ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ТЕЛЕКОМУНІКАЦІЙ
ТА ІНФОРМАТИЗАЦІЇ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра Комп'ютерної інженерії

КОМП'ЮТЕР ТА КОМП'ЮТЕРНА АРИФМЕТИКА

Навчальний посібник з дисциплін «Архітектура комп'ютерів» та «Комп'ютерна логіка» для студентів Навчально-наукового інституту Телекомунікації та інформатизацації
та Навчально-наукового інституту заочного та дистанційного навчання, що навчаються за спеціальністю 123 «Комп'ютерна інженерія»

Київ 2016

УДК 681.127

Укладачі:
В.М. Чегренець, Н.В. Руденко
Рецензенти:
Доктор технічних наук, професор С.Я. Жук (професор кафедри РТПС радіотехнічного факультету НТУ України "КПІ");
Доктор технічних наук, професор В.В. Вишнівський (Державний університет телекомунікацій, завідувач кафедри Інформаційних технологій).

Схвалено Вченою радою Державного університету телекомунікацій (протокол №18 від 06.07.2016 р.).

Чегренець В.М., Руденко Н.В. Комп'ютер та комп'ютерна арифметика. – К.: Державний Університет Телекомунікацій, Навчально-науковий Інститут Телекомунікацій та Інформатизації, 2016. – 120 с.

Навчальний посібник містить основи теорії з дисциплін «Архітектура комп'ютерів» та "Комп'ютерна логіка" для студентів Навчально-наукового Інституту Телекомунікацій та Інформатизації та Навчально-наукового Інституту заочного та дистанційного навчання, що навчаються за спеціальністю 123 «Комп'ютерна інженерія».

Представлений в посібнику матеріал відповідає вимогам робочої програми навчальної дисципліни «Комп'ютерна логіка» і навчальним планам і дозволяє охопити основні розділи даної дисципліни.

Для кожної з частин визначені короткі теоретичні положення та контрольні питання, що дозволяють оцінити знання студентів по кожному розділу дисципліни, що вивчається.

Крім того, посібник корисно для студентів інших спеціальностей: «Комп'ютерна інженерія», «Інженерія програмного забезпечення», «Системний аналіз», «Безпека інформаційних і комунікаційних систем» («Кібербезпека»), а також аспірантів і здобувачів наукового ступеня кандидата технічних наук за відповідними спеціальностями при вивченні ними інформаційно-комп'ютерних систем, автоматизації проектування комп'ютерних систем.

Призначено для студентів Навчально-наукового інституту Телекомунікацій та інформатизації та Навчально-наукового інституту Заочного та Дистанційного навчання за спеціальністю 123 «Комп'ютерна інженерія».

Зміст

ВСТУП

Вступаючи в XXI століття, людство відкриває новий етап розвитку – інформаційний, для якого характерним є домінуюча роль інформаційних ресурсів. Інформація – це не лише сукупність відомостей, фактів або даних, як це представлялося раніше (у тому числі і в законодавчій базі). Інформація – це продукт динамічної взаємодії об'єктивних та суб'єктивних даних. Такий підхід лежить в основі адекватного розуміння суті інформаційного ресурсообміну. Це не просто обмін відомостями у формі їх публікації (оголошення, мовлення). Це ще і сучасні засоби зв'язку, і засоби обчислювальної техніки, і програмні засоби інформаційних технологій.

Інформатика в XXI столітті природною наукою, що займає положення між іншими природничими, технічними і громадськими науками.Її предмет складають інформаційні процеси, що протікають у природі, суспільстві та технічних системах. Її методи у своїй більшості засновані на взаємодії програмних і апаратних засобів обчислювальної техніки з іншими технічними системами, з людиною і суспільством. Її мета – наукове обґрунтування ефективних прийомів створення, розподілу і споживання усіх типів інформаційних ресурсів і методологічне забезпечення розробки нових інформаційних систем. Її центральна роль полягає в наданні свого апарату і понятійної бази іншим природничим, громадським і технічним дисциплінам.

Інформаційна сфера і рівень розвитку інформаційних технологій стають вирішальними показниками розвиненості держави.

Особливо складним і проблемним питанням залишається інформатизація інтелектуальної, творчої діяльності посадовців органів управління, відповідальних за прийняття рішнн

Мета навчального посібника:

-формування знань щодо комп'ютерної логіки, як математичної практичної бази побудови сучасної комп'ютерної техніки та технологій, включаючи програмне та апаратне забезпечення;

-отримання знань основних положень та означень комп'ютерної логіки та інформаційних основ комп'ютерної техніки та алгебри перемикальних функцій;

-розкрити суть і можливості основ теорії цифрових автоматів з пам'яттю і методів синтезу цифрових автоматів з пам'яттю;

-розкрити форми подання та кодування чисел в комп'ютерах;

-пояснити операції з числами в форматі з фіксованою комою та операції з числами в форматі з плаваючою комою;

-визначити цифрові автомати як основу побудови комп’ютерів;

-розробляти алгоритми функціонування автоматів з пам’яттю, робити їх формалізований опис із застосуванням різних мов програмування;

-виконувати абстрактний та структурний синтез автоматів з використанням теорії часових функцій та композиції елементарних автоматів;

Викладений матеріал дозволяє навчитися розробляти алгоритми виконання основних арифметичних та алгебраїчних операцій з числами, що подані в форматі з фіксованою комою та в форматі з плаваючою комою;

- розробляти на функціональному рівні операційні автомати, що реалізують задані алгоритми перетворення даних;
- виконувати порівняльний аналіз різних технічних рішень.

Викладений матеріал направлений на формування знань про принципи подання чисел у різних системах числення, визначати властивості систем та застосовувати способи перетворення чисел із однієї системи числення в другу.

Навчальний посібник складається з чотирьох частин та частини 5 з контрольними питаннями, з шістьма додатками та списком літератури.

У першій частині висвітлені основні положення і означення пристроїв – сучасних комп’ютерів та загальні принципи побудови комп’ютерів на базі керуючих автоматів зі схемною логікою та програмованою логікою.

У другій частині розглядаються структура та принципи організації складових сучасних персональних комп’ютерів.

У третій частині розкриваються комп’ютерні аспекти теорії систем числення в комп’ютерній арифметиці. Окремо розглядаються форми представлення чисел з фіксованою комою.

Четверта частина висвітлює алгоритми виконання арифметичних операцій. Наводяться алгоритми додавання і віднімання чисел з фіксованою комою в прямих і доповняльних кодах. Розглядаються алгоритми додавання і віднімання чисел з плаваючою комою. Наводяться алгоритми прискореного комп’ютерного множення та основні алгоритми комп'ютерного ділення. Додатково розглядаються похибки подання чисел і арифметичних операцій з числами різних систем числення.

Автори особливо вдячні студентам КДУТ в перевірці та оформленні алгоритмів окремих арифметичних операцій з відповідними системами числення: Козенко Б.В. – в розділі 3.4, Євсєєв К.В. – в розділі 4.6, Хобта Б.М. – в розділі 4.8.

Частина 1. Основні положення та означення пристроїв сучасних персональних комп'ютерів

1.1. Загальні принципи побудови комп'ютерів на базі керуючих автоматів зі схемною логікою та програмованою логікою

Термін "автомат" використовують двояко. У техніці з поняттям "автомат" зв'язують деякий пристрій, здатний виконувати певні функції без втручання людини або з його обмеженою участю. У іншому, широкому аспекті ,автомат — це математична модель, що відображає фізичні або абстрактні явища найрізноманітнішої природи(обчислювальні машини, системи управління і зв'язку, лінгвістика). Універсальність теорії автоматів дозволяє розглядати з єдиної точки зору різні об'єкти, встановлювати зв'язки і аналогії між ними, переносити результати досліджень з однієї області в іншу. Узагальненим прикладом цифрового автомата є комп'ютер, виконуючий прийом, зберігання і перетворення дискретної інформації по заданих алгоритмах.

Загальна теорія автоматів підрозділяється на **абстрактну** і **структурну. Абстрактна** теорія вивчає поведінку автомата щодо зовнішнього середовища і не розглядає способи його побудови. **Структурна** теорія автоматів вивчає способи побудови логічних схем автоматів на основі алгоритму, заданого на абстрактному рівні.

Абстрактний автомат як систему задають впорядкованою сукупністю шести об'єктів ***{X, Y, Z, λ,δ,z1}***, де:

X *= {x1, x2,xn}* — множина вхідних сигналів;

Y *= {y1,y2, ...,yn}* — множина вихідних сигналів;

Z *= {z1,z2…zn}*— множина внутрішніх станів, визначених пам'яттю автомата;

δ – функція переходів*;*

λ – функція виходів;

z1 – початковий стан автомата.

Множини ***X, Y, Z*** називаються алфавітами, а їх елементи – буквами. Послідовності вхідних і вихідних букв утворюють відповідно вхідні і вигідні слова.

Поняття "пам'ять" автомата введено для опису систем, сигнали на виходах яких залежать як від вхідних сигналів в даний момент часу, так і від попередньої історії розвитку процесу. Загалом безліч стійких станів автомата — це сукупність поточних значень фізичних параметрів елементів пам'яті

(наприклад, напруга на виходах тригерів), яка зберігається до надходження відповідного вхідного сигнал.

Автомат працює в дискретному часі і перехід з одного стану в інший відбувається миттєво. За способом введення дискретного часу автомати діляться на синхронних і асинхронних. У синхронних автоматах дискретний час задають генератором синхросигналів: t = 0,1, 2, де t – номер машинного такту. В асинхронних автоматах моменти переходу з одного стану в інший заздалегідь не визначені і залежать від деяких подій. У таких автоматах інтервал дискретності змінний. У теорії найповніше описані і на практиці широко застосовуються синхронні автомати. Автомат, що має початковий стан, називається **ініціальним**. У початковий момент часу t = 0 ініціальний автомат завжди знаходиться в початковому стані $z1 = Z$.

За способом формування вихідних сигналів розрізняють автомати Мілі, Мура і С-автомати. Функція переходів всіх автоматів однакова і записується у вигляді:

$$Z(t) = \delta[X(t),Z(t-1)]$$

Функції виходів задають вирази:

$Y(t) = \lambda[X(t),Z(t-1)]$ — для автоматів Мілі;

$Y(t) = \lambda[Z(t)]$ — для автоматів Мура;

С-автомат об'єднує властивості автоматів Мілі і Мура.

Абстрактні автомати Мілі і Мура характеризуються одноканальними входами і виходами (рис.1).

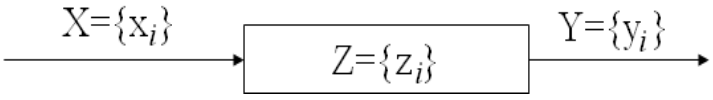


Рис. 1. Схеми абстрактних автоматів Мілі і Мура

Функція переходів δ показує всі можливі переходи з одного стану пам'яті *zi* в інший *zj* під дією вхідних сигналів. Функція виходів λ задає всі можливі вихідні сигнали, що виробляються автоматом в дискретний момент часу в залежності від $x_i(t)$ і $z_i(t)$.

У автоматах Мілі вихідні сигнали є функцією вхідних сигналів стану пам'яті. В автоматах Мура вихідні сигнали визначаються тільки станом пам'яті.

Автомат називається **кінцевим**, якщо його множини X, Y і Z кінцеві; інакше автомат є нескінченним. У кінцевому автоматі перехід з одного стану в будь-який інший закінчується за кінцеве число тактів.

Ряд процесів, якими управляють автомати, не вимагають для своєї роботи попередньої історії, в них вихідний сигнал $y_j(t)$ визначається тільки вхідним сигналами $x_i(t)$. Такі автомати мають тільки один стан, їх задають трійкою X,λ,Y де функція виходів λ – це відображення виду $y(t) = \lambda [x(t)]$. Автомати з одним внутрішнім станом називаються **автомати без пам'яті** або **комбінаційними схемами.**

На рівні абстрактної теорії функціонування автомата розглядається як перетворення вхідних букв (слів) у вихідні букви (слова).

Абстрактний автомат можна задавати за допомогою таблиць переходів і виходів, графів, матриць з'єднань або аналітичним способом. Абстрактний автомат навивається **повним,** якщо його функції переходів визначені для всіх пар (***Xi, Zj***) і частковим – в іншому випадку.

В таблиці переходів и виходів повного автомату Мілі рядки і стовпці позначені буквами вхідних сигналів і стан пам'яті. В комірках таблиці переходів на перетині рядків x_i і стовпця **zj** записується новий стан – результат переходу $Zk = \delta(xi,zj)$, а в таблиці виходів – вихідний сигнал $ym = \lambda(xi,zj)$ Для повного автомату Мілі з довільними множинами $X = \{x_1,x_2,x_3\}, Y = \{y_1,y_2,y_3\}$ і $Z = \{z_1,z_2,z_3\}$ функція переходів представлена в табл. 1, а функція виходів в табл. 2. Часто при представленні автомату Мілі використовують одну таблицю переходів и виходів (табл. 3).

Таблиця 1

X	Z			
	z_1	z_2	z_3	z_4
x_1	z_1	z_3	z_3	z_3
x_2	z_2	z_4	z_1	z_1
x_3	z_3	z_2	z_4	z_2

Таблиця 2

X	Z			
	z_1	z_2	z_3	z_4
x_1	y_1	y_3	y_2	y_2
x_2	y_1	y_3	y_1	y_3
x_3	y_2	y_3	y_3	y_3

Таблиця 3

X	Z/Y			
	z_1	z_2	z_3	Z_4
x_1	z_1/y_1	z_3/y_3	z_3/y_2	z_3y_2
x_2	z_2/y_1	z_4/y_3	z_1/y_1	z_1/y_3
x_3	z_3/y_2	z_2/y_3	z_4y_3	z_2y_3

Вигляд таблиці переходів не залежить от типу автомата. В частковому автоматі Мілі з довільними множинами ***X, Y*** і ***Z*** для невизначеного переходу ставиться прочерк, і будь-яка вхідна буква, що приводить до цього, заборонена. У комірках таблиці виходів часткового автомата Мілі може стояти прочерк, що означає відсутність вихідного сигналу. При цьому прочерк обов'язково ставиться в тих клітках таблиці виходів, які відповідають таким же кліткам з прочерком в таблиці переходів. Відмічена таблиця переходів часткового автомата Мілі зведена в табл. 4

Таблиця 4

X	Z/Y			
	z_1	z_2	z_3	z_4
x_1	z_2/y_1	–	z_3/y_3	–
x_2	–	z_4/y_2	$z_1/-$	z_2/y_3

У таблиці виходів повного автомата Мура з безліччю **X = {x1,x2}, Y = {y1,y2,y3}** і **Z = {z1,z2,z3, z4}** кожному стану пам'яті приписують свій вихідний сигнал, не залежний від букв вхідного алфавіту. Перевірки в деяких комірках таблиці виходів часткового автомата не пов'язані з прочерком в його таблиці переходів. Відмічена таблиця довільного часткового автомата Мура приведена в табл. 5.

Таблиця 5

X	Z/Y			
	z_1	z_2	z_3	z_4
	y_1	y_2	y_2	y_3
x_1	z_2	—	z_3	—
x_2	—	z_4	z_1	z_2

При графічному способі опису абстрактний автомат Мілі представляють орієнтованим графом, в якому стані зображуються вершинами графа, а переходи між станами дугами з позначенням вхідного і вихідного сигналів. Приклад графа автомата Мілі з відміченою табл. 3 показаний на рис. 2,а. При уявленні графом абстрактного автомату Мура вихідні сигнали записуються поряд з вершинами станів. Приклад автомату Мура, заданого табл. 5, показаний на рис. 2,б.

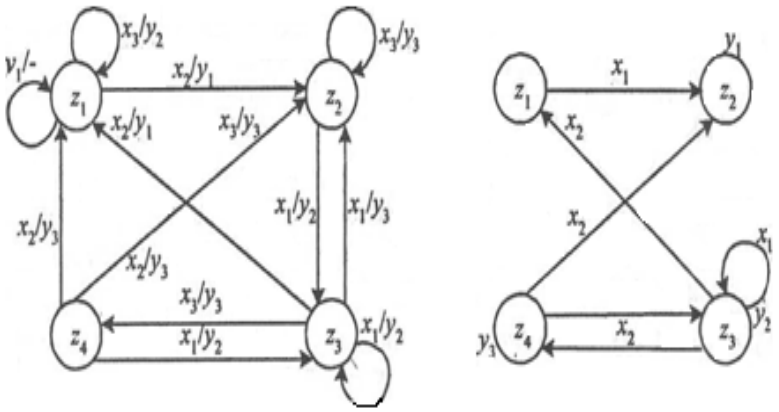


Рис. 2. Графи автоматів Мілі і Мура.

1.2. Структурний синтез керуючого автомата зі схемною логікою

Академік В.М. Глушков розробив канонічний метод структурного синтез; цифрових пристроїв, в якому закон функціонування абстрактного автомату реалізується комбінаційною схемою і набором тригерів. Процес побудови такої схеми називають **структурним синтезом.**

Набір тригерів і логічних елементів являється структурно повним, якщо на їх основі можна побудувати будь-який автомат. Для цього набір повинен містити функціонально повну систему логічних елементів і хоча б один тригер з повною системою переходів і виходів.

