувати зсув слів подвійної довжини. Наприклад, можна виконати одночасно дві наступні МО: $R0:=l(R0+R1)$ та $RQ:=lRQ$.

За реалізації структур АЛП важливо, щоб включення додаткової апаратури не впливало на швидкодію системи. Схема, зображена на рис. 2.17 дозволяє виконувати зсув RQ на два розряди $RQ:=l2RQ$ за один такт, але при цьому тривалість такту АЛП, у порівнянні зі схемою на рис. 2.16, збільшується на час виконання МО у зсувачі SQ. Порівнювати швидкодію конкретних АЛП необхідно з врахуванням тривалості мікро операцій. На структурних схемах АЛП, зображених на рис. 2.16 – 2.17, умовно не показані БРС.

Щодо розробки мікроалгоритмів функціонування АЛП із зосередженою логікою можна визначити наступні етапи:

1. Відповідно до цільової функції проектування обрати структуру АЛП.

2. Для кожної операції, що виконується в АЛП розробляється операційна схема та структурний мікроалгоритм.

3. Виконується моделювання виконання операцій із застосуванням цифрових діаграм стану регістрів із критичними значеннями операндів.

4. Розробляється функціональний змістовний мікроалгоритм або мікропрограма у мнемонічних кодах, що є вихідними даними для побудови пристрою управління.

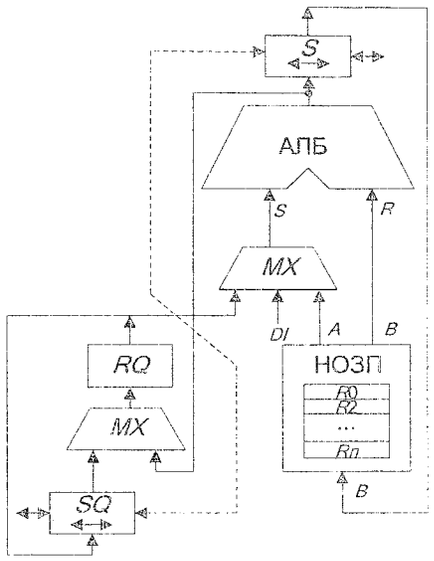


Рис. 2.16. Структура АЛП з регістром розширювачем

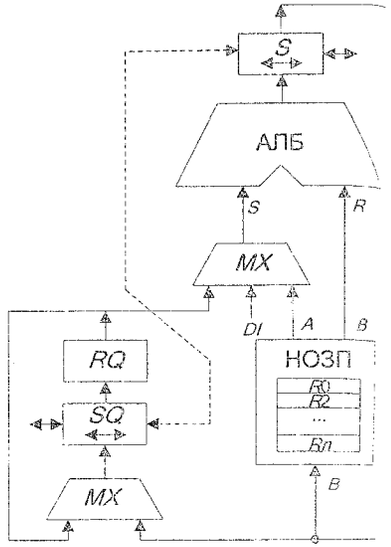


Рис. 2.17. Структура АЛП із зсувом на два розряди

Приклад 2.5. Розробити мікроалгоритм управління АЛП із зосередженою логікою для виконання операції множення за третім способом.

Виконання завдання

Для виконання задачі обираємо АЛП з односпрямованою магістраллю та двоадресним НОЗП, який дозволяє зменшити кількість мікрооперацій для перетворення даних.

Операційна схема множення за третім способом зображена на рис. 2.18. Операція множення виконуються із старших розрядів множника, за рахунок зсуву його вліво, сума часткових добутків також зсувається вліво, множене – нерухоме. Множник зберігається у регістрі *R*1, множене – у регістрі *R*3. У регістрі *R*2 накопичується сума часткових добутків. Регістр *R*4 застосовується як лічильник циклів. Розряд регістру стану застосовується для збереження ознак при зсуві регістрів *R*1 та *R*2. Формування чергової суми часткових добутків у регістрі *A* відбувається із поширенням переносу у регістр *R*1. При цьому на сумі-

торі *SM*1 підсумовується вміст регістру *R*1 з нулями, на вхід суматора *CI* подається розряд *C* регістру стану, де збережений вихідний перенос суматора *SM*2. Детальний опис принципу виконання операції множення за третім способом викладений у розділі 2.1.1.

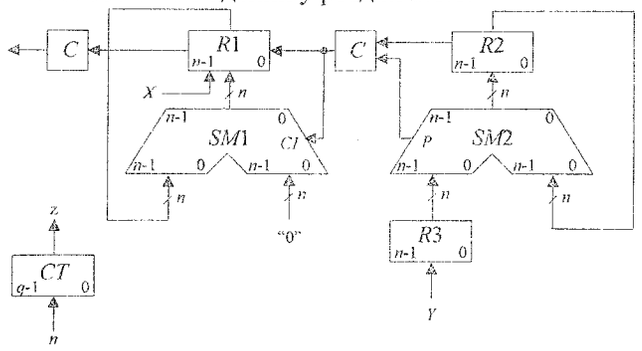


Рис. 2.18. Операційна схема операції множення

Цифрова діаграма операції множення для виконання операції $Z:=Y*X$, де $X = 0{,}0101$, $Y = 0{,}0111$, $0 < Y, X < 1$ наведена на рис. 2.19. Мікроалгоритм управління АЛП, розроблений на підставі операційного пристрою та цифрової діаграми операції множення, зображений на рис. 2.20.

№ такту	*C*	*RG*1 (*X*)	*RG*2 (*Z*)	*RG*3 (*Y*)	*RG*4	*z*	МО
ПС	0	0101	0000	0111	100	0	Початковий стан
1	0 0	1010	0000	0111	011	0	← *RG*2 ← *RG*1, C=0 *RG*4 – 1; *z* = 0
2	0 1 0	0100 0100 +0000 0100	0000 0000 +0111 0111	0111	010	0	← *RG*2 ← *RG*1, C=1 *RG*2+ *RG*3 *RG*4 – 1; *z* = 0
3	0 0	1000	1110	0111	001	0	← *RG*2 ← *RG*1, C=0 *RG*4 – 1; *z* = 0
4	1 1	0001	1100	0111			← *RG*2 ← *RG*1, C=1

	1	0001 +0001 0010	1100 +0111 0011		000	1	*RG2+ RG3* Поширення переносу *RG4 – 1; z = 0*

Рис. 2.19. Логічне моделювання роботи АЛП

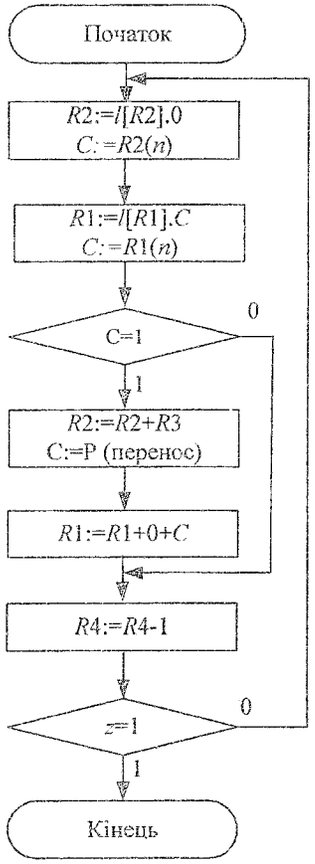


Рис. 2.20. Управляючий мікроалгоритм виконання операції множення

2.3. Лабораторна робота 1

Ціль роботи: Вивчити основні методи множення чисел у прямих кодах і способи їх апаратної реалізації, одержати навички в проектуванні й налагодженні схем управління операційними пристроями з розподіленою логікою.

Підготовка до лабораторного заняття

1. Розробити структурну схему операційного пристрою та змістовний мікроалгоритм множення додатних чисел відповідно до завдання наведеного у табл. 2.7), де $a_6,\ldots,a_1$ – молодші розряди двійкового номера залікової книжки. Для побудови схеми використати комбінаційний суматор, регістр-лічильник циклів та асинхронні регістри, що мають входи управління зсувами і занесенням інформації. На схемі повинні бути зазначені розрядність регістрів та шин.

2. Розробити функціональну схему операційного пристрою.

3. Виконати логічне моделювання роботи операційного пристрою за допомогою цифрової діаграми із зазначеними викладачем значеннями операндів.

4. Здійснити синтез пристрою управління, тип управляючого автомату обрати із табл.2.9. Пам'ять автомата реалізувати на тригерах, тип яких обрати з табл. 2.8. Ураховувати, що мікрооперації на регістрах виконуються за зворотним перепадом управляючих сигналів.

5. Побудувати часові діаграми роботи автомата для кожної комбінацій значень логічних умов.