При структурному синтезі автоматів використовують ***RS-, JK-, D-*** і ***T***-тригери, котрі являються елементарними автоматами Мура з повними системами переходів і виходів. Вони мають два внутрішніх стани, котрим відповідають два різних сигнали. Це дає можливість, позначати стани, вхідні і вихідні сигнали тригерів двохзначним структурним алфавітом (символами 0 і 1), так показано в табл. 6 для основних типів тригерів.

Таблиця 6

RS-тригер				*JK*-тригер				*D*-тригер			*T*-тригер		
R	S	Q		J	K	Q		D	Q		T	Q	
		0	1			0	1		0	1		0	1
0	0	0	1	0	0	0	1	0	0	0	0	0	1
0	1	1	1	0	1	0	0	1	1	1	1	1	0
1	0	0	0	1	0	1	1						
1	1	-	-	1	1	1	0						

Кінцеві автомати Мілі і Мура являються основою для побудови керуючих автоматів із схемною логікою. Для побудови структурного автомата необхідно мати пам'ять і дві комбінаційні схеми: КС1 – для вироблення функцій збудження, КС2 – для формування вихідних керуючих сигналів (рис. 3).

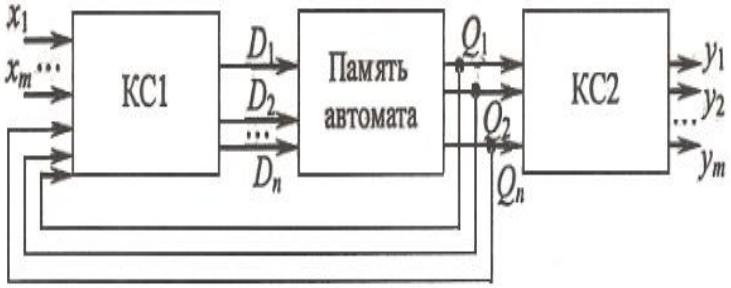


Рис. 3. Схема керуючого автомата із схемною логікою

Структурний синтез МПА з схемною логікою містить наступні етапи:

1. Розробка мікропрограми операції і запис її на мові мікрооперацій.
2. Побудова змістовного графа мікропрограми.
3. Побудова закодованого графа мікропрограми (рис. 4,а).

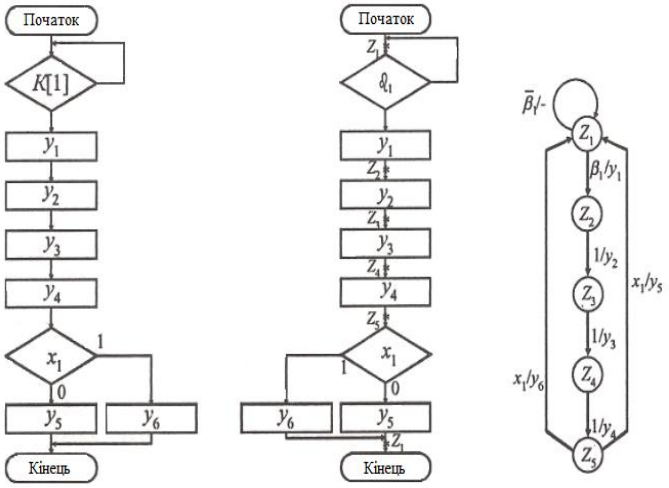


Рис. 4. Схеми графа мікропрограм

4. Для отримання числа стану пам’яті ***N*** автомата виконують розмітку закодованого графа. Розмітку графа мікропрограми для автомата Мілі виконують згідно з правилами (рис. 4,б):

- символом стану z_1 позначають вихід вершини "Початок" і вхід вершини "Кінець"; виходи операторних вершин позначають символами $z_2, z_3 \dots z_N$, де індекс N визначає максимальне число станів пам'яті автомата Мілі (рис. 4,б).

5. Будують граф автомата Мілі (рис. 4,в):

- зображають N вершин $z_1, z_2 \dots z_N$;

- шлях між двома вершинами включає одну операторну і довільне число умовних вершин і зображується орієнтованою дугою. Між двома вершинами може бути декілька шляхів і відповідно – орієнтованих дуг;

- біля дуги записують кон'юнкцію довільного числа сигналів x_i, записаних в умовних вершинах на даному шляху розміченого графа;

- для виходу з умовної вершини, позначеною одиницею, записують x_i, а позначеною нулем – $\overline{x_i}$;

- біля сигналів умов на дузі записують перелік керуючих сигналів y_i ;

за відсутності на шляху умовних вершин замість них записують одиницю;

- у випадку відсутності керуючих сигналів записують прочерк.

6. Розмічаючи граф мікропрограми для автомата Мура, символом z_1 позначають вершини "Початок" і "Кінець". Всі інші операторні вершини позначають символами $z_2, z_3, \dots z_L$, де L – максимальне число станів пам'яті. Кожному стану z_i приписують свій набір вихідних сигналів (рис. 5,а). На дугах графа переходів автомата Мура записують тільки сигнали логічних умов, а сигнали керування проставляють біля вершин графа (рис. 5,б).

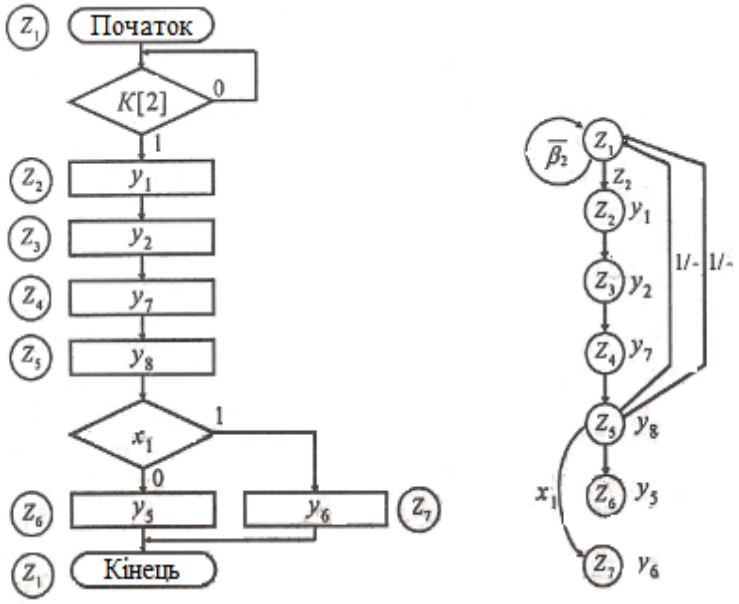


Рис. 5. Граф мікропрограми для автомата Мура

7. Кодування станів пам'яті автомата виконується двійковими наборами сигналів $Q_n, Q_n\text{-}1, \dots, Q_1$ з виходів тригерів. При кодуванні стану позиційним кодом кількість тригерів $\mathbf{n =]\log_2 N[}$ (для автомата Мілі) і $\mathbf{n =]\log_2 L[}$ [(для

автомата Мура). Як правило, число станів автомата Мура більше числа станів автомата Мілі.

8. При побудові комбінаційних частин автомата для формування вихідних сигналів і функцій збудження входів тригерів пам'яті використовують пряму структурну таблицю МПА, яка містить в своїх стовпцях;

- послідовність попередніх станів $\mathbf{z_i}$, і їх код $\mathbf{K[z_i];}$
- послідовність подальших станів $\mathbf{z_j}$, і їх код $\mathbf{K[z_j];}$
- набори вхідних сигналів $\{\mathbf{x_i}\}$, що викликають перехід $\mathbf{(z_i,\ z_j)}$, і вихідних сигналів y_α, y_β, y_w . Які формуються на даному переході;
- набори інформаційних входів тригерів, що повинні перемикатися на переході $\mathbf{(z_i,z_j)}$. Якщо використовують *RS*- і *JK*-тригери, то вони мають по два інформаційних входи, а якщо *D*- і Т-тригери, то вони мають по одному входу. Перемикання тригерів на переходах здійснюється згідно з даними табл.6.

1.3. Синтез мікропрограмного автомата з програмованою логікою

Мікропрограмний автомат з програмованою логікою будують на основі операційно-адресної структури з використанням загальних принципів програмного керування. При цьому алгоритм керування представляють упорядкованими наборами слів-мікрокоманд. Вони визначають порядок функціонування дискретного пристрою впродовж машинного циклу. Сукупність з $P_к$-розрядних мікрокоманд утворює їх масив, який зберігається в пам'яті мікрокоманд ПМК [***P:1***] автомата.

Структура МПА з програмованою логікою містить (рис. 6)**:**

формувач адреси мікрокоманди ***(ФАМК);***

регістр адреси мікрокоманди ***(RGAMK);***

дешифратор адреси мікрокоманд ***(DCA);***

пам'ять ***к*** -розрядних мікрокоманд ***ПМК*** [***P:1***] (***к***);

регістр мікрокоманд ***RGMK***, поділений на поля мікрооперацій ***Y*** (операційна частина), логічних умов ***X*** і адреси ***A;***

дешифратор мікрооперацій ***DCMO***, на виході якого формуються керуючі сигнали для ***ОА;*** у деяких типах ***МПА*** він відсутній.

У загальному випадку мікрокоманда містить інформацію про мікрооперації, які повинні виконуватися в даному такті, адреса наступної мікрокоманди і аналізуючі логічні умови.

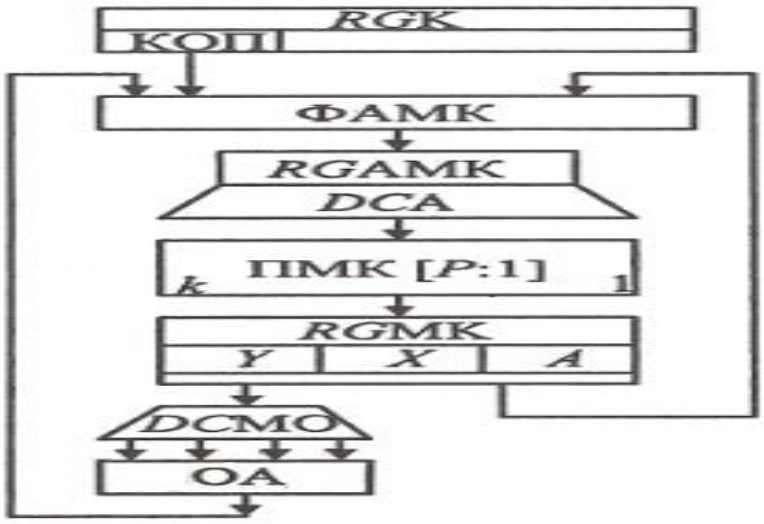


Рис. 6. Структура МПА з програмованою логікою.

Виконання кожної мікропрограми зводиться до послідовності наступних дій:

- зчитування мікрокоманд з **ПМК** за адресою, що формується вузлом **ФАМК**. Адресу першої мікрокоманди задають кодом операції **КОП** в регістрі команд ***RGK*** машини;
- запис зчитаної мікрокоманди в регістр ***RGMK***, дешифрація складового полів ***Y*** і ***X*** і видача набору керуючих сигналів в **ОА**;
- формування адреси наступної мікрокоманди з урахуванням полів ***X*** і ***Y***;
- видача сигналу "**Кінець**" мікропрограми;

Мікропрограмні автомати з програмованою логікою класифікують по наступних ознаках:

- типу пам'яті мікрокоманд – статичні (динамічні) і постійні (масочні і т. ін.);
- способу кодування мікрооперацій – горизонтальні, вертикальні, горизонтально-вертикальні, вертикально-горизонтальні;
- часу виконання мікрокоманд – синхронні і асинхронні і наявність фаз в такті – одно і багатофазні;
- способу адресації – природні, довільні (примусові) і способу організації умовних і безумовних адресних переходів;
- направленню вдосконалення структури і алгоритмів функціонування.

В якості ПМК використовують різні типи постійної пам'яті, а також ОП, завантажувану мікропрогмами з гнучких магнітних дисків при кожному включенню машини.

Розрізняють одно і багатофазні мікрокоманди. У першому випадку всі мікрооперації, указані в мікрокоманді, виконуються одночасно за один такт. У іншому – такт розбивається на інтервали, які мають назву фази або мікротакту; при цьому указані в мікрокоманді мікрооперацій виконуються в різних фазах.

При горизонтальному кодуванні поле ***Y*** містить ***M*** розрядів, де ***M*** – загальне число мікрооперацій (рис. 7,а).

Якщо в розряді стоїть одиниця, то відповідна мікрооперація виконується незалежно від значень інших розрядів.

Перевагою горизонтального кодування являється можливість одночасного виконання в одному такті мікрокоманди з любим набором з ***M*** мікрооперації і простоти формування керуючих сигналів – вони представляються на виходах розрядів поля ***Y***. Недоліком горизонтального кодування являється велика довжина микрокоманди, оскільки число ***M*** може досягати більше 100 значень.

При вертикальному кодуванні всі множини з ***M*** мікрооперацій кодуються n-розрядним двійковим позиційним кодом (рис. 7,б). При даному кодуванні в кожному такті виконується мікрокоманда з однією мікрооперацією. Перевагами вертикального кодування є коротка мікрокоманда, а недоліком необхідність використання складних дешифраторів і зменшення швидкодії автомата.

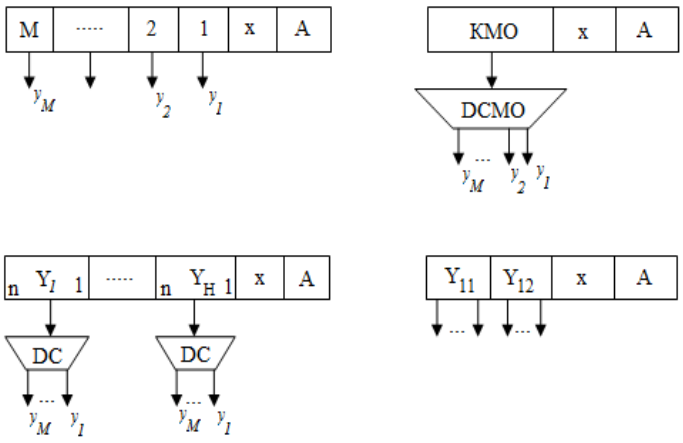


Рис. 7. Схеми кодування операцій мікропрограмних автоматів з програмованою логікою

При горизонтально-вертикальному кодуванні операційна частина ***Y*** мікрокоманди розбивається на ***N*** полів ***Y₁ Y₂, Yₙ*** (Рис.7, в). У кожному полі горизонтально розміщуються множини з ***M*** мікрооперацій ***y₁ ,y₂, ...yм*** , яким присвоєні номери ***1,2...,*** ***M***. Ці номери кодуються вертикальним способом – ***n***-розрядним двійковим кодом. Якщо поле ***Yᵢ*** порожнє (містить тільки нулі), то воно не ініціює ніякої мікрооперацій. Таким чином, залежно від коду в полях ***Yᵢ*** мікрокоманда може одночасно ініціювати від нуля до ***N*** мікрооперацій. Кожне поле ***Yᵢ*** дешифрується дешифратором на ***n*** входів і ***M*** виходів.

При вертикально-горизонтальному кодуванні (рис. 7,*г)* вся множина з ***M*** мікрооперацій розподіляється на ***к*** підмножин. У кожному з них об’єднуються мікрооперації, котрі частіше всього виконуються разом в одному такті.

Підмножини по можливості створюють рівнопотужні – з однаковим числом мікрооперацій. Операційна частина ***Y*** мікрокоманди складається з двох полів. У першому полі ***Y₁₁*** використовується горизонтальний спосіб кодування,

а друге поле Y_{12} показує, до якої підмножини належить мікрооперація в першому полі.

Спосіб адресації задає правила визначення адреси наступної мікрокоманди.

Існують наступні способи адресації:

- природна з послідовним розміщенням мікрокоманд;
- з довільним порядком проходження мікрокоманд;
- відносна – довільна в діапазоні адрес, визначених базовою адресою;
- по фіксаторам – адреса наступної мікрокоманди визначається складовим спеціальних комірок ПМК.

У кожному із способів адресації можна використовувати горизонтальний, вертикальний і змішаний спосіб кодування.

Природна адресація мікрокоманд реалізується таким чином.

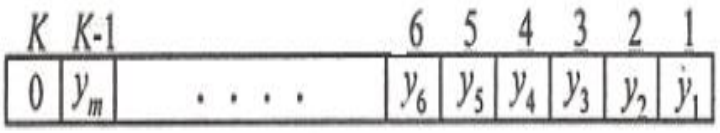
1. Використовують два формати мікрокоманд (рис. 8): ***операційний*** і ***адресний***. Формати відрізняються наявністю нуля в старшому ***к***-му розряді операційної мікрокоманди і одиниці – в адресній.

2. У операційній мікрокоманді записуються мікрооперації $\{y_i\}$, котре повинне буде виконане в даному такті. Природна адресація забезпечується лічильником адреси мікрокоманд ***СТАМК***, вміст якого автоматично збільшується на одиницю після виконання поточної мікрокоманди. У адресній мікрокоманді записують три параметри: однорозрядні значення логічної умови x_i (береться з графа автомата) і безумовного переходу БП, а також адреса переходу A_m , A_{m-1}, … , A_1. Адресу наступної мікрокоманди задають виразом:

R = МК[K]X_i $X`_i$ V **МК[*K*] БП,** де МК[K] – значення старшого розряду мікрокоманди;

x'_i - логічна умова з виходу ОА.

Таким чином, якщо $R = 0$, то відбувається природна адресація; при $R = 1$ адреса наступної мікрокоманди визначається її адресним полем A_m , A_{m-1}, … , A_1 . Описаний метод дозволяє використовувати ряд логічних умов, проте одиничне значення повинне сприймати тільки одне x_i .



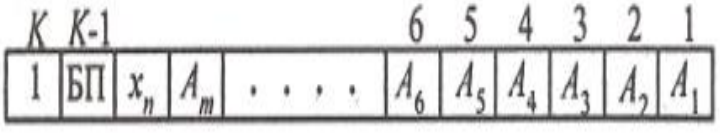


Рис. 8. Формати команд в мікропрограмних автоматах
з програмованою логікою.

Схема МПА, що використовує природну адресацію і горизонтальне кодування, показана на рис. 9.

У даній схемі при ***МК[К]*** =1 блокується видача керуючих сигналів y_r **..**y_1 і відбувається умовний перехід. При ***МК[К] =0*** видача керуючих сигналів дозволена. Інформація з пам'яті мікрокоманд зчитується по сигналу "СЧТ".

Схема МПА з природною адресацією і вертикальним кодуванням має вихідний дешифратор мікрооперацій. Поле мікрооперацій, як і поле адреси, кодується двійковим позиційним кодом, що значно скорочує довжину мікрокоманди (рис. 10). Адресація в схемі автомата на рис. 12 реалізована аналогічно схемі (рис. 11).

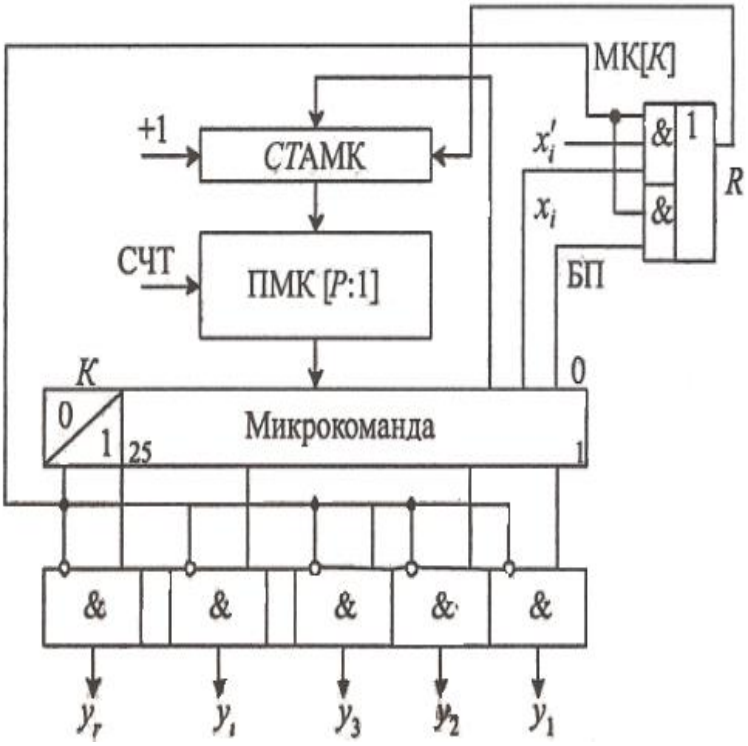


Рис. 9. Схема МПА з природною адресацією
і горизонтальним кодуванням

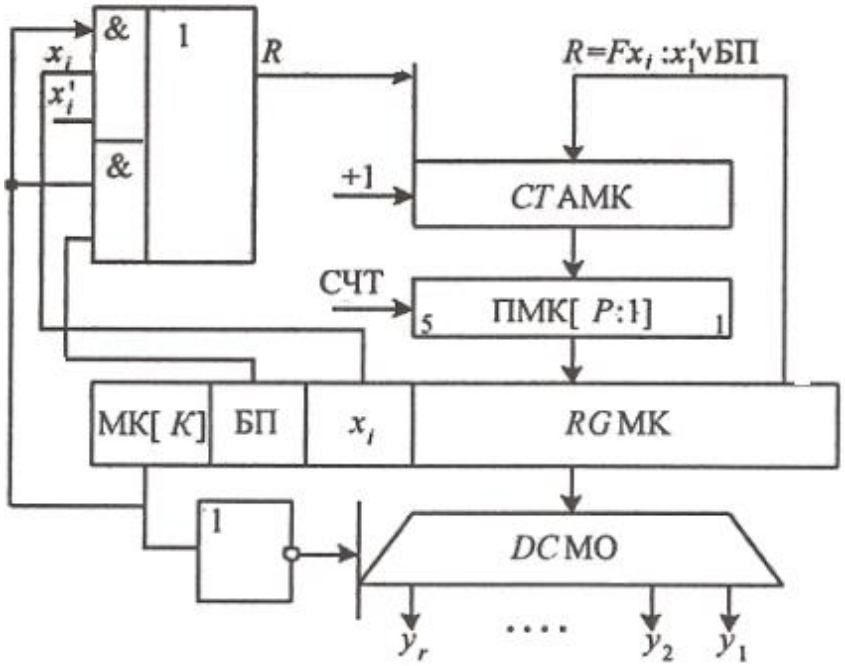


Рис. 10. Схема МПА з природною адресацією
і вертикальним кодуванням

Примусова адресація передбачає наявність в єдиному форматі мікрокоманди одного або двох адресних полів, в яких вказується адреса наступної мікрокоманди. Якщо є одне поле адреси, то для виконанння умовних і безумовних переходів використовують додаткову комбінаційну схему для інкремента на одиницю вмісту адресного поля.

Схема МПА з горизонтально-вертикальним кодуванням мікрооперацій і примусовою адресацією за допомогою двох адресних полів ***A0*** і ***A1*** показана на (рис. 11). При такій структурі мікрокоманди в кожному такті може виконуватися не більше двох мікрооперацій.

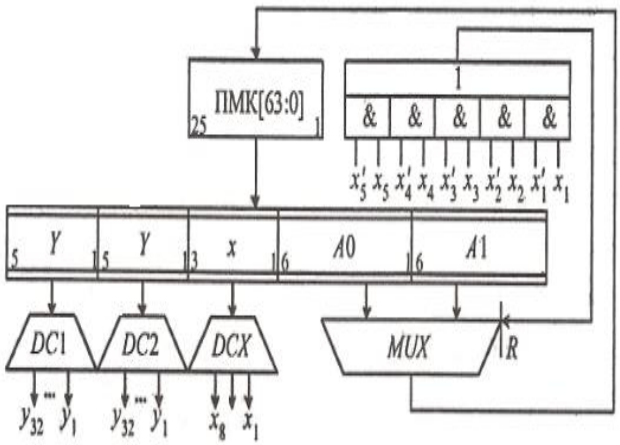


Рис. 11. Схема МПА з горизонтально-вертикальним кодуванням
мікрооперацій і примусовою адресацією за допомогою двох адресних полів

Мікропрограмні автомати з програмованою логікою характеризуються часом, витраченим на формування однієї мікрокоманди.

Цей час складається з трьох компонентів:

- часу формування адреси наступної мікрокоманди;
- часу звернення до ПМК;
- часу дешифрування операційної частини мікрокоманди.

Основна частка часу витрачається на зчитування мікрокоманди з ПМК, тому підвищена швидкодія може досягатися або за рахунок зменшення часу звернення до пам'яті, або за рахунок зменшення числа таких звернень.

Збільшення швидкодії МПА з програмованою логікою досягається при наступних умовах:

- застосування швидкодіючих ПМК;
- паралельна вибірка кількох мікрокоманд;
- випередженням вибірки мікрокоманд (до закінчення виконання попередньої).

Використання МПА з програмованою логікою порівняно з схемною має наступні переваги:

- забезпечується велика гнучкість і наочність, підвищується регулярність структури МПА і процесора в цілому;
- підвищується єфективність математичного забезпечення і продуктивність работи процесора;
- полегшується модернізація мікропрограм МПА як в процесі автоматизованого проектування, так і при роботі комп'ютера, що збільшує термін морального старіння машини;
- полегшується побудова ефективної мікродіагностики.

Мікропрограмний автомат з програмованою логікою має найширше використання в процесорах з малою і середньою продуктивністю. Обидва способи побудови автоматів керування отримали подальший розвиток на основі програмованих БИС, однорідних середовищ.

1.4. Центральний пристрій керування

Пристрій, виконуючий основні функції керуванням комп'ютером, називається центральним пристроєм керування.

Під ЦПК мають на увазі сукупність вузлів і блоків процесора, котрі забезпечують координацію функціонування всіх пристроїв машини і управління ними для всіх прийнятих режимів роботи.

Центральний пристрій керування реалізує системні і робочі програми, організовує, всі необхідні дії з оцінки і перетворення початкової інформації для отримання результату обчислень. Рішення будь-якої задачі зводиться до послідовності вибірки і виконання команд программи під керуванням ЦПК. Таким чином, ЦПК – це перетворювач первинної командної інформації, представленої командами программи, у вторинну командну інформацію виконуючі адреси і керуючі сигнали. До первинної командної інформації

відносяться також коди і сигнали,які характеризують стан процесора і окремих блоків.

Часто в міні і мікрокомп'ютерах , головною вимогою котрих являється мінімум вартості, пристрої суміщають по функціональному призначенню, наприклад, ЦПК і МПА процесора.

У ПК умовно виділяють дві основні частини – **мікропрограмну** і **програмну.**

Мікропрограмна частина – це МПА, виробляючий сигнал керування мікрооперaціями в АЛУ. Програмна частина визначає послідовність виконання програм команди, їх дешифрування, виробляє виконуючі адреси по яким зчитуються з пам'яті операнди в АЛУ і записується результат операції. Функції програмної частини реалізує ЦПК.

Для виконання своїх функцій ЦПК містить (рис.12**):**

- регістр команд ***RGK*** з полем коду операції **КОП** і полем адреси **АДР**;
- лічильник адреси команд **СТАК**;
- керуючий автомат **МПА** ;
- дешифратор коду операції ***DCKOП;***
- операціний блок (**D**), до складу якого входять суматор адреси, схеми аналізу режимів работи: готовності пам'яті і периферії до обміну інформації, запитів переривань і прямого доступу до пам'яті; інтерфейсні схеми;
- пульт керування "Пульт".

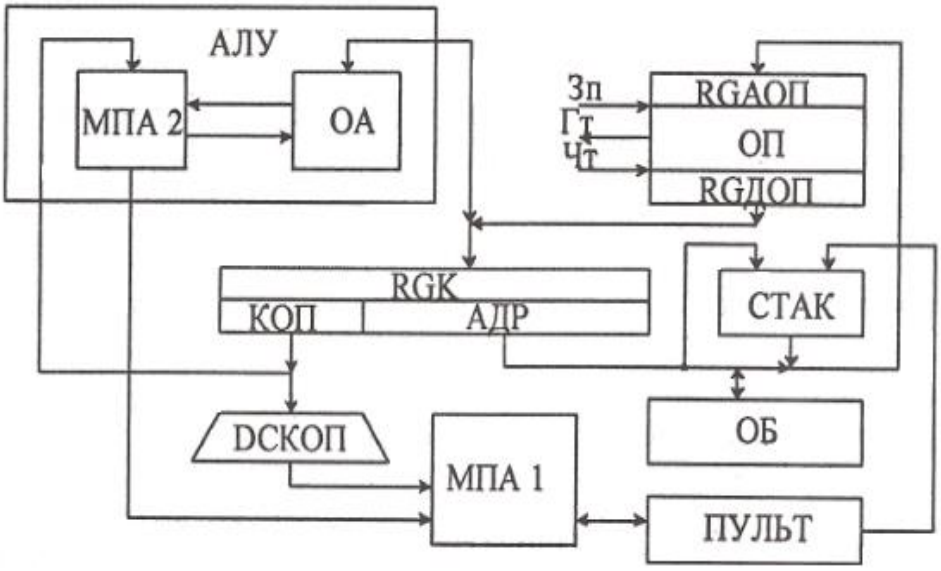


Рис. 12. Центральний пристрій керування.

Регістр команд ***RGK*** призначений для прийому команди з ОП і зберігання її протягом робочого циклу. Залежно від типу машини і складнощі операцій команда може мати довжину від одного до 10 і більш байт. У залежності від довжини шини вибірки даних за одне звернення до ОП може зчитуватьcя вся команда, її частина або декілька команд.

Лічильник адреси команд **СТАК** призначений для визначення адреси команди. Після зчитування поточної команди вміст **СТАК** автоматично збільшується на константу, рівну довжині команди в байтах.

Мікропрограмний автомат **МПА 1** розшифровує команди і забезпечує керуючими сигналами виконання програмної частини, а **МПА 2** – виконує операції в АЛУ. Кожний з автоматів може будуватись на основі схемної або програмованої логіки. При централізованому керуванні обидва автомати об'єднуються в єдиний **МПА.**

Суматор адреси служить для формування виконуючих адрес операндів і результату операції за інформацією, що міститься в коді команди. У загальному випадку виконуючі адреси отримують складанням трьох компонентів: базового адресу і индексу, розташованих в блоці РОН, і коди зсув у команді.

Пульт керування "Пульт" призначений для керування роботою комп'ютера користувачем. Він містить клавіатуру, переключаючі кнопки і засоби індикації для візуального контролю стану окремих пристроїв і проведення профілактики. На клавіатурі "Пульта" набирается команда ввода і адреса першої команди програми, яка повинна виконуватися після натиснення на кнопку **ЗАПУСК** (будь-якої клавіші). Ця інформація поступає в ЦПК, який керує введенням з магнітних дисків програми і початкових данних в ОП. Після закінчення введення програми ЦПК пересилає в лічильник **СТАК** адресу, першої призначеної для виконання команди.

Функціонування комп'ютера складається з робочих циклів, кожний з яких відповідає виконанню однієї команди програми. **У кожному робочому циклі в загальному випадку виконуються наступні типові дії:**

- вибірка з комірки ОП команди, яка повинна виконуваться, і формування адреси наступної команди. При цьому вміст лічильника адреси ***СТАК*** пересилається в регістр адреси пам'яті ***RGAOП.*** Зчитаний по даній адресі код поступає в регістр даних пам'яті, а звідти пересилається в регістр команд ***RGK,*** після чого вміст лічильника адреси ***СТАК*** збільшується на константу – довжину команди в байтах;
- формування виконуючих адрес і зчитування по них операндів з ОП. Вміст адресної частини команди пересилається в ОБ, де виробляються виконавчі адреси. Зчитані операнди поступають в **ОА** і зберігаються в блоці **РОН**;
- розшифровка коду операції в **МПА2**: виконується послідовність мікроопераций, які визначені мікропрограмою даної операції і запис результату операції в пам'ять або в РОН;
- виробка в **МПА2** сигналу кінця операції. Перехід до п. 1.

При кожному зверненні до пам'яті перевіряється сигнал готовності "Гт" пам'яті дообміну інформацією. Обмін (зчитуваня або запис) можливий при "Гт" = 1, інакше процесор переходить в режим очікування. Після кожного звернення до ОП перевіряється наявність сигналу запиту прямого доступу до пам'яті. Якщо такий сигнал є, то процесор перемикає свої шини в третій стан і забезпечує периферійним пристроям режим прямого доступу до пам'яті. Після

закінчення кожної команди перевіряється наявність сигналу запиту на переривання програми від периферії.

При запиті на керування виконується процедура реалізації підпрограми обслуговування того зовнішнього пристрою, який встановив запит на переривання.

Після обслуговування зовнішнього пристрою комп'ютер повертається до виконання первинної програми.

Частина 2. Структура та принципи організації складових сучасних персональних комп’ютерів

2.1. Арифметико-логічні пристрої і пристрої управління ПК

Усі обчислювальні системи мають однакову базову структуру, яка включає один чи декілька центральних процесорів і сопроцесорів, що здійснюють обробку інформації, пристрої збереження інформації, а також пристрої введення та виведення інформації для спілкування із зовнішнім середовищем та іншими обчислювальними системами.

У ранніх поколінь комп'ютерів процесори і зовнішні пристрої (пам'ять і пристрої введення-виведення) мали приблизно однакову і дуже невисоку тактову частоту (біля 10 Мгц). Тому взаємодія процесора і зовнішніх пристроїв здійснювалася через єдиний інтерфейс – системну шину, яка працювала з тактовою частотою процесора. Швидкодія процесорів збільшувалася швидше за зовнішні пристрої. Тому тактова частота системної шини зростала, а пам'ять і пристрої введення-виведення не могли працювати з такою частотою. Були розроблені спеціальні адаптери, які підключалися до системної шини і на знижених частотах взаємодіяли з зовнішніми пристроями. Так виникли шини пам'яті і шини розширення (ISA та ін.). При цьому, для збільшення швидкодії пам'яті була застосована концепція кеш пам'яті. Це пам'ять невеликого об'єму і високої швидкодії, яка працювала на високій частоті системної шини.