Таблиця 2.7. Варіанти завдання

a_6	a_5	a_4	Спосіб множення	Розрядність операндів
0	0	0	1	16
0	0	1	2	8
0	1	0	3	16
0	1	1	4	8
1	0	0	1	8
1	0	1	2	16
1	1	0	3	8
1	1	1	4	16

Таблиця 2.8. Варіанти завдання

a_3	a_2	Тип тригера
0	0	JK
0	1	T
1	0	RS
1	1	D

Таблиця 2.9. Варіанти завдання

a_1	Тип автомата
0	Мили
1	Мура

Порядок виконання роботи

1. В моделюючій програмі ПРОГМОЛС 2.0 побудувати схем операційного пристрою для множення чисел та доповнити її схемо управляючого автомата. На першому етапі виходи автомата до вході операційного пристрою не підключати. Налагодити окремо схему оп раційного пристрою та схему управляючого автомата в синхронном режимі. Опис програмного комплексу ПРОГМОЛС 2.0 наведений у д датку М.

2. Підключити до управляючих входів операційного пристрою ви ходи автомата. Зробити комплексне налагодження схеми в синхронн му режимі й переконатися в правильності одержання результату.

3. Перейти до асинхронного моделювання. Дослідити зазначе викладачем часові параметри схеми.

Зміст звіту

Звіт з лабораторної роботи повинен включати короткі теоретичн відомості, необхідні для виконання лабораторної роботи; структурні т функціональні схеми; таблиці та діаграми, отримані при виконанні тео ретичного завдання, а також у процесі моделювання схем; висновки з роботою.

Контрольні питання

1. Охарактеризуйте чотири основних методи множення чисел.
2. Як розрахувати розрядність вузлів операційного пристрою?
3. Визначить поняття: операція, мікроалгоритм, мікрооперація.
4. Що таке мікроалгоритм операції?
5. Визначте основне призначення арифметико-логічного пристр в ЕОМ.

6. Наведіть типи арифметико-логічних пристроїв, та їх основні відмінності.
7. Охарактеризуйте основні етапи проектування арифметико-логічного пристрою з розподіленою логікою.
8. Що відображує операційна схема виконання операції?
9. Що відображує функціональна схема пристрою?
10. В чому відмінність функціонального та структурного мікроалгоритмів?
11. Напишіть вирази, що визначають закони функціонування автоматів Милі та Мура.
12. У чому відмінність автоматів Милі та Мура?
13. Намалюйте узагальнену структурну схему управляючого автомата.
14. Охарактеризуйте основні етапи проектування управляючого автомата.
15. Як перейти від змістовного мікроалгоритму до закодованого мікроалгоритму?
16. Як побудувати граф автомата?
17. Як здійснюється оцінка станів автомата?
18. Як визначити необхідну тривалість управляючих сигналів?
19. Від чого залежить кількість тригерів, необхідних для побудови пам'яті автомата?
20. Як скласти структурну таблицю автомата?
21. Складіть таблицю переходів для *JK*-, *RS*-, *T*- і *D*-тригерів. Наведіть їх умовне графічне позначення.
22. Чи можливий перехід автомата в стан, що непередбачений графом, при використанні тригерів із внутрішньою затримкою (тригерів, керованих рівнем сигналів)?
23. Коли можливе виникнення помилкових управляючих сигналів (що непередбачені графом автомата) і чим визначається їх тривалість?
24. Наведіть способи усунення короткочасних помилкових управляючих сигналів.
25. У чому суть «протигоночного» кодування станів автомата?
26. Як забезпечити перепад управляючого сигналу у випадку, коли операторну вершину з цим сигналом охоплює «петля»?
27. Як визначити час переходу автомата з одного стану в інший?

РОЗДІЛ 3

Синтез блоків мікропрограмного управління

3.1. Призначення та класифікація блоків управління

Блоки управління є складовою частиною пристрою управління, що входить у склад процесору та забезпечує виконання програм ЕОМ.

Основне призначення БУ – формування всіх управляючих сигналів, які забезпечують виконання кожної команди в ЕОМ.

Можна проводити класифікацію БУ за різними ознаками.

За функціональними можливостями:

- БУ із *жорсткою логікою*, призначені для реалізації певного набору мікроалгоритмів; такі БУ реалізують у вигляді управляючих автоматів;
- БУ із *гнучкою логікою*, як правило, *з мікропрограмним управлінням*; такі БУ дозволяють забезпечити модифікацію та заміну нових мікропрограм, змінювати логіку управління в залежності від записаної у пам'яті мікропрограми.

За способами управління розрізняють централізовані та децентралізовані БУ:

- *централізовані* – мікропрограми формуються в одному пристрої для всіх пристроїв у системі. Такі БУ забезпечують виконання всіх МО послідовно у часі, що приводить до зменшення швидкодії системи.
- *децентралізовані* – кожен пристрій у системі має свій БУ, робота яких синхронізує централізований ПУ. Такий спосіб забезпечує виконання мікрооперацій водночас в різних складових частинах системи, що приводить до збільшення швидкодії, але й збільшує апаратурні витрати. У сучасних ЕОМ більш поширені децентралізовані БУ.

Розділяють синхронні та асинхронні БУ.

- *синхронні* – для виконання кожної МО виділяються однакові проміжки часу, що дорівнюють максимальній тривалості МО;
- *асинхронні* – для виконання кожної МО виділяється необхідний для виконання цієї МО проміжок часу.

Слід зазначити, що з точки зору швидкодії асинхронні БУ є більш швидкодіючими, але потребують збільшення апаратної складності.

На практиці застосовуються *комбіновані* БУ – МО поєднуються в групи за часом виконання. В одну групу уключають МО, що найбільш близькі за часом виконання і для всіх МО в цій групі виділяють час що дорівнює максимальній тривалості МО у межах групи.

Приклад 3.1. Для заданого МА тривалість t_i кожної МО y_i задана графічно (рис.3.1), де τ – такт машинного часу. На рис.3.2 наведені часові діаграми виконання заданого МА для різних способів управління.

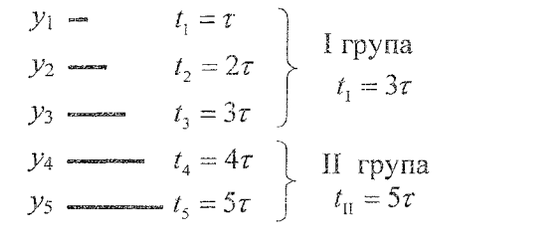


Рис. 3.1. Задана тривалість мікрооперацій

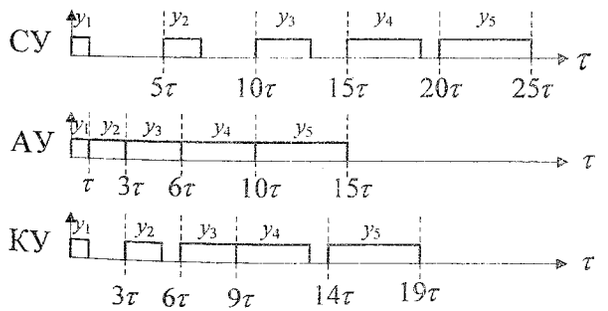


Рис. 3.2. Виконання МО для різних способів управління: СУ – синхронного; АУ – асинхронного; КУ – комбінованого.

3.2. Блоки мікропрограмного управління

Кожній команді, яка записана у основній пам'яті ЕОМ, відповід мікропрограма, що зберігається в пам'яті блоку мікропрограмно управління.

Мікропрограма – це зв'язаний список мікрокоманд, виконан яких забезпечує виконання заданої команди.

Мікрокоманда це інформаційне слово, що містить наступну інф рмацію:

- управляючі сигнали;
- тривалість управляючих сигналів;
- інформацію щодо формування адреси наступної МК.

БМУ функціонує у відповідності з *принципом мікропрограмно управління*, що полягає в наступному.

Під час виконання мікропрограми в кожному такті із постійн пам'яті БМУ зчитується та розшифровується чергова мікрокоманда. результаті виконання мікрокоманди формуються управляючі сигна необхідної тривалості, що поступають на всі функціональні части обчислювальної системи, а також формується адреса наступної мікр команди.

Можна виділити наступні етапи виконання команди в обчислюв льній системі.

1. *Вибірка команди.* З ОП зчитується команда в регістр кома процесора, для чого виконується відповідна МП, що записана у пам'я БМУ.

2. *Розпакування команди.* Команда розшифровується (аналізуют ся поля слова команди, визначаються операнди), що забезпечується в конанням відповідної МП.

3. *Виконання операції.* Виконується МП виконання заданої опер ції над визначеними операндами.

4. *Формування адреси наступної команди.* Відповідна МП фор адресу наступної команди у лічильнику команд.

Спрощена структурна схема БМУ наведена на рис. 3.3.