Із зростанням швидкодії окремих зовнішніх пристроїв довелося розробити локальні швидкодіючі шини (наприклад PCI, SCSI). Але надалі, всі зовнішні пристрої стали достатньо швидкодіючими і потреба в повільних шинах розширення відпала, їх місце зайняли локальні шини. Для обслуговування високошвидкісних пристроїв були розроблені спеціальні мікросхеми, які потім об'єдналися у великі інтегральні мікросхеми (ЧІП). Наприклад, «північний міст» і інші аналогічні.

Але все ще залишалися пристрої з принципово низькою швидкодією. Для їх обслуговування також створюються спеціальні мікросхеми, наприклад «південний міст». Для обслуговування дуже низькошвидкісних пристроїв, таких як модем, мишка, клавіатура, використовуються спеціальні мікросхеми (наприклад типу Super I/O).

2.2. Базова функціональна схема комп'ютера

Зараз паралельні шинні інтерфейси все частіше замінюються послідовними, а для різних по швидкодії груп пристроїв використовуються свої паралельні або послідовні інтерфейси.

На рис. 13 представлена спрощена функціональна базова схема, яка дозволяє одержати загальне уявлення про функції, пристрій і принцип дії комп'ютера. Основний елемент комп'ютера центральний процесор (ЦП) працює разом з математичним співпроцесором (сучасні процесори об'єднані із співпроцесорами в одному корпусі) і синхронізується тактовим генератором (Г). Із зовнішнім світом процесор взаємодіє через системну шину, яка через набір мікросхем системної логіки перетворюється у більш повільні інтерфейси. Загалом вони утворюють систему інтерфейсів, через яку з'єднується процесор і зовнішні пристрої.

Системна шина, до якої підключається процесор, ще називається процесорною шиною або передньою шиною (Front Side Bus – FSB) або внутрішньою шиною (Processor Side Bus - PSB). Існують ще шини розширення (ISA), локальні шини (PCI, SCSI та ін.). Блоки управління інтерфейсами (контролери) знаходяться в спеціальних мікросхемах – Чіпах системної логіки. Процесор видає адреси і сигнали управління і обмінюється інформацією з пам'яттю, пристроями введення-виведення і зовнішніми пристроями.

Рис. 13. Узагальнена базова функциональна схема комп'ютера

Через шинний інтерфейс підключена основна пам'ять. Вона складається з швидкодіючої пам'яті для короткочасного оперативного зберігання даних ОЗП (оперативний запам'ятовуючий пристрій або RAM – random access memory) і енергонезалежної пам'яті (ЭНЗУ), зокрема постійної пам'яті (ПЗП – постійний запам'ятовуючий пристрій, або ROM – read only memory). Для довготривалого

зберігання даних використовуються зовнішні накопичувачі. Наприклад, НЖМД (накопичувач на жорстких магнітних дисках) має низьку швидкодію, але великий об'єм і зберігає дані при виключенні живлення. НГМД – накопичувач на гнучких магнітних дисках, місткість його невелика, але дискети зручні для перенесення інформації.

Безпосередньо до процесора підключена кеш-пам'ять, завдяки цьому реалізується її висока швидкодія.

Зовнішні пристрої підключаються до інтерфейсів через спеціальні апаратні засоби адаптери і контролери (по термінології IBM це синоніми). Введення-виведення даних на зовнішні пристрої проводиться через відповідні контролери і порти. У операціях введення-виведення задіяний контролер переривань, а також контролер прямого доступу до пам'яті (ПДП – DMA).

При перериванні процесор тимчасово припиняє свою роботу і обслуговує пристрій введення-виведення, а при прямому доступі до пам'яті процесор відключається від шини і не заважає прямому високошвидкісному потоку даних між зовнішніми пристроями і пам'яттю.

Електроживлення всіх пристроїв здійснюється через блок живлення (БЖ).

2.3. Процесор, типова структура процесора

Процесор (центральний процесор – ЦП, мікропроцесор – МП, CPU – Central Processing Unit) – програмно керований пристрій обробки інформації. ЦП призначений для управління роботою комп'ютера і виконання арифметичних і логічних операцій.

Процесор це одна або декілька великих інтегральних схем (ЧІП), що містять мільйони транзисторів.

Існує велика кількість типів процесорів, які відрізняються своїми параметрами. На рис. 14 представлена логічна організація процесора с одним акумулятором. Двійкові числа, з якими виконуються дії, через інтерфейс поступають з системної шини і зберігаються в одному з буферних регістрів, акумуляторі або кеш-пам’яті. А арифметико-логічному пристрої (АЛП) виконуються арифметичні або логічні дії з цими числами, а результат зберігається у регістрах. Керує всіма діями пристрій керування.

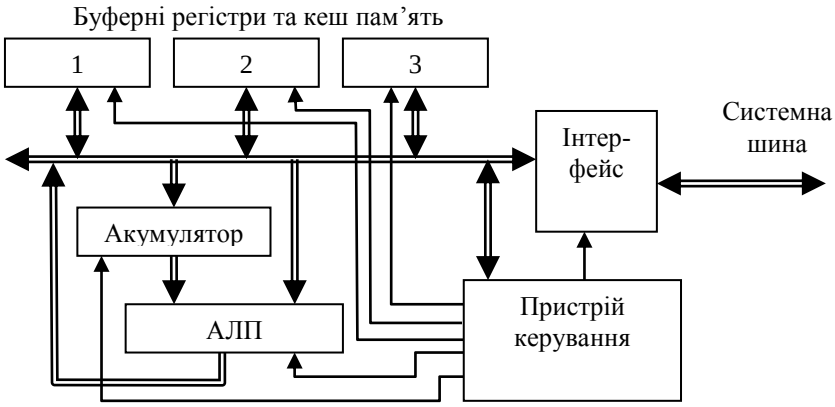


Рис. 14. Логічна організація процесора з одним акумулятором

На рис. 15 представлена спрощена типова структурна схема одного з можливих варіантів процесорів.

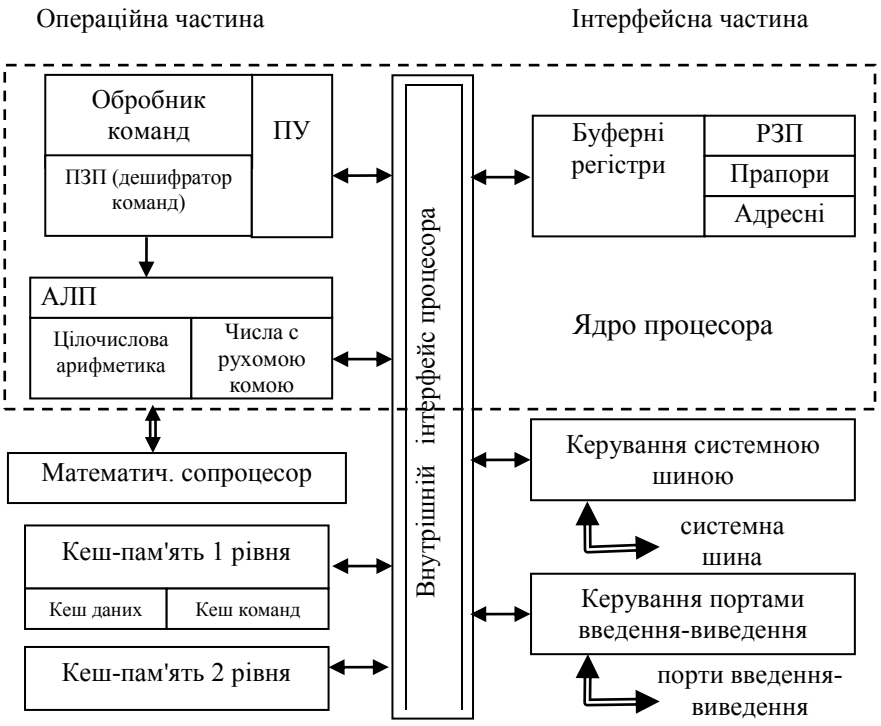


Рис. 15. Спрощена типова структурна схема процесора

Пристрій управління (ПУ) містить: дешифратор операцій, блок випереджаючого їх виконання і прогнозу галужень, ПЗП мікропрограм, вузол формування адреси. ПЗП мікрокоманд призначений для зберігання і дешифрування машинних команд і формування алгоритмів їх виконання.

Арифметико-логічний пристрій (АЛП або операційний блок) здійснює арифметичні і логічні операції, він містить: регістри, суматор, схему управління. У різних процесорах АЛП виконує різні функції, але основними є наступні:

- Складання з перенесенням і віднімання із заємом.
- Зсув вліво і управо.

– Логічне множення і складання.
– Порівняння цифрових кодів.

Деякі процесори мають цільове призначення, тому їх операції теж специфічні.

Регістри і кеш-пам'ять служать для тимчасового зберігання даних з дуже малим часом доступу. Всі пристрої взаємодіють між собою через внутрішній шинний інтерфейс, а із зовнішніми пристроями через блоки управління системною шиною і операціями введення-виведення.

Основні властивості процесора визначаються його системою команд, наявністю співпроцесора, тактовою частотою, числом буферних регістрів мікропроцесорної пам'яті і об'ємом кеш-пам'яті.

Процесори класифікуються по типу і розрядності системної шини і розрядності буферних регістрів. Сучасні процесори Pentium мають розрядність шини 64, а регістрів в основному – 32 (тільки найновіші 64). Таким чином, майже всі процесори працюють з 32 розрядними даними і командами при 64 розрядній шині даних. Важливими параметрами є об'єм кеш-пам'яті і об'єм оперативної пам'яті, що адресується процесором, цей об'єм досягає 64 Гбайт.

Для розширення функціональних можливостей процесора він доповнюється математичним співпроцесором. Основні функції співпроцесора: прискорення операцій над числами з фіксованою і рухомою комою, обчислення масивів і функцій, обробка мультимедійної інформації.

2.4. Мікропроцесорна пам'ять

Мікропроцесорна пам'ять має у своєму складі: регістри загального призначення (РЗП) або універсальні регістри, сегментні регістри, регістри зсуву, регістр прапорів. Вони призначені для тимчасового зберігання і швидкого доступу до адрес і проміжних даних при виконанні операцій процесором. Регістри поділяють на доступні програмісту і внутрішні. Сукупність регістрів ЦП, які доступні програмісту, називають реєстровим файлом або набором регістрів.

У базовій моделі процесора розглядається 14 регістрів пам'яті (рис. 16). Реально в різних процесорах буває до 256 регістрів розрядністю до 8 байт (Pentium).

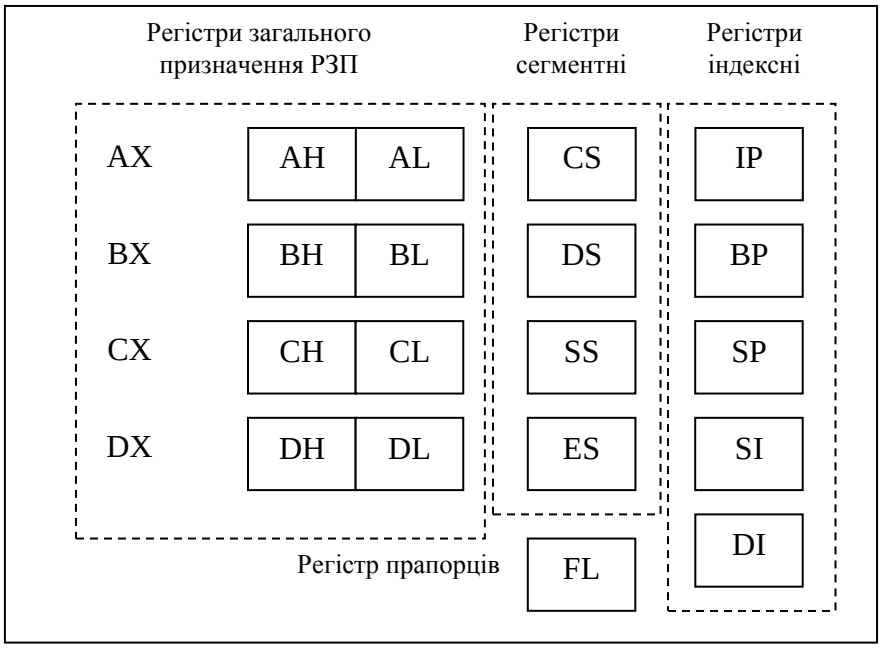


Рис. 16. Внутрішні регістри базової структури процесора

Регістри загального призначення (РЗП).

Однорозрядні позначаються однією буквою A, B, C, D.

16-ти розрядні (2-байтні) подвійними буквами AX, BX, CX, DX.

8-ми розрядні половинки 16-ти розрядних регістрів – додається буква L, H. Наприклад, AL, AH.

4-байтні – додається буква E: наприклад, EAX, EBX.

Спеціалізовані регістри:

A – акумулятор – накопичує результати обчислень. Це основний робочий регістр, в ньому зберігаються операнди майже всіх операцій і через нього проводиться основне звернення до пам'яті;

B – зберігання бази сегментної частини адреси;

C – лічильник повторень операцій;

D – регістр даних, використовується самостійно або спільно з A.

Регістр прапорців (станів) – FL

Служить для індикації стану процесора. Стан відбивається в окремих бітах, які можуть бути умовами для виконання різних дій. Биті умов називаються ще логічними змінними або прапорами. У різних процесорах можуть бути різні прапори, але більшість має наступні прапори.

Статусні прапорці:

CF (Carry) – прапорець перенесення із старшого розряду при арифметичних операціях і зсуві;

PF (Parity) – парність, результат має парне число одиниць;

AF (Auxiliary Cary) – перенесення в двійково-десяткових операціях;

ZF (Zero) – результат операції рівний нулю;

SF (Sing) – знак отриманого результату;

OF (Overflow) – переповнювання, результат операції дуже великий (наприклад, при діленні на нуль).

Прапорці, що управляють:

TF (Trap) – трасування, покрокове виконання програми;

IF (Interrupt) – дозволяються переривання;

DF (Direction) – напрям обробки строкових масивів шляхом зміни вмісту регістрів SI і DI на +1 або -1.

Адресні регістри

IP – вказівник (лічильник) команд (IP – instruction pointer). Містить зсув адреси команди, яка виконується. Спільно з регістром сегменту і базовим регістром указує фізичний елемент пам'яті, що адресується. Після виконання чергової команди стан лічильника автоматично збільшується і він вказує на наступний елемент пам'яті, де починається наступна команда.

BP – (base pointer) регістр базової частини адреси.

SP (stack pointer) – покажчик стека, зсув вершини стека.

SI (source index), DI (destination index) – регістри індексу джерела і призначення.

CS, DS, ES, SS – регістри сегментів коду, даних, додатковий і стека.

2.5. Кеш-пам'ять процесора

Швидкодія процесора на багато більше, ніж основній оперативній пам'яті ОЗП. А оскільки звернення до пам'яті відбувається постійно, то вона обмежує реальну швидкість процесора. Проблема вирішується використанням швидкодіючої кеш-пам'яті невеликого об'єму, в якій дублюється вміст тієї частини ОЗУ, до якої чергове звернення процесора буде найбільш вірогідне. Після цього процесор звертається вже не до ОЗП, а до кеш-пам'яті, швидкодія якої у багато разів більша ніж у ОЗП. Але можливі випадки, коли прогнози не виправдовуються, тоді доводиться наново завантажувати інформацію з ОЗП у кеш-пам'ять. На це йде час, в перебігу якого процесор знаходиться в режимі очікування.

В результаті при збільшенні об'єму кеш-пам'яті реальна швидкість роботи процесора значно збільшується, але не нескінченно. Природно, що вона обмежена можливостями самого процесора. На рис.17 показана орієнтовна залежність швидкодії комп'ютера від місткості кеш-пам'яті.

Кеш-пам'ять має три рівні L1 – усередині процесора, L2 – поза процесором і працює з внутрішньою частотою процесора (у процесорах після 1999 року інтегрована в ядро), L3 – на системній платні. Перший рівень самий швидкодіючий, але обмежений за об'ємом розмірами кристала процесора.

Зовнішні пристрої пам'яті і пристрої введення-виведення також мають свою кеш пам'ять для прискорення їх роботи.

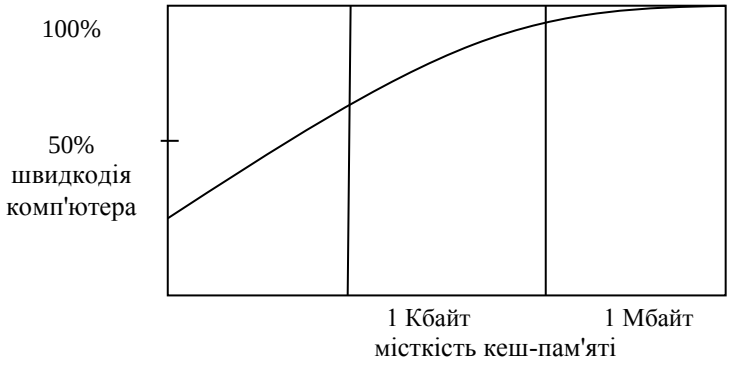


Рис. 17. Залежність швидкодії комп'ютера від місткості кеш-пам'яті

Сучасні процесори

Сучасні процесори мають тактову частоту до 3-4 ГГц, розрядність до 128 біт, об’єм кеш-пам’яті знаходиться у межах 8-256 кбайт, кількість регістрів до 256. В сучасних процесорах застосовується величезна кількість різних технологічних новацій, які підняли як їх продуктивність, так і складність на висоту, яка ще 10 років назад здавалася неймовірною.

До першої групи новацій слід віднести технології обробки чисел з плаваючою комою. Другу групу складають технології паралельного і конвеєрного виконання операцій. Так, наприклад, технологія EPIC та інші дозволяють виконувати до 20 операцій за один такт і до 8 операцій над числами з рухомою комою. Один із способів досягти цього полягає в збільшенні числа суматорів арифметичного пристрою, числа регістрів і т.д. Число регістрів вже досягло 128 для цілих чисел 128 для чисел з рухомою комою.

Операції виконуються не тільки паралельно, але навіть забігаючи вперед. Спеціальні блоки і алгоритми дозволяють передбачати команди і галуження програми, що прискорює в середньому роботу комп'ютера.

2.6. Пам'ять комп'ютера

Пам'ять другий після процесора компонент комп'ютера по важливості і вартості. У комп'ютері є багато різних видів пам'яті, їх наявність і параметри визначають основні властивості всього комп'ютера.

2.6.1. Типи пам'яті

Коротка класифікація основних властивостей усіх типів пам'яті зображена на рис. 18.

За способом запису і вибірки даних з пам'яті розрізняють запам'ятовуючі пристрої (ЗП) з довільною вибіркою (Random Access Memory – RAM) і пам'ять з послідовною вибіркою. Перше означає, що записувати і читати можна

довільні комірки пам’яті у будь-якій послідовності. Для цього досить указувати відповідні адреси комірок. При послідовній вибірці звернення до даних проводиться тільки строго у певному порядку, тому тут немає необхідності указувати адреси, що спрощує роботу з пам'яттю. Наприклад, послідовну вибірку мають пристрої пам'яті із стрічковими носіями. Іншим прикладом послідовної вибірки є стекова пам'ять.

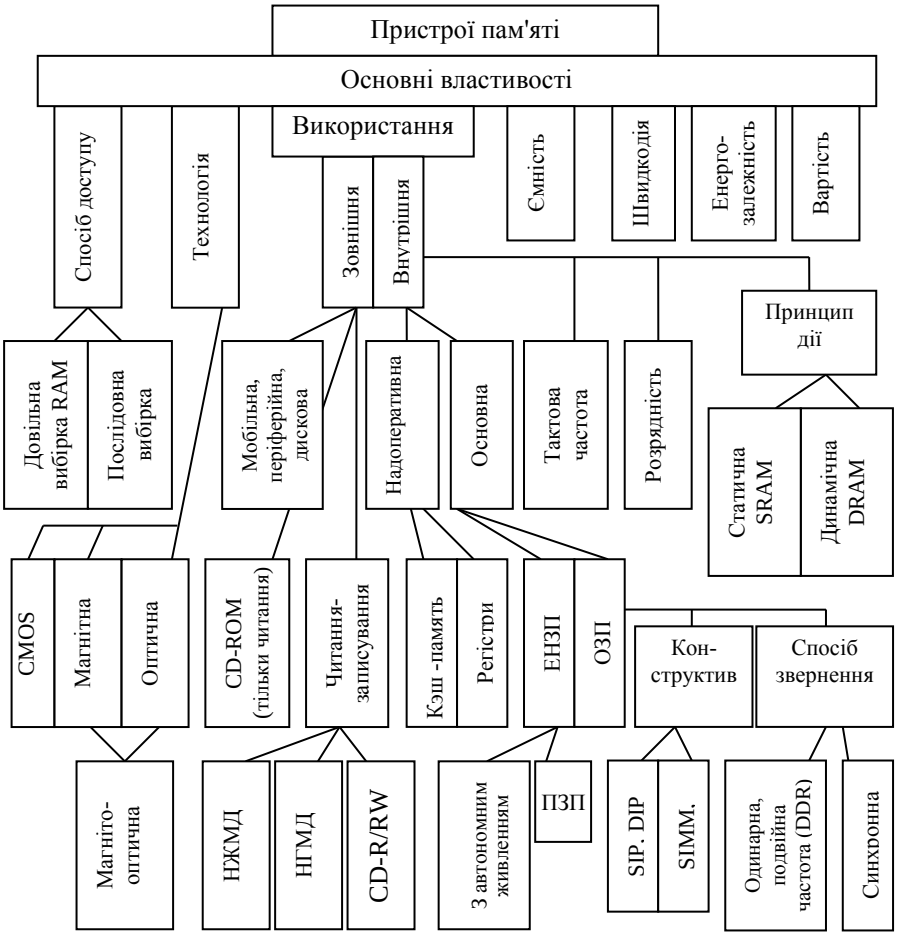


Рис. 18. Коротка класифікація основних властивостей пам’яті

По місцю використання у комп'ютері і функціям можна виділити наступні типи пам'яті:

– Мікропроцесорні регістри (МПР)

– Кеш-пам'ять (КЕШ)

– Основна пам'ять (ОЗП – оперативний ЗП, ЕНЗП – енергонезалежний ЗП, ПЗП – постійний ЗП)

– Зовнішня пам'ять (змінна, флеш, периферійна, дискова, стрічкова – ПМЛ, оптична, – CD)

Дискова пам'ять: НЖМД – накопичувач на жорстких магнітних дисках («вінчестер»), НГМД – накопичувач на гнучких магнітних дисках.

CD - ROM, CD - R, CD - RW – накопичувачі на лазерних компакт-дисках (ROM – тільки для читання, R – записуваний, RW – перезаписуваний).

CMOS – пам'ять на основі технології малоспоживаючих структур працює з автономним живленням.

Всі типи пам'яті мають різні параметри і характеристики, достоїнства і недоліки. Немає жодного типу пам'яті, що повністю задовольняє всім вимогам. Саме це і обумовлює їх різноманіття у комп'ютері. На рис. 19 представлено порівняння різних типів пам'яті по параметрам об'єм і швидкодія. Від параметрів залежить область застосування кожного типу пам'яті.

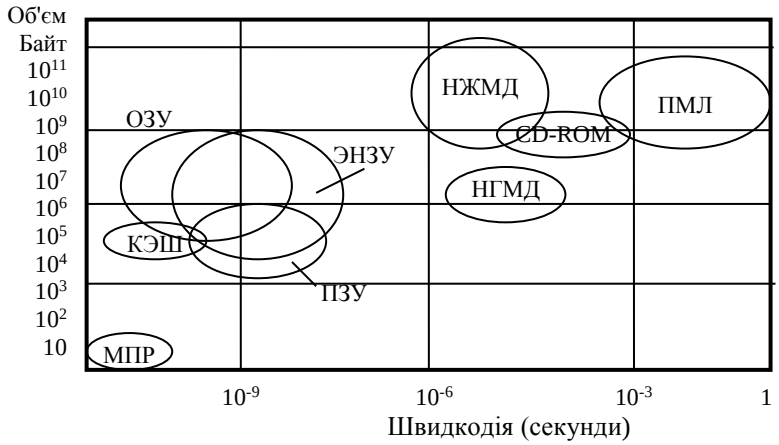


Рис. 19. Порівняння різних видів пам'яті по об'єму та швидкодії

Надоперативна пам'ять має дуже високу швидкодію і невисоку місткість. Місткість основної пам'яті значно вищий, але швидкодія гірше. Ці види пам'яті використовують для тимчасового зберігання інформації при включеному комп'ютері.

Зовнішня пам'ять має значні об'єми і дуже низьку швидкодію. Позитивна властивість цієї пам'яті в тому, що її вміст зберігається при виключенні живлення. Всі ці якості визначають її застосування для довготривалого зберігання інформації.

Пам'ять на магнітній стрічці застосовують іноді для резервного копіювання даних, де не потрібен швидкий пошук інформації.

З принципу дії пам'ять комп'ютера буває наступних типів:

1. Статична пам'ять (SRAM) побудована на статичних тригерах, має високу вартість і застосовується там, де потрібен висока швидкодія і невелика місткість (регістрова пам'ять і кеш-пам'ять з швидкодією 4 нсек). Використовується також в енергонезалежній пам'яті з автономним живленням.

2. Динамічна (DRAM) – інформація зберігається у вигляді електричного заряду і потрібне постійне її оновлення. Вона дешевша і може бути більшого об'єму чим статична. Використовується в пристроях оперативної пам'яті (ОЗП).

Порівняльна характеристика двох видів пам'яті:

Таблиця 7

Тип пам'яті	параметри				
	Швидкодія	Об'єм	Енерго-споживання	Вартість	Застосування
SRAM	висока	низький	високе	висока	МПР, КЕШ
DRAM	мала	високий	низьке	низька	ОЗП

Окрім цих параметрів при виборі враховуються експлуатаційні, завадостійкість і інші характеристики.

З рис. 19 видно, що динамічний ОЗП має високу швидкодію (близько 16-533 Мгц або 2-60 нсек), але об'єм істотно менше, ніж у НЖМД, окрім того НЖМД зберігає інформацію при виключенні живлення. Тому вони застосовуються завжди у парі. Спочатку інформація завантажується з НЖМД в ОЗП і лише після цього процесор з високою швидкістю її обробляє. Програми і дані, що знаходяться в НЖМД просто зберігаються, а завантажені в ОЗУ вважаються запущеними або активованими. При виключенні живлення інформація з ОЗП знову перевантажується в НЖМД. Точно також здійснюється перевантаження з ОЗП в кеш-пам'ять, швидкодію і вартість якої ще вище, а об'єм істотно нижче, ніж у ОЗП.

В процесі роботи комп'ютера відбувається постійний обмін даними між ОЗП і НЖМД: деякі дані додатково прочитуються з НЖМД, якісь тимчасово розташовуються в НЖМД щоб звільнити місце в ОЗП (файли підкачки).

Не дивлячись на високу швидкодію динамічного ОЗП, воно у декілька разів менше швидкодії процесора (1-3 ГГц). Тому для роботи з ОЗУ використовується спеціальний інтерфейс з тактовою частотою нижче, ніж у процесора, а при зверненні до ОЗП процесору доводиться якийсь час чекати відповіді. Такі простої знижують загальну швидкодію комп'ютера.

З тактовою частотою процесора може працювати тільки статична пам'ять, яка має час доступу менше 2 нсек, тому вона використовується в кеш-пам'яті. При зверненні до кеш-пам'яті процесор відразу ж одержує відповідь, якщо в ній міститься необхідна інформація. Якщо ж контролер помилився і неправильно передбачив область пам'яті, яка кешується, то процесору доведеться звертатися в основну пам'ять. Такі помилки знижують ефективність роботи кеш-пам'яті.

2.6.2. Основна пам'ять

Основна пам'ять комп'ютера це пам'ять з довільною вибіркою (RAM) призначена для зберігання даних і програм під час роботи комп'ютера. Як видно з рис. 18, основна пам'ять складається з оперативної (ОЗП) і

енергонезалежної (ЕНЗП) пам'яті. ОЗП це робоча область пам'яті процесора призначена для тимчасового запису і читання даних і програм тільки під час роботи комп'ютера, при виключенні живлення інформація в ній втрачається. На відміну від цього, ЕНЗП зберігає дані і програми також і при вимкненому живленні. Для цього пам'ять має власне автономне живлення, а деякі типи пам'яті (які виконані за спеціальною технологією) для зберігання інформації живлення не вимагають.

Постійна пам'ять (ПЗП / ROM - Read Only Memory – тільки для читання) незалежно від живлення постійно зберігає дані і програми, які не передбачається часто оновлювати, а іноді взагалі неможливо відновити. Існують різні типи ПЗП. Наприклад, масочні ПЗП (ROM) – інформація в них заноситься при виробництві і не змінюється (не перепрограмується) в процесі роботи. ППЗП (PROM) – пам'ять, одноразово перепрограмована на спеціальних пристроях. EPROM – багато разів перепрограмована пам'ять. Сюди відноситься електрично перепрограмована пам'ять EEPROM, використовувана в пристроях флеш-пам’яті.

Конструктивно мікросхеми пам'яті випускаються в різних типах корпусів:

SIP – корпус з однорядним розташуванням виводів;

DIP – корпус з дворядним розташуванням виводів.

Декілька мікросхем розташовані на платні утворюють модуль пам'яті:

SIMM – модулі (плата) пам'яті з однорядним розташуванням контактів роз'єму (застарілі);

DIMM – нові модулі з контактами в два ряди мають високу місткість і тактову частоту 100МГц і 133 МГц;

RIMM – новітні швидкодіючі модулі пам'яті.

Нові типи оперативної пам'яті відрізняються способом обміну даними:

FPM DRAM, RAM EDO – застарілі типи пам'яті;

BEDO DRAM – пам'ять, в якій обмін з процесором відбувається за один такт блоками постійної максимальної довжини;

SDRAM – новіший тип пам'яті, синхронізований з роботою процесора, має блоковий спосіб обміну і конвеєрну технологію, при якій поки встановлюється адреса в одній частині пам'яті (банку) відбувається вибірка з іншого банку;

DDR SDRAM – синхронна динамічна пам'ять з подвійної частотою передачі даних по двом фронтам тактового сигналу. Випускаються у вигляді модуля DIMM;

DRDRAM – перспективний тип пам'яті з двобайтовою шиною Rambus з частотою 800 МГц;

FERAM, MRAM – фероелектрична і магнітна пам'ять високої швидкодії.

2.6.3. Принципи організації основної пам'яті

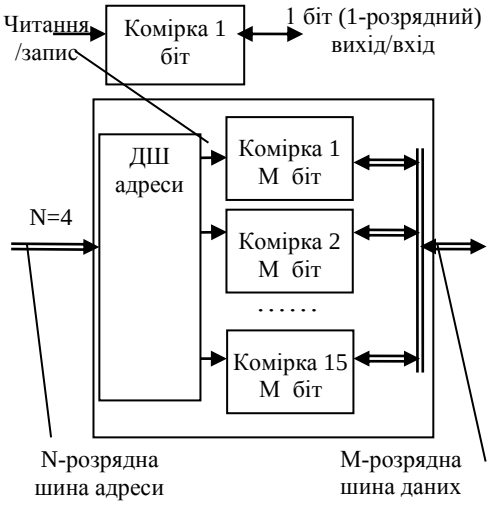
Основна пам'ять складається з комірок, кожна з яких зберігає інформацію 1 біт. Якби комірка була одна, до неї можна було б звертатися безпосередньо,

але коли їх багато їх вибирають за адресою, а звертаються через шину адреси і дешифратор адреси ДШ (рис. 19).

Для М-розрядних шин даних 1-розрядні комірки об'єднуються в М-розрядні із зверненням до них за однією адресою, а обмін даними здійснюється по М проводам шини даних. Обмін інформацією процесора з пам'яттю відбувається за час від одного до трьох тактів плюс час очікування.

ОЗП і ПЗП мають єдиний адресний простір, який визначає максимальну кількість комірок основної пам'яті, що безпосередньо адресуються. При розрядності адресної шини N, місткість адресного простору рівно 2^N.

На рис. 19 приведена таблиця адрес для 4-розрядної шини адреси. В цьому випадку, місткість адресного простору пам'яті рівна 16.



№ комірки	Код адреси			
	A1	A2	A3	A4
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Рис. 19. Принцип доступу до комірок

2.6.4. Сегментація пам'яті

Якщо процесор оперує N-розрядними адресами, а модуль пам'яті має Z комірок тобто його розрядність К=log Z, тоді можливі 3 варіанти:

1. Місткість ОЗП рівна місткості адресного простору тобто N = K. Підключення пам'яті проводиться по схемі рис. 20.

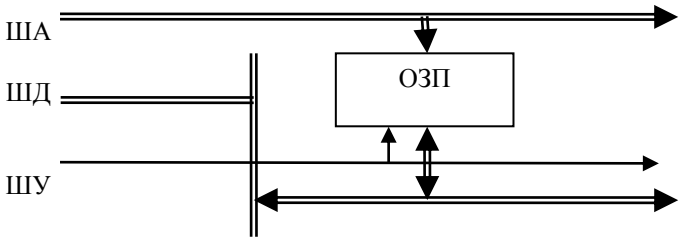


Рис. 20. Об'єм ОЗП дорівнює ємності адресного простору

2. Місткість одного модуля ОЗП менше місткості адресного простору (N > K). В цьому випадку пам'ять складається з декількох модулів або сегментів. Для вибору комірки усередині модуля або сегменту використовуються К розрядів (зсув), а N-K розрядів адреси, що залишилися, використовуються через дешифратор для вибору модуля або сегменту (рис. 21).