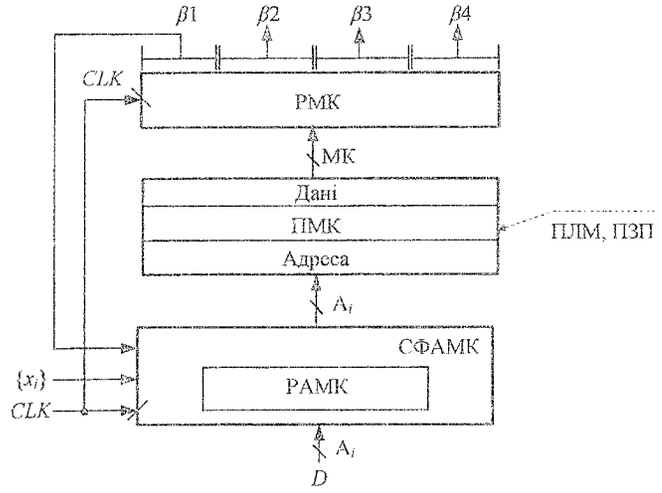


Рис. 3.3. Структурна схема БМУ

Основні функціональні частини БМУ:

РАМК – регістр адреси МК;
СФАМК – схема формування адреси МК;
ПМК – пам'ять МК;
РМК – регістр МК;
A_i – адреса МК;
CLK – синхросигнал;
$\{x_i\}$ – логічні умови;
D – вхід завдання початкової адреси мікропрограми.

МК розміщуються у пам'яті мікрокоманд. На рис. 3.4 наведений формат мікрокоманди.

Рис. 3.4. Формат мікрокоманди

Слово МК складається з чотирьох зон:

$\beta 1$ – зона формування адреси наступної МК;
$\beta 2$ – зона управляючих сигналів;
$\beta 3$ – зона визначення тривалості управляючих сигналів;
$\beta 4$ – зона службових розрядів.

У кожному такті за синхросигналом *CLK* адреса мікрокоманди поновлюється у РАМК і надходить на адресний вхід ПМК. За адресою, що надійшла в ПМК, обирається відповідна мікрокоманда і видається на вихід даних ПМК. Слово мікрокоманди записуються у РМК за зворотним перепадом синхросигналу *CLK*.

Сигнали зони $\beta 2$ управляють вузлами комп'ютера, зони $\beta 3$ – визначають тривалість цих сигналів, сигнали зони $\beta 1$ разом із логічними умовами $\{x_i\}$ поступають на вхід СФАМК і формують адресу наступної МК. За черговим сигналом *CLK* адреса наступної МК буде сформована у РАМК. Зона $\beta 4$ використовується для виконання допоміжних функцій, наприклад контролю апаратури.

3.2.1. Структура зони управляючих сигналів $\beta 2$

Зона $\beta 2$ застосовується для кодування управляючих сигналів. Існують два основні способи кодування управляючих сигналів у зоні $\beta 2$:

- *горизонтальне мікропрограмування*, яке також називають мінімальним кодуванням управляючих сигналів;
- *вертикальне мікропрограмування*, яке також називають максимальним кодуванням управляючих сигналів.

Під час мінімального кодування кожен управляючий сигнал відображується одним розрядом слова мікрокоманди (рис. 3.5), де *R*, *W*, *I*, *O* – відповідно управляючі сигнали зчитування, запису, вводу та виводу інформації, що показані у якості прикладу.

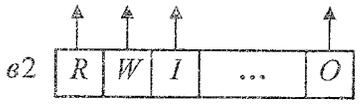


Рис. 3.5. Структура зони $\beta 2$ при мінімальному кодуванні (горизонтальне мікропрограмування)

За мінімального способу кодування довжина зони $\beta 2$ дорівнює

$$n_{\beta 2} = N_{УС}, \tag{3.1}$$

Де $N_{УС}$ – кількість управляючих сигналів.

Під час максимального кодування розряди зони $\beta 2$ формуються за допомогою коду $\alpha_i ... \alpha_1$ на дешифраторі (рис. 3.6).

Довжина зони $\beta 2$ в цьому випадку дорівнює

$$n_{\beta 2} =]\log_2 N_{УС}[. \tag{3.2}$$

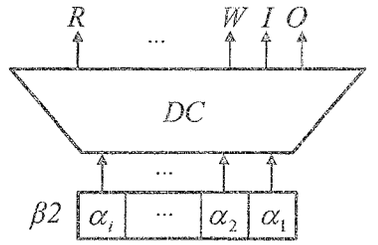


Рис. 3.6. Структура зона $\beta 2$ при максимальному кодуванні (вертикальне мікропрограмування)

Недоліки горизонтального мікропрограмування полягають в тому, що кодування великої кількості УС потребує більшої довжини слова МК. До переваги слід віднести можливість формувати водночас будь яку кількість УС, що приводить до сумісного виконання декількох МО в одному такті.

До переваг вертикального мікропрограмування відносять значне скорочення довжини зони $\beta 2$. Недоліки – в кожному такті можна сформувати тільки один УС, що ускладнює сумісне виконання МО.

На практиці зазвичай використовують *комбінований спосіб кодування* УС – сигнали поєднують у групи, таким чином, що сигнали, які мають бути виконані водночас розміщуються в різних групах. Усередині групи використають ВМ, а між групами ГМ.

На рис. 3.7 наведена можлива реалізація зони $\beta 2$ при комбінованому способі кодування.

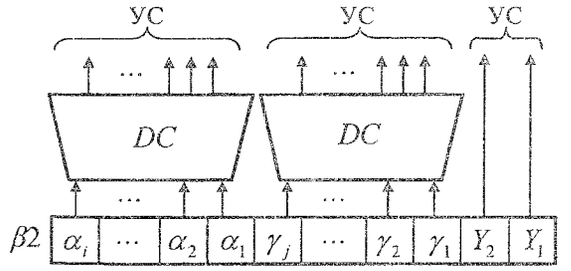


Рис. 3.7. Структура зони $\beta 2$ при комбінованому способі кодування

Приклад 3.2. Побудувати структуру та карту програмування зони управляючих сигналів $\beta 2$ для ефективної реалізації МА, що заданий у вигляді лінійної схеми МА:

$$\Pi\, y_1 y_2 (y_1 y_2 y_3)\, y_4 y_5 (y_2 y_3 y_5)\, y_6 y_7 K$$

Виконання завдання

Для формування зони $\beta 2$ застосуємо комбіноване мікропрограмування. Розподілимо управляючі сигнали на групи так, щоб сигнали які формуються водночас розміщувались у різних групах, отримаємо:

I	II	III
y_1	y_2	y_3
y_5		
y_4		
y_6		
y_7		

Розрахуємо довжину зони $\beta 2$.

Для кодування сигналів першої групи використаємо дешифратор. За виразом (2.2) розрахуємо довжину зони:

$$n_{DC} =]\log_2 6[= 3.$$

Враховуючи, що для кодування сигналів груп II та III необхідно ще два розряди зони $\beta 2$, отримаємо:

$$n_{\beta 1} = 5.$$

Кодування входів дешифратора наведене в табл. 3.1. Карта програмування зони $\beta 2$ – в табл. 3.2.

Таблиця 3.1. Таблиця кодування УС

α_3	α_2	α_1	УС
0	0	0	немає сигналів
0	0	1	y_1
0	1	0	y_5
0	1	1	y_4
1	0	0	y_6
1	0	1	y_7

Таблиця 3.2. Карта програмування

№ такту	УС	$\beta 2$				
		α_3	α_2	α_1	y_2	y_3
1	y_1	0	0	1	0	0
2	y_2	0	0	0	1	0
3	y_1, y_2, y_3	0	0	1	1	1
4	y_4	0	1	1	0	0
5	y_5	0	1	0	0	0
6	y_2, y_3, y_4	0	1	0	1	1
7	y_6	1	0	0	0	0
8	y_7	1	0	1	0	0

Структурна схема зони $\beta 2$ зображена на рис. 3.8.

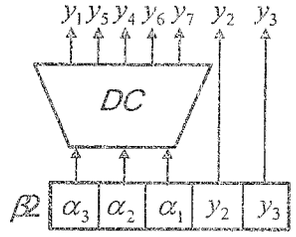


Рис. 3.8. Структурна схема зони $\beta 2$

3.2.2. Структура зони визначення тривалості управляючих сигналів $\beta 3$

Зона $\beta 3$ відповідає за тривалість виконання мікрооперацій. Під час асинхронного та комбінованого способу управління мікрооперація виконується за один або декілька тактів, тобто необхідно забезпечити необхідну затримку управляючого сигналу при виконанні МО.

Найбільш розповсюдженим способом є використання лічильника тактів, у який заноситися константа, що визначає час затримки УС.

У кожному такті виконується декремент лічильника (або інкремент, якщо у лічильнику записане від'ємне число), за нульовим вмістом якого дозволяється зміна інформації в РАМК і відбувається формування наступних УС.