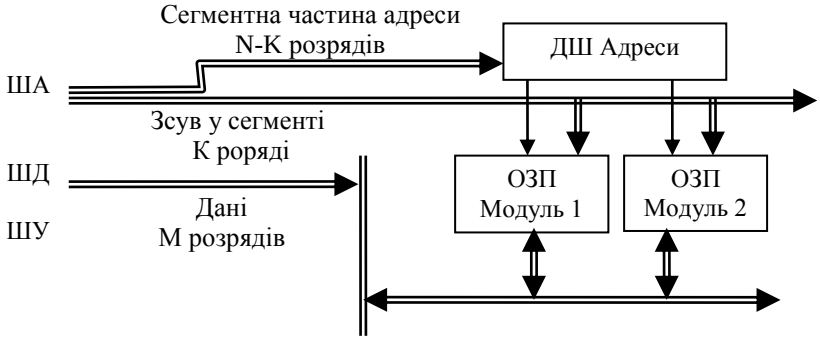


Рис. 21. Ємність одного модуля ОЗП менше ємності адресного простору

3. Місткість ОЗП більше місткості адресного простору. В цьому випадку пам'ять ділиться на сегменти по 2^N комірок. Усередині сегменту адресація проводиться по N-розрядній шині, ця частина адреси називається зсувом, а для адресації самих сегментів створюються додаткові розряди адресної шини (рис. 21).

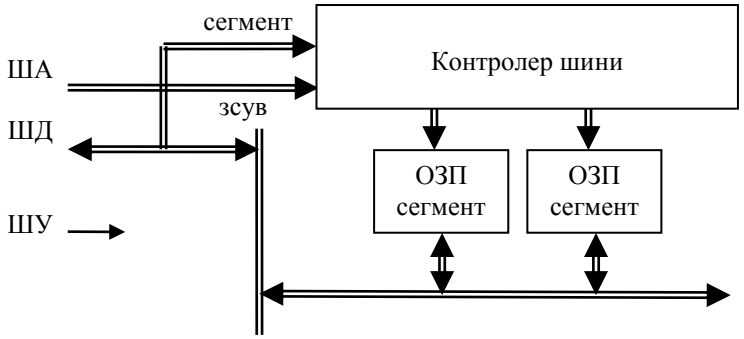


Рис. 22. Ємність ОЗП більша за ємність адресного простору

Наприклад, при N = 16 і необхідному об'ємі пам'яті 1 Мбайт (К = 20) в адресній шині не вистачає 4 розрядів. Реальна фізична 20-розрядна адреса реалізується за допомогою схеми рис. 23. Маємо $2^4 = 16$ сегментів з числом комірок у сегменті $2^{16} = 65536$ (64 Кбайт).

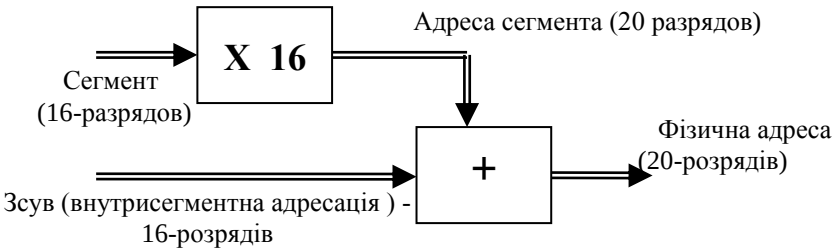


Рис. 23. Розрахунок 20-розрядної фізичної адреси

Реально частіше буває випадок – 3. Для вибору реальної комірки пам'яті треба вказати у якому сегменті вона знаходиться (адреса сегменту) і яке місце займає комірка у цьому сегменті (адреса зсуву). Кожна програма звично має свій сегмент пам'яті і не має доступу в інший. Іноді для зручності програмування і оптимізації програм адресу даних обчислюють з урахуванням інших даних, наприклад, адреси базової і індексної.

2.6.5. Структура основної пам'яті

Загальний об'єм пам'яті може нарощуватися, при цьому її початкова частина розбивається стандартним чином на області, які мають своє призначення і користувачів.

Таблиця 8

Область пам'яті (адреса)	Тип	Назва	Розмір	Призначення
- 512 Мбайт і більш				Програми і дані ОС і користувача
- 64 Мбайт	Розширена пам'ять			
- 1088 Кбайт		Висока пам'ять		
- 1024 Кбайт (1Мбайт)	Пам'ять, що безпосередньо адресується (1 Мбайт)	Верхня пам'ять 384 Кбайт	128 Кбайт ПЗП	Програми BIOS
			256 Кбайт ОЗУ	Службова пам'ять
- 640 Кбайт		Стандартна пам'ять 640 Кбайт	576 Кбайт ОЗУ	Програми і дані ОС і користувача
0 – 64 Кбайт			64 Кбайт ОЗУ	Службові програми і дані ОС

Частина 3. Комп’ютерні аспекти теорії систем числення в комп’ютерній арифметиці

3.1. Форми подання та кодування чисел

Комп’ютерна арифметика – сукупність принципів та форм представлення чисельної інформації, методів та алгоритмів виконання арифметичних операцій та обчислення елементарних функцій, що розглядаються на рівні внутрішньої структурної організації технічних засобів комп’ютерів та комп’ютерних систем

(ККС). Комп'ютерна арифметика – частина обчислювальної математики, що орієнтована на логічний, самий нижній, але не фізичний рівень описання обчислювальних структур та процесів в них.

Сукупність принципів і форм подання числової інформації, методів і алгоритмів виконання арифметичних операцій і обчислення елементарних функцій, що розглядаються на рівні внутрішньої структурної організації технічних засобів ККС узагальнено називають комп'ютерною арифметикою.

Комп'ютерна арифметика – частина обчислювальної математики, що орієнтована на логічний рівень описання обчислювальних структур та процесів в них.

Система числення- сукупність прийомів і правил для позначення і найменування чисел. Кожному числу однозначно відповідає деяка кількість, інваріантна щодо способу її вираження, яку називають кількісним еквівалентом числа. У будь – якій системі числення число подають сукупністю символів, які називають цифрами. Кожній цифрі в запису числа також відповідає деяка кількість, що виражається цією цифрою і називається кількісним еквівалентом цифри. Розрізняють непозиційні і позиційні системи числення. Систему числення називають непозиційною, якщо кожній цифрі на будь-якому її місці в запису числа однозначно відповідає один і той же кількісний еквівалент. Такі системи є більш ранніми в історичному плані, наприклад, загальновідома римська нумерація. Однак непозиційні системи числення знаходять відносно обмежене застосування в ККС, тому що вони характеризуються дуже складними і громіздкими алгоритмами подання чисел і виконання обробних арифметичних операцій.

Систему числення називають позиційною, якщо одній і тій же цифрі в залежності від номеру її місця в запису числа – *розряду* – відповідають різні кількісні еквіваленти. Для визначення кількісного еквіваленту багато розрядного запису числа в позиційній системі числення використовують деяку функцію від кількісних еквівалентів цифр. Якщо цією функцією є функція додавання, то система – *адитивна,* якщо функція множення, то система – мультиплікативна. Для більшості практично використовуваних систем це функція десяткового додавання. Для отримання кількісного еквіваленту числа необхідно додавати кількісні еквіваленти цифр за правилами десяткової арифметики.

Якщо в позиційній системі числення кожна цифра зображена заздалегідь визначеним символом, то це *система з безпосереднім поданням цифр.* Існують *системи з кодовим поданням цифр*, де кожна цифра кодується визначеною комбінацією цифрами іншої системи числення.

3.2. Системи числення з безпосереднім представленням цифр

ККС забезпечують автоматичне виконання послідовностей операцій обробки інформації відповідно до заданого алгоритму, що в підсумку і приводить до одержання результату або вирішення задачі. Алгоритм – це

спосіб перетворення інформації, що задається за допомогою скінченної системи правил обов'язкових для виконання.

Для вирішення найрізноманітніших задач ККС повинні мати алгоритмічно повну систему операцій. Тобто ККС повинні бути здатні виконати будь-який і який завгодно складний алгоритм обробки інформації.

Для алгоритмічної повноти системи операцій достатньо чотири операції:

1) пересилання інформації з будь-якої клітинки пам'яті в будь-яку іншу клітинку;
2) додавання (віднімання) одиниці до (від) числа;
3) умовного переходу по точному збігу чисел;
4) безумовного зупину.

Однак використання ККС із мінімальним числом операцій приводило б до надмірної довжини програм, неефективного використання пам'яті, значного погіршення характеристик продуктивності ККС у цілому. Тому в складів складі сучасних ККС є технічні (апаратні) засоби, що розширюють систему операцій до десятків і сотень разів. Цим забезпечується велика функціональна гнучкість ККС. Кількість операцій, що близька до мінімально необхідної для алгоритмічної повноти, характерна лише для деяких найпростіших мікропроцесорів, що використовуються для створення систем керування побутовою технікою, електронними іграшками і т.п.

Інформація, що підлягає обробці на ККС, звичайно подається у вигляді сукупності цифр (чисел) у деякій системі числення, самі ж цифри відображаються сигналами, що мають скінченне число рівнів квантування (найчастіше – два рівні). Перед початком обробки інформації алгоритм обробки повинен бути записаний як послідовність таких операцій, для виконання яких у складі ККС є відповідні засоби. Такий запис і являє собою програму обробки інформації. Будь-яка програма складається з окремих команд, кожна з яких визначає дії технічних засобів ККС по виконанню однієї операції. При цьому числа (слова), з якими виконується операція, називаються операндами.

Всі операції в ККС виконуються як послідовності в просторі і в часі деяких найпростіших, елементарних операцій, що називаються мікрооперaціями. До числа основних класів мікрооперацій у ККС відносяться:

1. Передача (прийом, видача) операнда, слова.
2. Зсув (арифметичний, циклічний, логічний, модифікований) операнда на задане число розрядів.
3. Додавання до операнда або віднімання від нього одиниці (у більш загальному випадку – деякої постійної величини).
4. Порівняння операндів (за принципом "більше–менше–дорівнює").
5. Порозрядні логічні операції (диз'юнкції, кон'юнкції, рівнозначності, додавання за модулем 2).
6. Арифметичне додавання двох операндів, що відповідають числам в одній і тій же системі числення.
7. Перетворення кодів операндів (включаючи інверсію, доповнення, цифрацію, дешифрацію та ін.).

Усі команди в ККС можна умовно розділити на обробні (арифметичні і логічні) і допоміжні команди. У свою чергу допоміжні команди складаються з команд пересилання, розгалуження, звертання до підпрограм і управління вводом-виводом інформації. З числа цих команд найбільш складно реалізуються команди обробки – за спеціальними алгоритмами і з використанням спеціальних технічних засобів. Обробні команди задають, як правило, арифметичні й алгебраїчні операції додавання і віднімання, множення і ділення, додавання і віднімання одиниць (так звані операції інкрементації та декрементації), зсуву, обчислення елементарних функцій і великого числа модифікацій усіх перерахованих операцій. За логічними командами виконуються порозрядні логічні операції й операції порівняння.

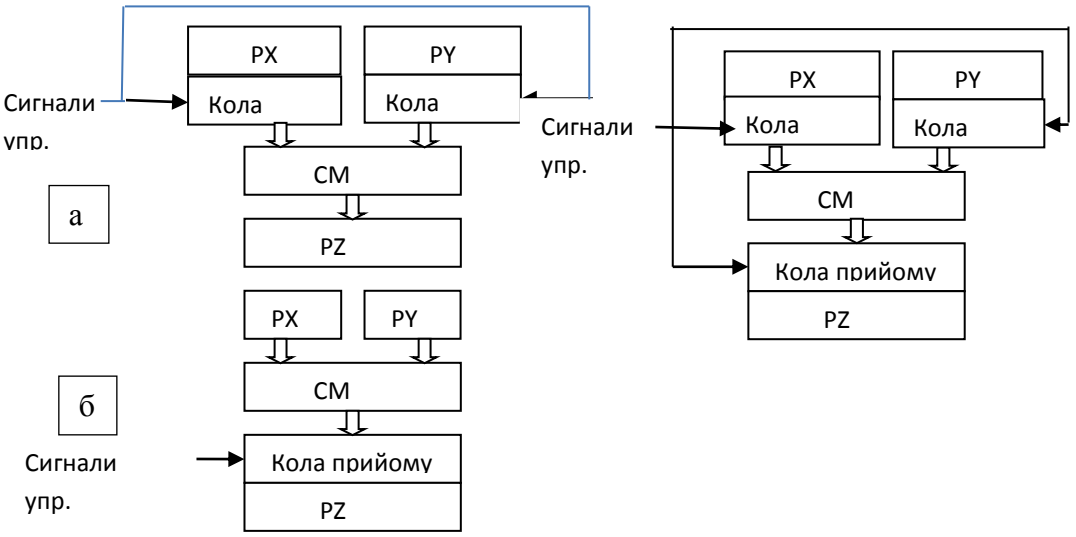


Рис. 24. Схема та порядок виконання будь-якої операції ККС

Команди пересилання забезпечують двосторонній обмін інформацією між внутрішніми вузлами ККС або між вузлами процесора і пам’яттю. Реалізація цих команд зводиться до комутації інформаційних магістралей між вузлами і не викликає труднощів. Команди розгалуження задають умовні і безумовні переходи в програмах. Команди звертання до підпрограм забезпечують перехід до підпрограм і наступне повернення до основної програми. Команди управління вводом-виводом інформації визначають операції з обміну інформацією між внутрішніми вузлами або оперативною пам’яттю з одного боку та пристроями зовнішньої пам’яті або периферійними пристроями з іншого боку. Ці команди сукупно називають також командами управління обчислювальним процесом. Виконання цих команд не пов’язано із змістовною обробкою операндів і полягає в реалізації спеціальних алгоритмів формування адрес команд інших типів.

Реальні команди досить часто можна віднести до декількох типів одночасно. Наприклад, команда додавання двох операндів з переходом за

переповненням розрядної сітки є і командою обробки, і командою розгалуження.

Мікроархітектура – апаратна організація і логічна структура ККС, структура і можливості засобів для збереження даних під час виконання операцій, структура засобів управління, інформаційних магістралей і арифметико-логічних блоків. Мікроархітектури – це найбільш істотні риси, ознаки, характеристики ККС з погляду їх розробника і виробника.

Макроархітектура – це подання ККС з погляду програміста-користувача, для якого найбільш важливими є система операцій, типи оброблюваних даних, формати даних і команд, режими й особливості адресації та інші чисто зовнішні атрибути. Реалізація команд обробки інформації зводиться до деякого алгоритму. Загальний стандартний спосіб запису алгоритмів називають *алгоритмічною системою.* Відомі декілька алгоритмічних систем, орієнтованих на інженерне застосування, однак найбільш наочними і зручними із них слід вважати *граф-схеми алгоритмів* (ГСА).

Для запису алгоритмів у вигляді ГСА використовують графічні символи. які називають вершинами ГСА. Початок і кінець алгоритмів позначають вершинами у вигляді еліпсів. Для позначення початкової вершини використовують абревіатуру *BG(BEGIN),* а для позначення кінцевої вершини – слово *END*. Будь-який алгоритм повинен мати тільки по одній початковій і кінцевій вершині. Операторною вершиною у вигляді прямокутника називають умовне або змістовне позначення виконуваної мікрооперації. Змістовне позначення зручно записувати за допомогою *оператора присвоювання* :=. Наприклад: *X:=X+R*. Сукупність операторів присвоювання, записаних в одній і тій же операторній вершині, виконується одночасно (реально вони виконуються як просторово розділені дії – на різних функціональних вузлах ККС). *Умовна* вершина у вигляді ромбу відповідає логічній умові, яку необхідно перевірити. Два можливих результати такої перевірки позначають цифрами 0 (умова не виконується) і 1 (умова виконується) або відповідно словами “Ні “ та “Так “.

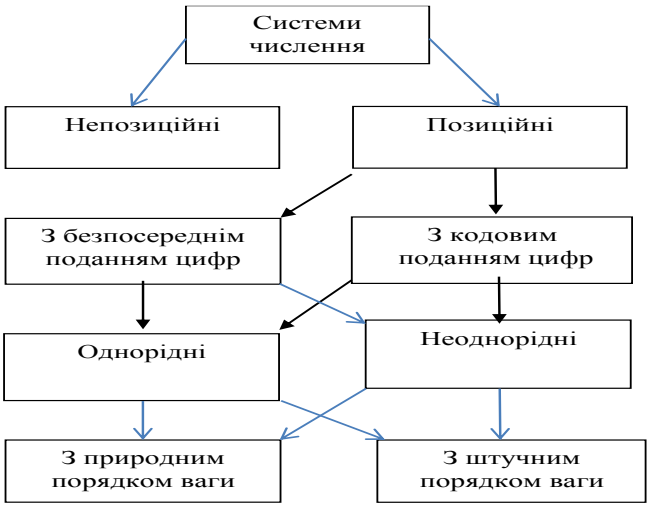


Рис. 25. Діаграма класифікації систем числення ККС

Вимоги до систем числення в обчислювальній техніці: однозначність; кінцевість; ефективність.

Однозначність – кожному числу з певним кількісним еквівалентом повинно відповідати єдине його подання в заданій системі числення і навпаки.