З точки зору апаратної реалізації – частина РМК (зона $\beta 3$) виконується у вигляді лічильника тактів *CT* (рис. 3.9).

$\beta 1$	$\beta 2$	$\beta 3$	$\beta 4$
RG	*RG*	*CT*	*RG*

Рис. 3.9. Апаратна реалізація регістру мікрокоманди

Довжина зони $\beta 3$ визначається за формулою

$$n_{\beta 3} =]\log_2 k[+ 1, \qquad (3.3)$$

де k – максимальна затримка управляючих сигналів у тактах, один додатковий розряд (+1) застосовується для урахування знакового розряду (ЗР).

Структурна схема БМУ з урахуванням зони затримки управляючих сигналів зображена на рис. 3.10.

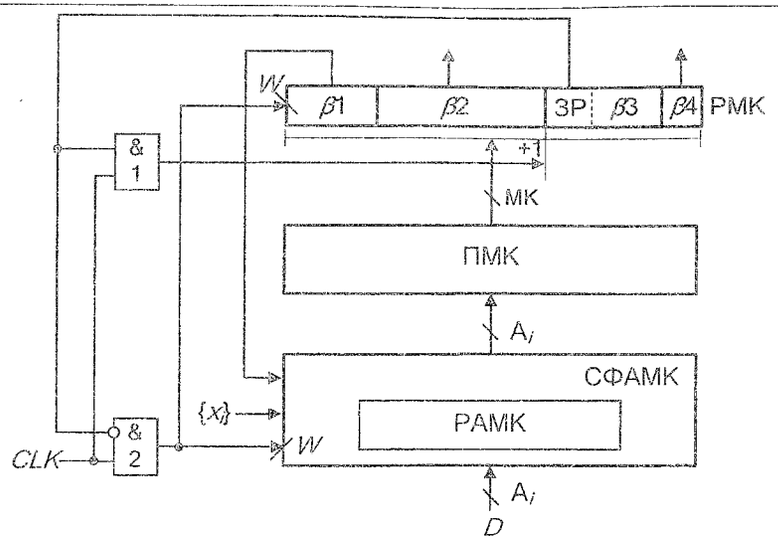


Рис. 3.10. Структурна схема БМУ з урахуванням зони затримки УС

Принцип функціонування. Кількість тактів, на які необхідно затримати МО, записують у двійковому доповнювальному коді від'ємного числа в лічильник *СТ*. Це означає, що в ЗР буде записана одиниця, якщо затримка не дорівнює нулю. У кожному такті до вмісту лічильника *СТ* додається одиниця, поки вміст лічильника не дорівнюватиме нулю. За нульовим кодом відбувається завантаження адреси наступної МК у РАМК. До схеми додаються два логічні елементи І (ЛЕ1 та ЛЕ2). За наявністю одиниці у ЗР лічильника ЛЕ2 блокує сигнал *CLK*, що має поступити на вхід *W* СФАМК. З виходу ЛЕ1 сигнал поступає на вхід (+1) лічильника. За нульовим вмістом лічильника, тобто коли ЗР дорівнює нулю, навпаки є заблокованим ЛЕ1 і сигнал *CLK* поступає на вхід *W* СФАМК, що дозволяє формування адреси наступної МК.

Приклад 3.3. Для БМУ з асинхронним принципом формування управляючих сигналів розробити структуру й карту програмування зони $\beta 3$.

Вихідні дані

- максимальна тривалість МО – 25 тактів,
- карту програмування побудувати для МО тривалістю 5 тактів.

Виконання завдання

Визначимо довжину зони $\beta 3$ за виразом (3.3):

$$n_{\beta 3} =]\log_2 24[+1=5+1=6,$$

де $\Delta = 24\tau$ – максимальна затримка МО.

Якщо тривалість чергової МО $t_i = 5\tau$, то час затримки дорівнюватиме $\Delta = 4\tau$.

Подамо знайдену затримку у двійковому доповнювальному коді від'ємного числа з урахуванням знакового розряду:

$$-4_{10} = 1.00100_2;$$
$$1.00100_{[ПК]} = 1.11100_{[ДК]}.$$

Тоді інформація в зоні $\beta 3$ для карти програмування ПМК має вигляд:

$\beta 3$	1	1 1 1 0 0

Зміна станів лічильника зони $\beta 3$ наведена у табл. 3.3.

Таблиця 3.3. Стани лічильника

№ такту	СТ		Примітки
	ЗР		
ПС	1	11100	ЗР =1 (ЛЕ2 заблокований)
1		+1	
	1	11101	ЗР =1 (ЛЕ2 заблокований)
2		+1	
	1	11110	ЗР =1 (ЛЕ2 заблокований)
3		+1	
	1	11111	ЗР =1 (ЛЕ2 заблокований)
4		+1	ЗР =0 (ЛЕ1 заблокований, формування наступної адреси)
	0	00000	

3.2.3. Призначення зони службових розрядів $\beta 4$

У обчислювальних системах зона $\beta 4$ може складатися із сотні розрядів. Найчастіше цю зону використають для контролю апаратури.

Як приклад можна навести використання зони для контролю слова мікрокоманди на парність або непарність.

Схема контролю має вигляд зображений на рис. 3.11. Для контролю використають операцію згортки (суму за модулем 2). У цьому випадку зона $\beta 4$ має довжину 1 розряд, вміст цього розряду доповнює кількість 1 у слові мікрокоманді до парної (або непарної, при контролі слова МК на непарність).

Під час контролю на парність сигнал «помилка»=1 визначає, що слово МК вміщує непарну кількість 1, тобто наявна помилка.

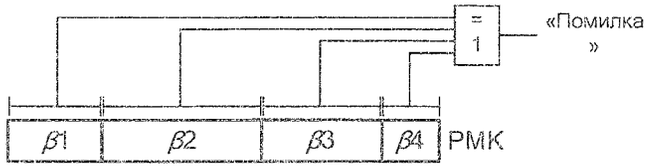


Рис. 3.11. Схема контролю слова МК на парність

Приклад 3.4. Сформувати зону $\beta 4$ для контролю заданих слів МК в зонах $\beta 1$, $\beta 2$ і $\beta 3$ на парність (табл. 3.4).

Виконання завдання

Таблиця 3.4. Кодування зони $\beta 4$

$\beta 1$	$\beta 2$	$\beta 3$	$\beta 4$
01010	1110	1010	1
11001	1101	1110	1
10100	0100	1000	0

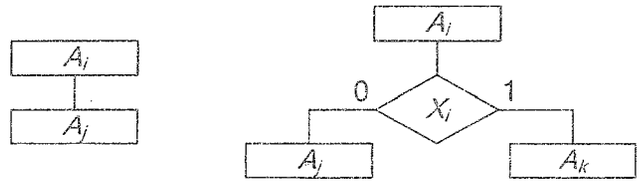
3.2.4. Способи формування адреси МК. Структура зони $\beta 1$

Адресація мікрокоманд у БМУ забезпечується зоною $\beta 1$. Для переходу на наступну МК у зоні $\beta 1$ поточної МК розміщується адреса переходу, або інформація для обчислення адреси переходу. Для виконання розгалуження мікроалгоритмів застосовуються наступні основні конструкції мікроалгоритмів:

- безумовний перехід;
- умовний перехід;
- цикли;
- мікроподпрограми.

Для реалізації безумовного переходу, зона $\beta 1$ мікрокоманди, що розміщується в БМУ за адресою A_i, містить частину або всю адресу переходу A_j. Тобто адреса переходу визначається як $[A_i] \rightarrow [A_j]$. За цим способом формуються адреси на лінійних ділянках МА (рис. 3.12, *а*).

За умовного переходу у зоні $\beta 1$ мікрокоманди A_i вказується інформація щодо адреси A_j або адреси A_k, куди здійснюється перехід в залежності від умови X_i, де X_i – будь-яка умова, що формується поза БМУ (рис. 3.12, *б*).



а *б*

Рис. 3.12. Типові конструкції МА:
а – безумовний перехід; *б* – умовний перехід

Циклічні мікроалгоритми можуть бути організовані двома способами:

- за умовою X_i, що формується поза БМУ;
- за кількістю повторень N, що формується за лічильником циклів усередині БМУ.

Цикли у свою чергу поділяються на:

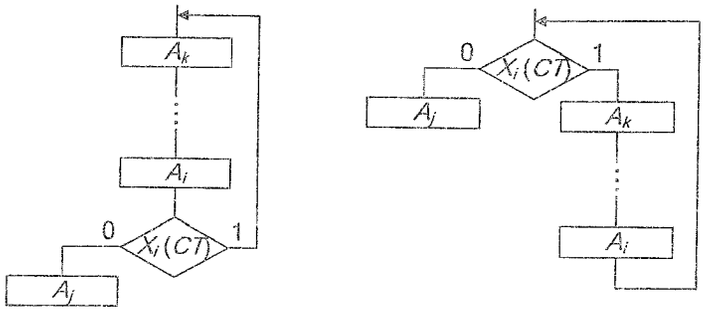
- цикли з перевіркою на вході (рис. 3.13, *а*);
- цикли з перевіркою на виході (рис. 3.13, *б*).