Скінченність – кожному цілому числу з певним кількісним еквівалентом в заданій системі числення повинно відповідати його подання (операнд, слово) скінченої довжини.

Ефективність – повинен існувати алгоритм, за допомогою якого за скінчене число кроків здійснювався би перехід від подання числа в заданій системі числення до його кількісного еквівалента, записаного в деякій загальноприйнятій системі (коротше: до самого числа). При переході від кількісного еквівалента числа до його подання в заданій системі повинні існувати алгоритми, які для цілого числа реалізують цей перехід за скінчене число кроків, а для дробового числа дозволяють за скінчене число кроків одержати подання числа, кількісний еквівалент якого відрізняється від числа не більше ніж на задану величину похибки.

ККС забезпечують автоматичне виконання послідовностей операцій обробки інформації відповідно до заданого алгоритму, що в підсумку і приводить до одержання результату або вирішення задачі. Алгоритм – це спосіб перетворення інформації, що задається за допомогою скінченної системи правил обов’язкових для виконання.

Для вирішення найрізноманітніших задач ККС повинні мати алгоритмічно повну систему операцій. Тобто ККС повинні бути здатні виконати будь-який і який завгодно складний алгоритм обробки інформації.

Для алгоритмічної повноти системи операцій достатньо чотири операції:

1) пересилання інформації з будь-якої клітинки пам’яті в будь-яку іншу клітинку;
2) додавання (віднімання) одиниці до (від) числа;
3) умовного переходу по точному збігу чисел;
4) безумовного зупину.

Однак використання ККС із мінімальним числом операцій приводило б до надмірної довжини програм, неефективного використання пам’яті, значного погіршення характеристик продуктивності ККС у цілому. Тому в складів складі сучасних ККС є технічні (апаратні) засоби, що розширюють систему операцій до десятків і сотень разів. Цим забезпечується велика функціональна гнучкість ККС. Кількість операцій, що близька до мінімально необхідної для алгоритмічної повноти, характерна лише для деяких найпростіших мікропроцесорів, що використовуються для створення систем керування побутовою технікою, електронними іграшками і т.п.

Інформація, що підлягає обробці на ККС, звичайно подається у вигляді сукупності цифр (чисел) у деякій системі числення, самі ж цифри відображаються сигналами, що мають скінченне число рівнів квантування (найчастіше – два рівні). Перед початком обробки інформації алгоритм обробки

повинен бути записаний як послідовність таких операцій, для виконання яких у складі ККС є відповідні засоби. Такий запис і являє собою програму обробки інформації. Будь-яка програма складається з окремих команд, кожна з яких визначає дії технічних засобів ККС по виконанню однієї операції. При цьому числа (слова), з якими виконується операція, називаються операндами.

Всі операції в ККС виконуються як послідовності в просторі і в часі деяких найпростіших, елементарних операцій, що називаються мікрооперaціями. До числа основних класів мікрооперацій у ККС відносяться:

1. Передача (прийом, видача) операнда, слова.
2. Зсув (арифметичний, циклічний, логічний, модифікований) операнда на задане число розрядів.
3. Додавання до операнда або віднімання від нього одиниці (у більш загальному випадку – деякої постійної величини).
4. Порівняння операндів (за принципом ”більше–менше–дорівнює”).
5. Порозрядні логічні операції (диз’юнкції, кон’юнкції, рівнозначності, додавання за модулем 2).
6. Арифметичне додавання двох операндів, що відповідають числам в одній і тій же системі числення.
7. Перетворення кодів операндів (включаючи інверсію, доповнення, цифрацію, дешифрацію та ін.).

Усі команди в ККС можна умовно розділити на обробні (арифметичні і логічні) і допоміжні команди. У свою чергу допоміжні команди складаються з команд пересилання, розгалуження, звертання до підпрограм і управління вводом-виводом інформації. З числа цих команд найбільш складно реалізуються команди обробки – за спеціальними алгоритмами і з використанням спеціальних технічних засобів. Обробні команди задають, як правило, арифметичні й алгебраїчні операції додавання і віднімання, множення і ділення, додавання і віднімання одиниць (так звані операції інкрементації та декрементації), зсуву, обчислення елементарних функцій і великого числа модифікацій усіх перерахованих операцій. За логічними командами виконуються порозрядні логічні операції й операції порівняння.

Команди пересилання забезпечують двосторонній обмін інформацією між внутрішніми вузлами ККС або між вузлами процесора і пам’яттю. Реалізація цих команд зводиться до комутації інформаційних магістралей між вузлами і не викликає труднощів. Команди розгалуження задають умовні і безумовні переходи в програмах. Команди звертання до підпрограм забезпечують перехід до підпрограм і наступне повернення до основної програми. Команди управління вводом-виводом інформації визначають операції з обміну інформацією між внутрішніми вузлами або оперативною пам’яттю з одного боку та пристроями зовнішньої пам’яті або периферійними пристроями з іншого боку. Ці команди сукупно називають також командами управління обчислювальним процесом. Виконання цих команд не пов’язано із змістовною

обробкою операндів і полягає в реалізації спеціальних алгоритмів формування адрес команд інших типів.

Реальні команди досить часто можна віднести до декількох типів одночасно. Наприклад, команда додавання двох операндів з переходом за переповненням розрядної сітки є і командою обробки, і командою розгалуження.

3.3. Форми комп'ютерного представлення чисел з фіксованою комою

Комп'ютерна арифметика – сукупність принципів та форм представлення чисельної інформації, методів та алгоритмів виконання арифметичних операцій та обчислення елементарних функцій, що розглядаються на рівні внутрішньої структурної організації технічних засобів комп'ютерів та комп'ютерних систем (ККС).

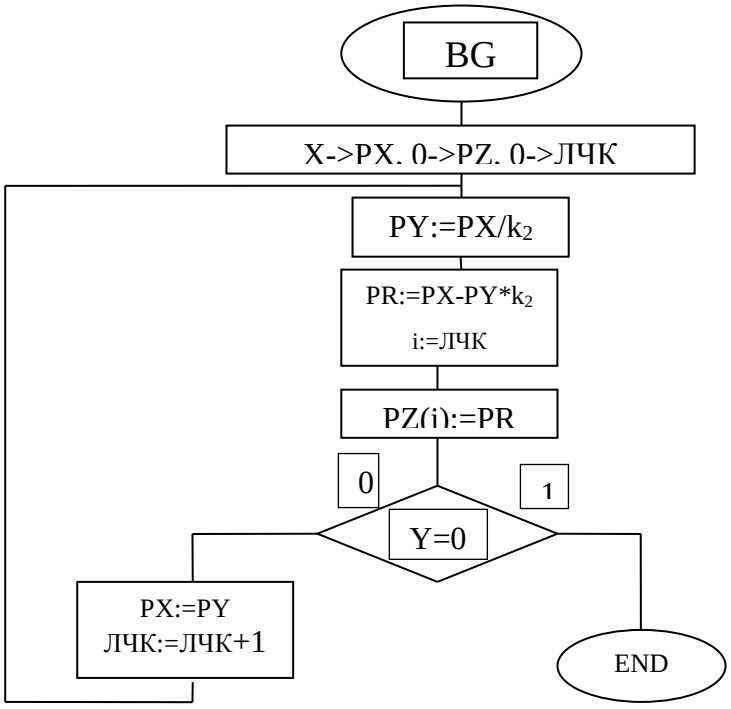


Рис. 26. Алгоритм переводу цілих чисел із системи числення з основою k_1 у систему з основою k_2 у випадку $k_1 > k_2$.

В таблиці 9 приведене зображення деяких цілих чисел і дробових чисел у системах числення з основами k = 10,2,8,16. Шість старших цифр шістнадцяткової системи позначені буквами *A,B,C,D,E,F* (відповідно десятковим числам 10,11,12,13,14,15).

Таблиця 9

k				k			
10	2	8	16	10	2	8	16
0	00000	0	0	15	01111	17	F
1	00001	1	1	16	10001	20	10
2	00010	2	2	17	10001	21	11
3	00011	3	3	18	10010	22	12
4	00100	4	4	19	10011	23	13
5	00101	5	5	20	10100	24	14
6	00110	6	6	0,1	0,0001,,,	0,06…	0,19…
7	00111	7	7	0,2	0,0011…	0,14…	0,33…
8	01000	10	8	0,3	0,0100…	0,23…	0,4C…
9	01001	11	9	0,4	0,0110…	0,31…	0,66…
10	01010	12	A	0,5	0,1	0,4	0,8
11	01011	13	B	0,6	0,1001…	0,46…	0,99…
12	01100	14	C	0,7	0,1011…	0,54…	0,B3…
13	01101	15	D	0,8	0,1101…	0,63…	0,CC…
14	01110	16	E	0,9	0,1110…	0,71…	0,E6…

Таблиця 10

	PX	PY	PR	PZ	Лік. кроків
	57			000000	
0		28	1		
				1	
	28				1
1		14	0		
				01	
	14				2
2		7	0		
				001	
	7				3
3		3	1		
				1001	
	3				4
4		1	1		
				11001	
	1				5
5		0	1		
		END			
				111001	

В таблиці 10 приведений послідовний порядок переведення числа X = 57 в системі числення $K_1 = 10$ в систему числення $K_2 = 2$. В таблиці представлені: 1 рядок – початковий стан регістрів, а кожний наступний рядок відповідає станам регістрів після чергової мікрооперації ділення.

Таблиця 11

75	2							
74	37	2						
1	36	18	2					
K_2^0	1	18	9	2				
	K_2^1	0	8	4	2			
		K_2^2	1	4	2	2		
			K_2^3	0	2	1	2	
				K_2^4	0	0	0	
					K_2^5	1		
						K_2^6		

Ручне переведення цілих чисел 75 із системи числення $K_1 = 10$ в систему числення $K_2 = 2$ записи ведуться “в стовпчик” (подібно ручному діленню “в стовпчик”), приведене в таблиці 11 та представляє Z = 1001011.

Для переводу числа із системи числення з від’ємною основою в систему з такою ж додатною основою необхідно спершу також розділити операнд X на дві частини A та B. Непарні розряди числа A дорівнюють нулю, а парні розряди числа A дорівнюють парним розрядам числа X. Число B має в непарних розрядах ті ж цифри, що і число X в непарних розрядах. Парні розряди числа B дорівнюють нулю. Далі від A віднімають B, якщо X додатне, або від B віднімають A, якщо X від’ємне. Знак X визначається знаком ваги старшого не рівного нулю розряду. ГСА такого переводу для цілих чисел приведена на рис. 26. Для дробових операндів рахунок розрядів необхідно вести в праву сторону, починаючи з -1.

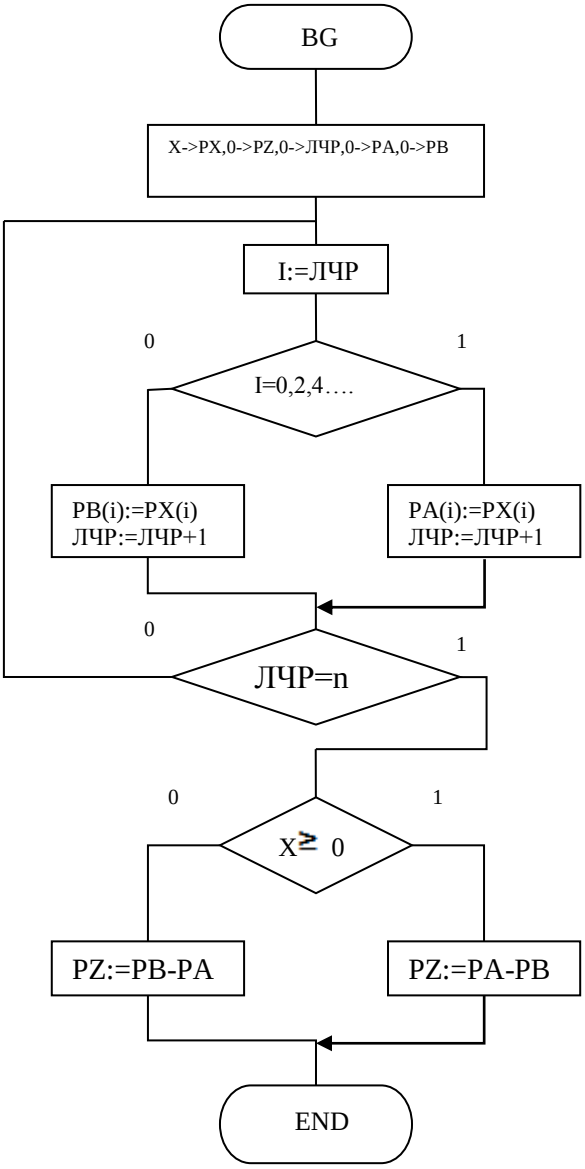


Рис. 27. ГСА переводу числа із системи числення з від’ємною основою

3.4. Форми комп’ютерного представлення чисел, приклади переведення десяткових чисел у інші системи числення

Система числення – сукупність прийомів і правил для позначення і найменування чисел. Кожному числу однозначно відповідає деяка кількість, інваріантна щодо способу її вираження, яку називають кількісним еквівалентом числа. У будь-якій системі числення число подають сукупністю символів, які називають цифрами. Кожній цифрі в запису числа також відповідає деяка кількість, що виражається цією цифрою і називається кількісним еквівалентом цифри. Розрізняють непозиційні і позиційні системи числення.

Систему числення називають позиційною, якщо одній і тій же цифрі в залежності від номеру її місця в запису числа – розряду – відповідають різні кількісні еквіваленти: адитивні та мультиплікативні.

ККС забезпечують автоматичне виконання послідовностей операцій обробки інформації відповідно до заданого алгоритму, що в підсумку і приводить до одержання результату або вирішення задачі. Алгоритм – це спосіб перетворення інформації, що задається за допомогою скінченної системи правил обов'язкових для виконання.

Усі команди в ККС можна умовно розділити на обробні (арифметичні і логічні) і допоміжні команди. У свою чергу допоміжні команди складаються з команд пересилання, розгалуження, звертання до підпрограм і управління вводом-виводом інформації. З числа цих команд найбільш складно реалізуються команди обробки – за спеціальними алгоритмами і з використанням спеціальних технічних засобів. Обробні команди задають, як правило, арифметичні й алгебраїчні операції додавання і віднімання, множення і ділення, додавання і віднімання одиниць (так звані операції інкрементації та декрементації), зсуву, обчислення елементарних функцій і великого числа модифікацій усіх перерахованих операцій. За логічними командами виконуються порозрядні логічні операції й операції порівняння.

Команди пересилання забезпечують двосторонній обмін інформацією між внутрішніми вузлами ККС або між вузлами процесора і пам'яттю. Реалізація цих команд зводиться до комутації інформаційних магістралей між вузлами і не викликає труднощів. Команди розгалуження задають умовні і безумовні переходи в програмах. Команди звертання до підпрограм забезпечують перехід до підпрограм і наступне повернення до основної програми. Команди управління вводом-виводом інформації визначають операції з обміну інформацією між внутрішніми вузлами або оперативною пам'яттю з одного боку та пристроями зовнішньої пам'яті або периферійними пристроями з іншого боку. Ці команди сукупно називають також командами управління обчислювальним процесом. Виконання цих команд не пов'язано із змістовною обробкою операндів і полягає в реалізації спеціальних алгоритмів формування адрес команд інших типів.

Для представлення чисел застосовують цифри або, точніше кажучи, цифрові ряди або цифрові слова, що утворяться шляхом упорядкування кінцевого числа знаків з кінцевої множини основних знаків (алфавіту системи числення).

Розрізняють два типи систем числення: ***непозиційні*** (прикладом яких може служити римська система числення) і ***позиційні.*** У позиційній системі числення вибирають деяке натуральне число p, більше одиниці, і використовують його як базисне число (p-йкова система числення); для ***p,*** рівного одиниці, позиційної системи числення не існує. Вводять ***p*** основних знаків, які називаються ***цифрами.*** Ці знаки використовують для утворення цифрових послідовностей, що служать для представлення натуральних чисел. Будемо позначати цифри так: a_o, a_1, a_2,..., a_i . Кожної цифрової послідовності, утвореної тільки з однієї цифри, однозначно протиставимо одне з a_i) перших

натуральних чисел або нуль. Тоді всяке натуральне число ***a*** має точно одне представлення в р-йковій системі числення:

$$a = b_s \mathbf{p}^{s} + b_{s-1} \mathbf{p}^{s-1} + \ldots + b_0 \mathbf{p}^{0}$$

де ***p*** позначає однозначно визначене натуральне число, $\boldsymbol{b_s}$ цифри, причому b_s – цифра, відмінна від цифри, що відповідає нулеві.

Ряд знаків $\boldsymbol{b_s, b_{s-i}, \ldots, b_o}$ є цифровим представленням числа ***a.*** Число нуль представляється послідовністю ***i***, що складається з однієї цифри, яка відповідає нулеві. У позиційній системі число, що представляється, утвориться адитивно, причому кожна цифра $\boldsymbol{b_s}$ має ***числове*** значення (число, що відповідає цифрі $\boldsymbol{b_s}$) і ***позиційне*** значення (вага) p^s. Адитивний внесок цієї цифри в значення числа дорівнює $\boldsymbol{b_s p^s}$.