В залежності від умови, що перевіряється на вході/виході циклу, або стану лічильника CT, відбувається вихід з циклу – зона $\beta 1$ містить адресу A_j ($[A_i] \rightarrow [A_j]$), або перехід на початок циклу – зона $\beta 1$ містить адресу A_k ($[A_i] \rightarrow [A_k]$) (рис. 3.13, *а*, *б*).

За способом формування зони $\beta 1$ розрізняють наступні способи адресації мікрокоманд:

- примусова адресація МК;
- відносна адресація МК;
- природна адресація МК (інкрементна).

Для кожного способу адресації мікрокоманд використається відповідний формат зони $\beta 1$.



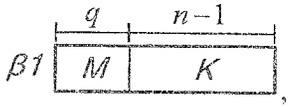
а *б*

Рис. 3.13. Циклічні МА:
а – цикл із перевіркою на виході;б – цикл із перевіркою на вході

3.3. БМУ з примусовою адресацією

3.3.1. Синтез БМУ з примусовою адресацією

За примусової адресації зона $\beta 1$ має наступний формат:



де M – поле управління мультиплексором;
q – довжина поля управління мультиплексором;
K – константа, що визначає адресу наступної мікрокоманди;
n – розрядність адреси мікрокоманди.

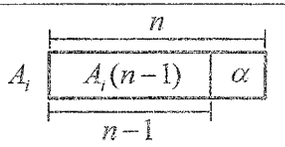
Довжина поля управління мультіплексором визначається за формулою:

$$q =]\log_2(k+2)[, \qquad (3.4)$$

де k – кількість зовнішніх умов.

Поле константи K являє собою $(n-1)$ старших розрядів адреси мікрокоманди.

Формат адреси мікрокоманди має наступний вигляд:



де α – визначає умову переходу, яка формується на виході мультиплексора в залежності від логічних умов X_i.

Спрощена структурна схема БМУ з примусовою адресацією зображена на рис. 3.14. На цій та подальших схемах БМУ входи для занесення початкової адреси D в РАМК умовно не показані.

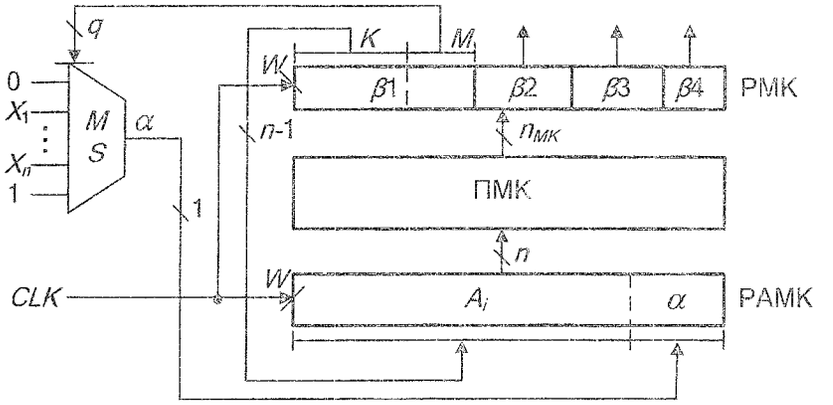


Рис. 3.14. Структурна схема БМУ з примусовою адресацією

Приклад 3.5. Розробити схему БМУ для реалізації заданого мікроалгоритму (рис. 3.15). Розробити карту настроювання БМУ.

Вихідні дані:

- спосіб управління асинхронний;
- ПМК ємністю 32 слова;
- комбіноване кодування зони $\beta 2$;
- забезпечити контроль слова МК на парність;
- час формування управляючих сигналів у тактах: $t(y_1)=1$, $t(y_2)=2$, $t(y_3)=5$, $t(y_4)=1$, $t(y_5)=12$, $t(y_6)=7$, $t(y_7)=9$.

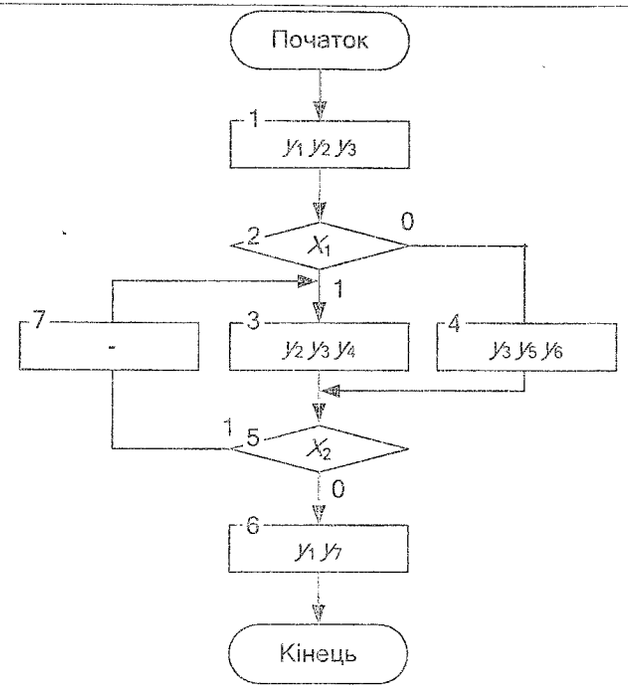


Рис. 3.15. Вихідний мікроалгоритм

Виконання завдання

1. Формат зони $\beta 1$.

Враховуючи, що ємність ПМК дорівнює 32 слова, розрахуємо розрядність адреси:

$$n =]\log_2 32[= 5.$$

Виходячи з розрядності адреси, отримаємо довжину поля константи:

$$K = 5 - 1 = 4.$$

Кількість управляючих входів мультиплексора, що визначає розрядність поля M розрахуємо за виразом (3.4), враховуючи, що кількість управляючих сигналів дорівнює двом (X_1, X_2):

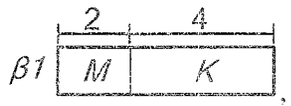
$$q =]\log_2(2+2)[= 2.$$

Складемо таблицю кодування розрядів поля управління мультиплексором (табл. 3.5).

Таблиця 3.5. Кодування поля M

$m_2\,m_1$	УС
00	0
01	X_1
10	X_2
11	1

Враховуючи наступний формат зони $\beta 1$



отримаємо:

$$n_{\beta 1} = 6.$$

2. Формат зони $\beta 2$:

Розподілимо управляючи сигнали за групами, так, що сигнали, які виробляються в одному такті будуть знаходитися у різних групах:

I	II	III
y_1	y_2	y_3
y_5	y_6	
y_4	y_7	

Для формування сигналів першої і другої групи будемо застосовувати дешифратори. Розрахуємо кількість розрядів кодів дешифраторів за виразом (2.2):

$$n_1 = n_1 =]\log_2 3[= 2.$$

Наведемо таблиці кодування сигналів у зоні $\beta 2$ (табл. 3.6 і 3.7).

Таблиця 3.6. Кодування сигналів

$\alpha_2\,\alpha_1$	УС
00	–
01	y_1
10	y_4
11	y_5

Таблиця 3.7. Кодування сигналів

$\gamma_2\,\gamma_1$	УС
00	–
01	y_2
10	y_6
11	y_7

В результаті отримали наступну структуру зони $\beta 2$:

$в2$	α_1	α_2	γ_1	γ_2	y_3

Тоді довжина зони $\beta 2$:

$$n_{\beta 2} = 5.$$

3. Формат зони $\beta 3$.

Максимальна тривалість МО дорівнює: t_{max}=12. Тоді максимальна затримка дорівнює: $\Delta t_{max} = 11$.

За виразом (3.3) розрахуємо довжину зони $\beta 3$:

$$n_{\beta 3} =]\log_2 11[+ 1 = 5.$$

Враховуючи попередні обчислення, а тож те, що для перевірки на парність у зоні $\beta 3$ необхідно виділити один розряд, довжина слова МК дорівнює:

$$n_{МК} = 17.$$

4. Розміщення мікрокоманди у ПМК.

Правило. Мікрокоманди в альтернативних вершинах МА розміщуються у ПМК таким чином, щоб їх адреси відрізнялися лише одним молодшим розрядом (рис.3.16).

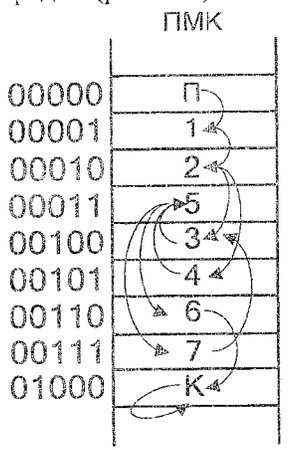


Рис. 3.16. Розміщення мікрокоманд в ПМК

5. Карта програмування наведена у табл. 3.8.

Таблиця 3.8. Карта програмування БМУ

№ МК	Адреса	$\beta 1$		$\beta 2$			$\beta 3$		$\beta 4$
		K	M	$\alpha_2\ \alpha_1$	$\gamma_2\ \gamma_1$	y_3	ЗР		
П	00000	0000	11	00	00	0	0	0000	0
1	00001	0001	00	01	01	1	1	1100	0
2	00010	0010	01	00	00	0	0	0000	1
3	00100	0001	11	10	01	1	1	1100	1
4	00101	0001	11	11	10	1	1	0101	0
5	00011	0011	10	00	00	0	0	0000	1
6	00110	0100	00	01	11	0	1	1000	0
7	00111	0010	00	00	00	0	0	0000	1
К	01000	0100	00	00	00	0	0	0000	1

6. Структурна схема БМУ з примусовою адресацією мікрокоманд, розробленого для реалізації заданого мікроалгоритму, зображена на рис. 3.17.

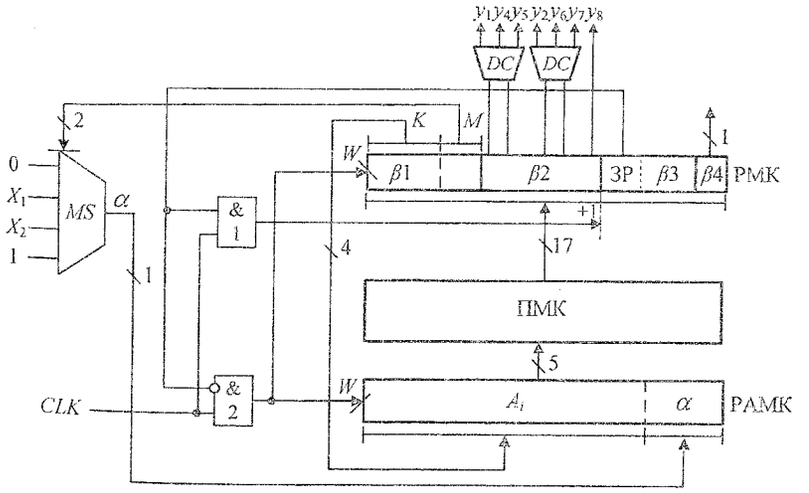


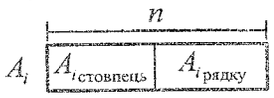
Рис. 3.17. Схема БМУ з примусовою адресацією

3.3.2. Скорочення зони $\beta 1$ для примусової адресації МК

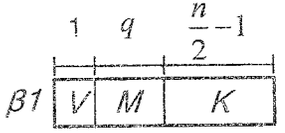
Для скорочення довжини зони $\beta 1$ ПМК розглядають як двомірну матрицю. У цьому випадку розрядність адреси n поділяють на дві частини, одна з яких визначає номер рядку, а друга – номер стовпця, при-

чому, розбіжність у розрядності номерів стовбців й рядків має бути не більш ніж один розряд.

Формат адреси в цьому випадку має такий вигляд:



Формат зони $\beta 1$ для двовимірної ПМК має наступний вигляд:



де V – напрямок переходу: $V = 0$ (за рядком „→"), $V = 1$ (за стовпцем „↓");

M – поле управління мультиплексором, довжиною q розрядів;

K – номер стовпця або рядку;

n – розрядність адреси мікрокоманди.

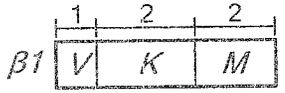
Приклад 3.6. Для БМУ із двовимірної ПМК розробити структуру зони $\beta 1$ і карту програмування для заданого МА (рис. 3.18), якщо ємність ПМК – 64 слова.

Виконання завдання

1. Формат зони $\beta 1$:

$$n_a = \lceil \log_2 64 \rceil = 6;$$
$$n_K = 6:2-1 = 2;$$
$$n_M = \lceil \log_2 3 \rceil = 2;$$
$$n_{\beta 1} = 5.$$

Отримаємо:



2. Розміщення МК у ПМК (рис. 3.19).

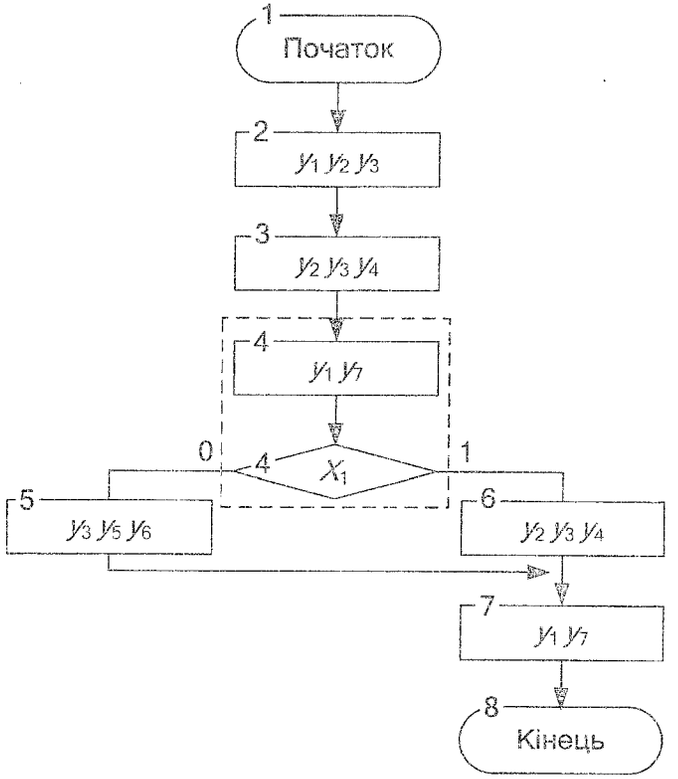


Рис. 3.18. Вихідний мікроалгоритм

ПМК

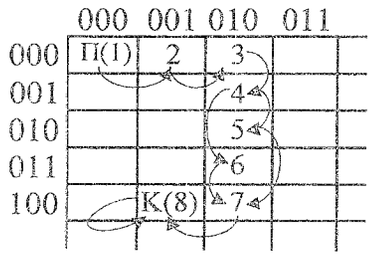


Рис. 3.19. Розміщення мікрокоманд в ПМК

3. Карта програмування зони $\beta 1$ (табл. 3.9).

Таблиця 3.9. Карта програмування БМУ

МК	Адреса МК		$\beta 1$		
	Номер рядку	Номер стовпця	V	K	M
1	000	000	0	00	11
2	000	001	0	01	00
3	000	010	1	00	11
4	000	010	1	01	01
5	010	010	1	10	00
6	011	010	1	10	00
7	100	010	0	00	11
8	100	001	0	00	11

4. Структурна схему БМУ з двовимірною організацією ПМК наведена на рис. 3.20.

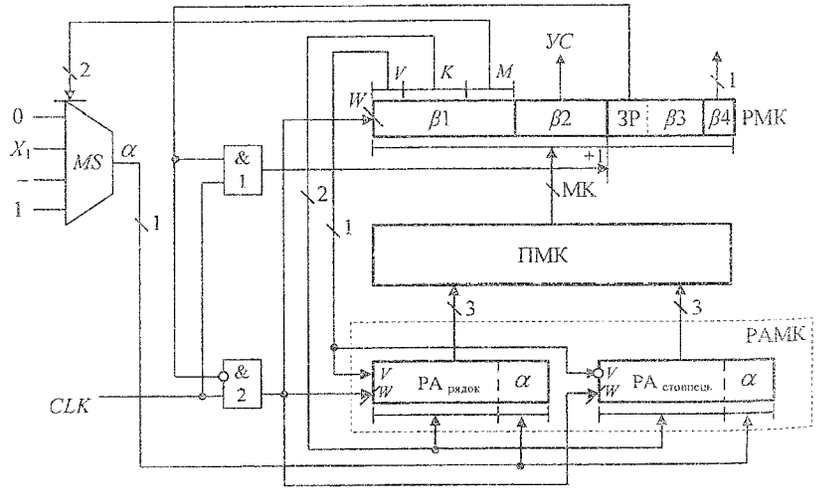


Рис. 3.20. Структурна схему БМУ з матричною ПМК

3.4. БМУ з відносною адресацією

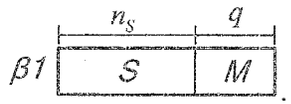
За відносної адресації адреса наступної МК визначається за формулою:

$$A_{i+1} = A_i + S \div \alpha, \tag{3.5}$$

де S – приріст адреси МК;

α – сигнал на виході мультиплексора, що залежить від логічних умов X_i.

Формат зони $\beta 1$ у загальному вигляді:



Довжину поля S визначають за виразом:

$$n_S =]\log_2 N[+ 1, \tag{3.6}$$

де N – максимальний приріст, додатковий знаковий розряд додається для визначення напрямку переходу (зменшення або збільшення адреси).

Приклад 3.7. Побудувати ПМК із відносною адресацією мікрокоманд для мікроалгоритму заданого на рис. 3.21.

Вихідні дані:

- ємність ПМК – 64 слова;
- максимальний перехід – 8 рядків;
- мінімальне кодування управляючих сигналів;
- синхронний спосіб управління.

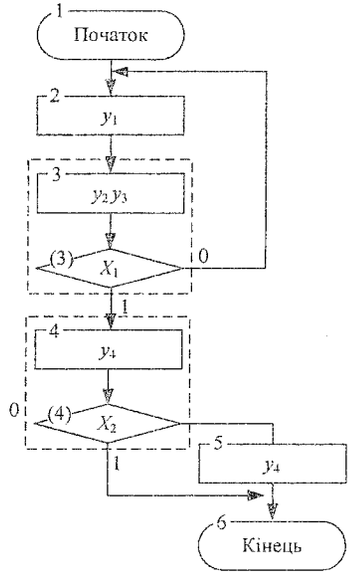


Рис. 3.21. Вихідний мікроалгоритм

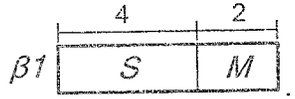
Виконання завдання

1. Формат зони $\beta 1$.

Виходячи з кількості логічних умов та заданого максимального збільшення, за виразами (3.6) та (3.4) відповідно, визначимо:

$$n_M =]\log_2 4[= 2;$$
$$n_S =]\log_2 8[+ 1 = 4;$$
$$n_{\beta 1} = 6.$$

Отримаємо формат зони:



2. Розрядність адреси ПМК:

$$n_a =]\log_2 64[= 6.$$

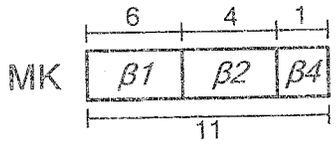
3. При синхронному способі управління для виконання кожної мікрооперації виділяється однакові проміжки часу, тому зону $\beta 3$ не використовуємо.

4. Формат зони $\beta 2$.

При мінімальному кодуванні управляючих сигналів довжина зони $\beta 2$ дорівнює кількості управляючих сигналів:

$$n_{\beta 2} = 4.$$

Враховуючи попередні обчислення отримаємо формат мікрокоманди ($n_{МК} = 11$):



5. Розміщення мікрокоманди у ПМК (рис. 3.22).

Правило. Мікрокоманди, що відповідають альтернативним вершинам мікроалгоритму, розміщуються у ПМК таким чином, щоб їх адреси відрізнялися на одиницю.

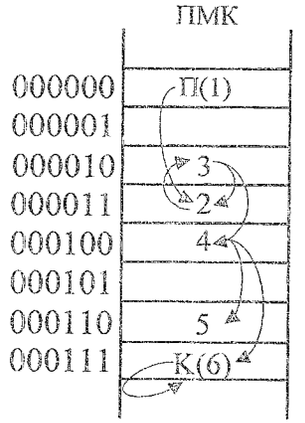


Рис. 3.22. Розміщення мікрокоманд в ПМК

6. Карта програмування зображена у табл. 3.10.

Таблиця 3.10. Карта програмування БМУ

№ МК	Адреса МК	$\beta 1$		$\beta 2$ $y_1y_2y_3y_4$	$\beta 4$
		S	*M*		
1	000000	0011	00	0000	0
2	000011	1111	00	1000	1
3	000010	0001	01	1010	1
4	000100	0010	10	1000	0
5	000110	0001	00	1000	0
6	000111	0000	00	0000	1

7. Структурна схема БМУ наведена на рис. 3.23.

Очевидно, що тривалість такту в даному випадку відносно БМУ з примусовою адресацією збільшується за рахунок затримки сигналів у суматорі. Таким чином, БМУ із примусовою адресацією мають вищу швидкодію, але більшу розрядність зони $\beta 1$ ніж у БМУ з відносною адресацією. Складність БМУ з відносною адресацією зменшується за рахунок скорочення зони $\beta 1$, але це приводить до втрати швидкодії пристрою.

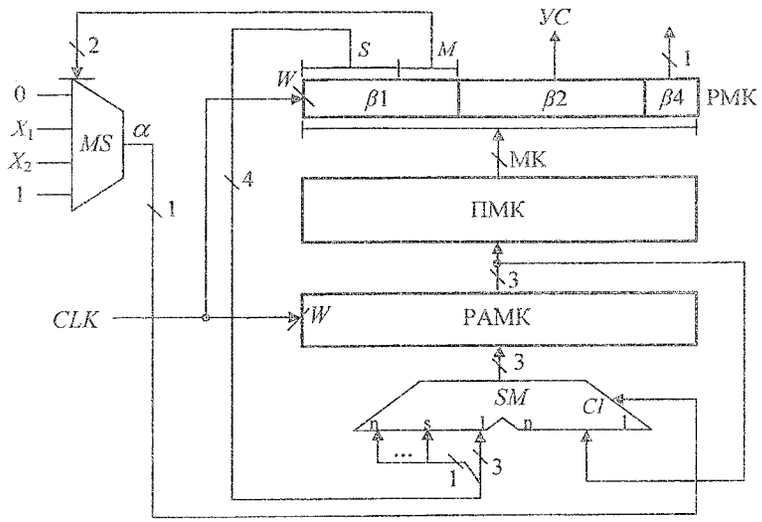


Рис. 3.23. Структурна схема БМУ з відносною адресацією

Приклад 3.8. Побудувати структурну схему БМУ і карту пам'яті мікропрограм для мікроалгоритму виконання операції множення. Мікроалгоритм повинен забезпечувати управління арифметико-логічним пристроєм із розподіленою логікою.

Вихідні дані:

- Спосіб адресації мікрокоманд – примусовий;
- Структура ПМК – лінійна;
- Ємність ПМК – 16 слів;
- Тривалість мікрооперації підсумовування – 4 такти;
- Початкова адреса мікропрограми – 0007*h*;
- Виконати перевірку слова МК на непарність;
- Розрядність операндів – 16 розрядів;
- Розрядність регістрів та суматорів – 8 розрядів.

Виконання завдання

Структурна схема пристрою для виконання операції множення першим способом з урахуванням елементної бази наведена на рис.

3.24. Мікроалгоритм управління роботою пристрою наведений на рис. 3.25. Змістовний МА наведений на рис. 3.26.

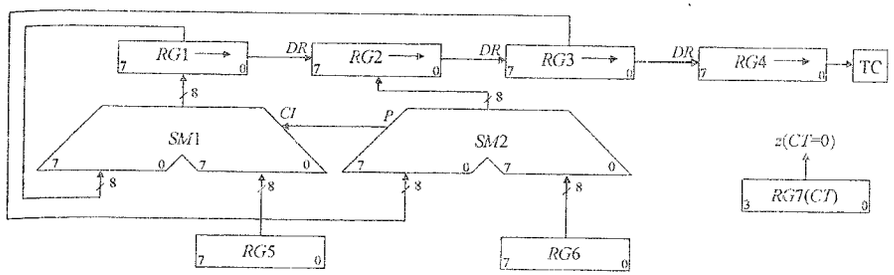


Рис. 3.24. Структурна схема пристрою множення

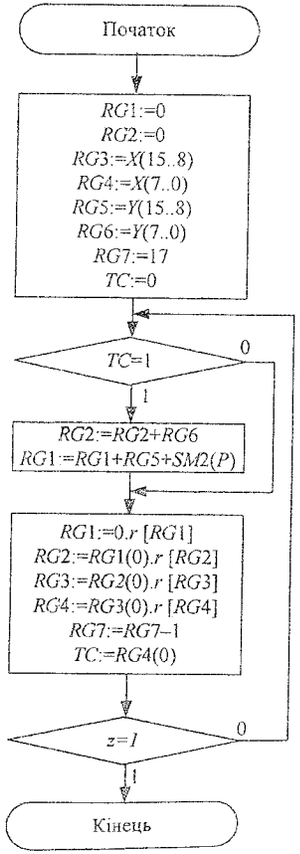


Рис. 3.25. Змістовний мікроалгоритм

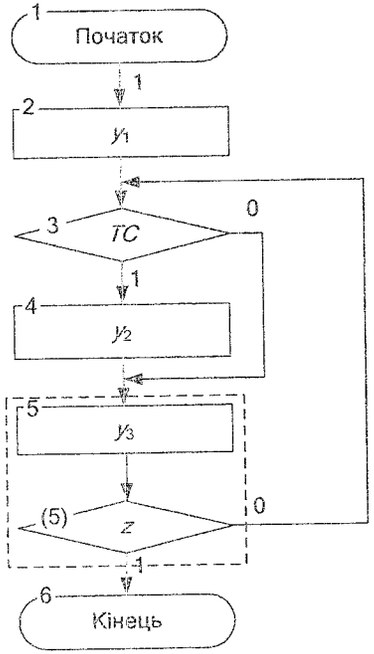


Рис. 3.26. Закодований алгоритм управління пристроєм множення

Визначимо формат зони $\beta 1$:

$$n_a = \lceil \log_2 16 \rceil = 4;$$
$$n_K = 3;$$
$$n_M = \lceil \log_2 4 \rceil = 2;$$
$$n_{\beta 1} = 5.$$

Визначимо спосіб управління мультиплексором (табл. 3.11).

Таблиця 3.11. Кодування поля M

$m_2\, m_1$	УС
00	0
01	TC
10	z
11	1

Визначимо формат зони $\beta 2$. Для максимального способу кодування управляючих сигналів розрахуємо розрядність коду дешифратора за виразом (3.2):

$$n_{\beta 2} = \lceil \log_2 4 \rceil = 2.$$

Наведемо кодування сигналів у зоні $\beta 2$ (табл. 3.12).

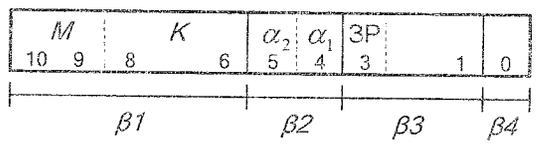
Таблиця 3.12. Кодування сигналів

$\alpha_2\, \alpha_1$	УС
00	–
01	y_1
10	y_2
11	y_3

За виразом (3.3) розрахуємо довжину зони $\beta 3$:

$$\Delta t_{\max} = 3;$$
$$n_{\beta 3} = \lceil \log_2 3 \rceil + 1 = 3.$$

Для перевірки на парність у зоні $\beta 4$ необхідно виділити один розряд. Отримаємо наступний формат мікрокоманди ($n_{\text{МК}} = 10$):



Розміщуємо мікрокоманди в пам'яті мікрокоманд (рис. 3.27).

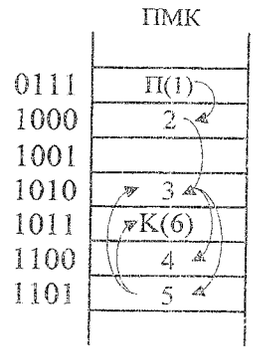


Рис. 3.27. Розміщення мікрокоманд в ПМК

Карта програмування БМУ наведена у табл. 3.13.

Таблиця 3.13. Карта програмування БМУ

№ МК	Адреса	$\beta 1$		$\beta 2$	$\beta 3$		$\beta 4$
		K	M	$\alpha_2\ \alpha_1$	ЗР		
П(1)	0111	100	00	00	0	00	0
2	1000	101	00	01	0	00	0
3	1010	110	01	00	0	00	1
4	1100	110	11	10	1	01	1
5	1101	101	10	11	0	00	0
К(6)	1011	101	11	00	0	00	1

Структурна схема БМУ із лінійною ПМК та примусовим способом адресації зображена на рис. 3.17.

3.5. Лабораторна робота 2

Ціль роботи: Дослідити засоби побудови блоків мікропрограмного управління. Одержати навички в проектуванні й налагодженні схем пристроїв управління з мікропрограмним управлінням.

Підготовка до роботи

1. Побудувати структурну схему БМУ і карту пам'яті мікропрограм для мікроалгоритму виконання операції множення. Мікроалгоритм повинен забезпечувати управління арифметико-логічним пристроєм із розподіленою логікою відповідно до варіанту лабораторної роботи 1 (БМУ повинен замінити управляючий пристрій із жорсткою логікою).

2. Під час виконання завдання необхідно враховувати дані наведені у табл. 3.14 – 3.15.

Таблиця 3.14. Вихідні дані до проектування

a_4	a_2	Спосіб адресації мікрокоманд	Структура ПМК	Ємність ПМК (слів)	Використати зону $\exists_4$ для перевірки слова МК
0	0	примусовий	лінійна	64	на непарність
0	1	примусовий	матрична		на парність
1	0	відносна	лінійна		на непарність
1	1				на парність
Спосіб мікропрограмування – горизонтальний;					
Забезпечити занесення початкової адреси мікроалгоритму в регістр адреси мікрокоманд.					

Таблиця 3.15. Вихідні дані до проектування

a_6	a_5	a_4	Тривалість мікрооперації підсумовування	Початкова адреса мікропрограми
0	0	0	7	18*h*
0	0	1	4	0*ah*
0	1	0	3	06*h*
0	1	1	6	0*eh*
1	0	0	11	13*h*
1	0	1	4	07*h*
1	1	0	5	11*h*
1	1	1	2	0*bh*

Виконання роботи

1. Використовуючи моделюючу систему ПРОГМОЛС 2.0 побудувати і налагодити БМУ. Опис програмного комплексу ПРОГМОЛС 2.0 наведений у додатку М.

2. На спроектованому пристрої виконати числовий приклад із заданими викладачем значеннями операндів.

3. У протоколі навести функціональну схему пристрою для виконання операції множення.

Зміст звіту

Звіт з лабораторної роботи повинен включати короткі теоретичні відомості, необхідні для виконання лабораторної роботи; структурні та функціональні схеми; таблиці та діаграми, отримані при виконанні тео-

ретичного завдання, а також у процесі моделювання схем; висновки за роботою.

Контрольні питання

1. Наведіть загальну конструктивно-функціональну структуру ЕОМ, пояснити загальне призначення БМУ та АЛП.
2. Наведіть порівняльну характеристику АЛП з розподіленою та зосередженою логікою.
3. Приведіть етапи побудови АЛП із розподіленою логікою.
4. Визначіть призначення АЛП у ЕОМ. Наведіть класифікацію АЛП.
5. Визначіть призначення БМУ у ЕОМ, наведіть класифікації БМУ.
6. Поясніть, що розуміють під принципом мікропрограмного управління?
7. Наведіть класифікацію БМУ з точки зору забезпечення тривалості виконання мікрооперацій. Наведіть недоліки і переваги кожного із способів.
8. Як забезпечується тривалість виконання мікрооперацій при асинхронному способі управління виконанням МК у БМУ?
9. Наведіть формат слова мікрокоманди і поясніть призначення кожної із зон.
10. Як визначити довжину зони $\beta 2$ при горизонтальному способі мікропрограмування?
11. Назвіть способи формування структури зони $\beta 2$, переваги та недоліки кожного із способів.
12. Як визначити довжину зони $\beta 3$ формування управляючих сигналів БМУ при асинхронному способі управління виконанням МК?
13. Назвіть способи формування структури зони $\beta 1$, переваги та недоліки кожного із способів.
14. Як скоротити довжину зони $\beta 1$ при застосуванні примусової адресації МК?
15. Наведіть приклад застосування зони $\beta 4$.

РОЗДІЛ 4

Проектування пристроїв з мікропрограмним управлінням для виконання арифметичних операцій

4.1. Операція ділення

Існують два основних методи ділення чисел:

- з відновленням від'ємного залишку;
- без відновлення від'ємного залишку.

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 більш ефективним, тому надалі будемо розглядати саме цей метод.

Нехай ділене X і дільник Y є n-розрядними правильними дробами, поданими в прямому коді. В цьому випадку знакові й основні розряди операндів обробляються окремо. Знак результату визначається шляхом підсумовування за модулем два 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 2^0, дорівнює 1, а при $R_0 < 0$ – дорівнює 0. Різниця R_0 є залишком.
2. Подвоїти залишок (тобто одержати значення $2R_i$).
3. При $2R_i < 0$, додати Y, в зворотному випадку, якщо $2R_0 \geq 0$, відняти Y. Якщо знову отриманий залишок $R_{i+1} \geq 0$, то $Z_{i+1} = 1$, інакше $Z_{i+1} = 0$.
4. Повторити дії описані в пунктах 2 та 3 $(n - 1)$ раз.

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

Під час реалізації ділення *за першим способом* здійснюється зсув вліво залишку при нерухомому дільнику, такий спосіб називається *діленням із зсувом залишку*. На рис. 4.1 показаний принцип побудови пристрою для ділення чисел. Черговий залишок формується в регістрі