Десяткова система: p = 10, цифри 0,1, 2,3,4, 5, **6**, 7, 8, 9.

Наприклад: $3* 10^5 + 2*10^4 + 0* 10^3+0* 10^2+9*10^1+ 1* 10° = 320091.$

Двійкова система (або ***бінарна*** система): p = 2, цифри 0, 1.

Наприклад: $1* 2^6+0* 2^5+1* 2^4+0* 2^3+0* 2^2+1* 2^1+1* 2°=1010011.$

Записана в двійковій системі числення послідовність 1010011, позначає число, значення якого в десятковій системі є 83; дійсно, 64 + 16 + 2 + 1 = 83.

Щоб мати можливість представляти в позиційній системі також і деякі раціональні числа, позиційні значення цифр у записі числа поширюють на ступені ***p*** з негативними показниками. Для виділення позиційного значення (ваги) ***p°*** необхідний додатковий знак (знак дробі), у якості якого звичайно береться кома (іноді крапка). Цей знак розміщається в цифровій послідовності безпосередньо праворуч від цифри з позиційним значенням. Відповідно до цього цифрова послідовність $\boldsymbol{b_s, b_{s-i}, \ldots, b_o, b_{-i}, b_{-s-i}, b_{-s}}$ позначає число, задане р-йковим уявленням:

$$a = b_n \mathbf{p}^{n} + b_{n-1} \mathbf{p}^{n-1} + \ldots + b_{-n} \mathbf{p}^{-n}.$$

Цифра b_s може мати числове значення нуль.

Приклад 1.

Десяткова система: $23{,}040 = 2* 10^1 + 3* 10° + 0* 10^{-1} + 4* 10^{-2} + 0* 10^{-3}.$

Двійкова система: $10{,}011 = 1* 2^1 + 0* 2° + 0* 2^{-1} + 1* 2^{-2} + 1* 2^{-3}.$

Запис числа 10,011 у десятковій системі є 2,375, дійсно

$$2 + 0{,}25 + 0{,}125 = 2{,}375.$$

Кожне число, яке представлено в позиційній системі числення послідовністю цифр кінцевої довжини, є ***раціональним числом.*** Навпаки, у кожній позиційній системі можна представити точно тільки деяку підмножину раціональних чисел (що залежить від вибору ***p***). Наприклад, раціональне число 1/3 не може бути представлене в десятковій системі числення у виді кінцевої послідовності цифр; 1/25 у десятковій системі записується як 0,04, а в двійковій системі числення 1/25 кінцевою послідовністю цифр представлено бути не може.

Якщо ***a/b*** – раціональне число (***a*** і ***b*** – взаємно прості натуральні числа), то може бути точно представлене в позиційній системі з базисом ***p*** тоді і тільки

тоді, коли кожен простий множник у розкладанні числа ***a/b*** є простим множником у розкладанні ***p.***

Таким чином, у десятковій системі представляються тільки такі раціональні числа ***a/b*** (***a, b*** — взаємно прості), у яких ***b*** містить лише прості множники 2 і 5.

Системи числення у комп'ютерній техніці.

Поняття системи числення, з точки зору інформатики, має трошки інший зміст, ніж у математиці. Якщо, у побуті поширеною є десяткова система числення (числа, які утворюються цифрами 0,1,2, ..., 9), то у комп'ютерній техніці використовується двійкова система (числа утворюються з цифр 0 та 1). Це пов'язано насамперед з можливістю зберігання, передачі та обробки інформації у комп'ютерах лише у двійковій системі числення. Наприклад, цифра 0 відповідає відсутності струму або рівня напруги, а цифра 1 навпаки – наявності. Зберігання та обробка інформації за іншою системою числення є складною технічної задачею, вирішення якої потребувало розробки спеціальних технічних пристроїв, які були б набагато складніше існуючих цифрових пристроїв (сучасних мікросхем) та працювали б з більшою кількістю помилок.

Тому розглянемо поняття системи числення, з точки зору інформації, яка обробляється у комп'ютерній техніці.

Основною одиницею збереження даних у комп'ютері є біт. У більшості комп'ютерів вісім бітів об'єднані в байт, при цьому кожен біт байта може бути встановлений (=1) або скинутий (= 0), допускаючи 256 різних варіантів. Таким чином, в одному байті можна представити 256 різних символів або ціле число в діапазоні від 0 до 255. Ці числа можуть записуватися у двійковій або шістнадцятковій формі – їх значення при цьому не змінюються. Замість того щоб говорити, що в одному байті можуть зберігатися числа від 0 до 255, можна сказати, що можуть зберігатися двійкові числа від 00000000 до 11111111 або шістнадцяткові числа від 00 до 1Н. Оскільки можна переплутати різні форми, то двійкові і шістнадцяткові числа відзначаються спеціальним образом. У мові асемблера за двійковими числами записується буква В, а за шістнадцятковими числами – буква Н, наприклад, 11111111В або РРН.

Двійкові числа.

Коли вміст байта представляється в двійковій формі, то потрібно 8 цифр. Кожна цифра відповідає одному бітові; біти нумеруються від 0 до 7. Як і в десяткових числах, цифри розташовуються зправа наліво, від молодших до старших розрядів. На відміну від десяткових чисел, у яких кожна наступна цифра в 10 разів більше своєї сусідки праворуч, двійкові цифри збільшуються тільки в два рази. Таким чином, сама права цифра визначає одиниці, друга – двійки, третя – четвірки і т.д. до значення 128 для восьмої цифри байта. Це означає, що якщо перша цифра дорівнює 1, то додаток до неї 1 приводить до того, що вона стане 0, а 1 буде перенесена в другу цифру, як для десяткових чисел 9 + 1 = 0 і переніс одиниці в наступний розряд (табл. 12).

Таблиця 12

000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

У цій послідовності більшість нулів ліворуч необов'язкові, тобто числа можна записати й у виді 0, 1, 10, 11, 100, 101 і т.д. Для полегшення уявлення зв’язку між двійковими та десятковими числами доцільно використовувати наступну таблицю:

Таблиця 13

Біт	7	6	5	4	3	2	1	0
значення	2^7=128	2^6=64	2^5=32	2^4=16	2^3=8	2^2=4	2^1=2	2^n=T

Коли зустрічається двійкове число 10000001, де встановлені біти 7 і 0. Біт 7 відповідає 128, а біт 0-1, тому десяткове значення байта дорівнює 129. Навпаки, щоб визначити послідовність бітів для числа 65, представимо його як 64 + 1, що відповідає 01000001B.

Шістнадцяткові числа.

У той час як у двійкових числах кожна наступна цифра вдвічі більше попередньої, у шістнадцяткових числах кожна наступна цифра більше в 16 разів. У десяткових чисел перша позиція відповідає одиницям, друга – десяткам, третя – сотням. У двійкових числах перша позиція відповідає одиницям, друга двійкам, третя – четвіркам. У шістнадцяткових числах перша позиція відповідає одиницям, друга – 16, третя – 256 і т.д. Це означає, що коли в позиції одиниць розташована цифра 9, то додаток одиниці не приводить до переносу в наступний розряд, як це було б у випадку десяткових чисел. Шістнадцяткові числа, для запису чисел, використовують цифри 0,1,2 ... 9 та перші 6 букв латинського алфавіту в якості додаткових цифрових символів:

Таблиця 14

0	0	0100	4	100	8	11	C
0	1	0101	5	100	9	11	D
0	2	0110	6	101	A	11	E
0	3	0111	7	101	B	11	F

Корисність шістнадцяткових чисел спирається на той факт, що одна шістнадцяткова цифра описує вміст рівно 1/2 байта. Наприклад, у числі B6, B відповідає старшим чотирьом бітам байта, а 6 – молодшим чотирьом бітам

(чотири біти, узяті разом, називаються тетрадою). Нескладно обчислити двійковий еквівалент четвірки бітів. ВН = 1111В, а 6Н = 0110В (НиВ – суфікси, що допомагають відрізнити 11 двійкове від 11 десяткового і 11 шістнадцяткового). Таким чином, число В6Н представляє послідовність бітів 11110110. Двобайтове число (ціле) може дорівнювати і 6ВВ6Н. У цьому випадку послідовність бітів для нього має вигляд 0110111111110110. Якщо число складається тільки з трьох цифр, то верхня половина старшого байта дорівнює нулеві, наприклад, числу В6ВН відповідає послідовність бітів 0000111101101111.

3.5. Переведення десяткових чисел у двійкову, вісімкову та шістнадцяткову системи числення

Переведення чисел з однієї системи числення в іншу

Нехай потрібно перевести число $N_{(p)}$, що задане в системі числення з основою p, в його представлення $N_{(q)}$ в системі числения з основою q.

Існують загальні правила переведення:

- для переведення целої частини числа користуються правилом послідовного ділення;
- для переведення дробової частини користуються правилом послідовного множення.

3.5.1. Правило переведення цілої частини – правило послідовного ділення:

Для переведення цілої частини числа $N_{(p)}$ з основою p в ціле число $N_{(q)}$ з основою q необхідно послідовно ділити цілу частину числа $N(p)$ та одержувані часткові на основу нової системи числения q, представлене в системе счисления p, до тих пір, доки часткове не стане меньше q. Старшою цифрою запису числа $N_{(q)}$ служить останнє часткове, а слідуючі за нею цифри являються залишками від ділення часткових часток, записаними в порядку зворотньому Їх отриманню.

Правило переведення дробової частини – правило послідовного множення:

Для переведення дробової частини числа $N\ (p)$ з основою p в дробове число $N(q)$ з основою q необхідно послідовно множити вихідну дробову частину числа $N\ (p)$ і одержувані часткові дробової частини множень на основу нової системи числення q, представлене в системі числення p. Кількість множень визначається заданою точністю. Цілі частини отримуваних множень дають послідовність цифр представлення дробу в системі числення з основою q.

1.1. Приклад перетворення в указані системи числення числа **142.378**$_{(10)}$. Зверніть увагу, що окремо перетворюється ціла частина числа, використуя правило послідовного ділення, та окремо перетворюється дробова частина числа, використуя правило послідовного множення.

Перетворення цілої частини числа 142$_{(10)}$ в двійкове, вісімкове та шістнадцяткове числа:

2-я с.с.	Часткова частка	Залишок	8-я с.с.	Часткова частка	Залишок	16-я с.с.	Часткова частка	Залишок
142:2 =	71	**0**	142:8 =	17	**6**	142:16 =	**8**	**14 ⇒ E**
71:2 =	35	**1**	17:8 =	**2**	**1**			
35:2 =	17	**1**						
17:2 =	8	**1**						
8:2 =	4	**0**						
4:2 =	2	**0**						
2:2 =	**1**	**0**						
142$_{(10)}$ = 10001110$_{(2)}$			142$_{(10)}$ = 216$_{(8)}$			142$_{(10)}$ = 8E$_{(16)}$		

Стрілками показуєтьсяся порядок запису цифр в новій системі числення.

Для шістнадцяткової системи числення цифри **10**, **11**, **12**, **13**, **14**, **15** позначаються відповідно латинськими буквами **A, B, C, D, E, F.**

Перетворення дробової частини числа 0.378$_{(10)}$ в двійкову, вісімкову та шістнадцяткову

2-я с.с.	Частковий добуток	Ціла частина	8-я с.с.	Частковий добуток	Ціла частина	16-я с.с.	Частковий добуток	Ціла частина
0.378×2 =	0.756	**0**	0.378×8 =	3.024	**3**	0.378×16 =	6.048	**6**
0.756×2 =	1.512	**1**	0.024×8 =	0.192	**0**	0.048×16 =	0.768	**0**
0.512×2 =	1.024	**1**	0.192×8 =	1.536	**1**	0.768×16 =	12.288	**12⇒C**
0.024×2 =	0.048	**0**	0.536×8 =	4.288	**4**	0.288×16 =	4.608	**4**
0.048×2 =	0.096	**0**	0.288×8 =	2.304	**2**			
0.096×2 =	0.192	**0**						
0.192×2 =	0.384	**0**						
0.384×2 =	0.768	**0**						
0.768×2 =	1.536	**1**						
0.378$_{(10)}$ = 0.011000001$_{(2)}$			0.378$_{(10)}$ = 0.30142$_{(8)}$			0.378$_{(10)}$ = 0.60C4$_{(16)}$		

Результат:

142.378$_{(10)}$= 10001110.011000001$_{(2)}$= 216.30142$_{(8)}$= 8E.60C4$_{(16)}$

3.5.2. Переведення двійкових, вісімкових та шістнадцяткових чисел в десяткову систему числення

В основі перетворення двійкового, вісімкового та шістнадцяткового чисел в десяткову систему числення лежить представлення любого числа в вигляді поліному:

$$N_{(p)} = a_{n-1} \cdot p^{n-1} + a_{n-2} \cdot p^{n-2} + \ldots + a_1 \cdot p^1 + a_0 \cdot p^0 + a_{-1} \cdot p^{-1} + \ldots a_{-m} \cdot p^{-m} = \sum_{i=-m}^{n-1} a_i \cdot p^i$$

де a_i – одна з дозволених цифр системи числення за основою p, що знаходиться на i-й позиції; n та m – число цілих та дробових розрядів числа, відповідно.

3.5.3. Переведення вісімкових та шістнадцяткових чисел в двійкові

Переведення вісімкових та шістнадцяткових чисел в двійкові дуже простий: достатньо кожну цифру замінити еквівалентною їй двійковою тріадою (трійкою цифр) або тетрадою (четвіркою цифр). При цьому незначущі нулі зліва від цілої частини числа, та справа від дробової частини відкидуютьсяся.

Розбір прикладу перетворення чисел $237.4_{(8)}$ и $1A3.F8_{(16)}$ в двійкову систему числення. Пам’ятайте, що кожна цифра замінюється еквівалентною їй двійковою триадою (трійкою цифр) або тетрадою (четвіркою цифр). При цьому незначущі нули ліворуч від цілої частини числа та праворуч від дробової частини відкидуються.

8-я с.с.	Тріади				16-я с.с.	Тетради				
$237.4_{(8)}=$	010	011	111.	100	$1A3.F8_{(16)}=$	0001	1010	0011.	1111	1000
	↓	↓	↓	↓		↓	↓	↓	↓	↓
	2	3	7	4		1	A	3	F	8
Результ.:	$10011111.1_{(2)}$				Результ.:	$110100011.11111_{(2)}$				

3.5.4. Переведення двійкових чисел у вісімкову та шістнадцяткову системи числення

Переклад двійкового числа в вісімкову і шістнадцяткову системи здійснюється також просто: двійкове число розбивається вправо і вліво від точки, яка відділяє цілу частину від дробової, на тріади (для вісімкової системи числення) або тетради (для шістнадцяткової системи числення). При необхідності крайню ліву тріаду (тетраду) цілої частини і крайню праву (дробової частини) доповнюють нулями, а потім кожну тріаду (тетраду) замінюють вісімковою (шістнадцятковою) цифрою.

Розбір прикладу перетворення числа 101010111111101.1101001 (2) в зазначені системи числення. Пам'ятайте, що двійкове число розбивається вправо і вліво від точки, відділяючи цілу частину від дробової, на тріади (для вісімкової системи числення) або тетради (для шістнадцяткової системи числення). При необхідності крайню ліву тріаду (тетраду) цілої частини і крайню праву (дробової частини) доповнюють нулями, а потім кожну тріаду (тетраду) замінюють вісімковою (шістнадцятковою) цифрою.

	Напрямок розбиття цілої частини на тріади ←					Напрямок розбиття дробової частини на тріади →		
Початкове число	(0)10	101	011	111	101.	110	100	1(00)
	↓	↓	↓	↓	↓	↓	↓	↓
Цифри 8 с.с.	2	5	3	7	5	6	4	4
Результат:	$25375.644_{(8)}$							

	Напрямок розбиття цілої частини на тетради ←				Напрямок розбиття дробової частини на тетради →	
Початкове число	(00)10	1010	1111	1101.	1101	001(0)
	↓	↓	↓	↓	↓	↓
Цифри 16 с.с.	2	A	F	D	D	2
Результат:	$2AFD.D2_{(16)}$					

3.5.5. За зразком п. 3.5.1 переводяться числа, отримані завдяки розрахункам згідно п. 3.5.2 двійкові числа, у вісімкову (шістнадцяткову) системи числення. Результати порівнюються з початковими числами, що представлені у вісімковій (шістнадцятковій) системах числення.

Таким чином, особливості функціонування комп'ютерної техніки, можливість здійснення обробки, передачі та зберігання інформації лише у дискретному вигляді призвели, що використовувати звичайну для побуту десяткову систему числення стало неефективно. Єдиною системою придатною для використання у комп'ютерах стала двійкова система числення. Проте вона характеризується дуже поганим візуальним уявленням чисел, які представляються у двійковій системі. Велика кількість цифр, які необхідні для представлення невеликих, у розумінні десяткової системи числення, та однотонність цифр (використовуються лише дві цифри – 0 та 1) робить двійкову систему числення не придатною для візуального уявлення даних. Тому, для представлення даних користувачу широкого поширення набула шістнадцяткова система числення. Шістнадцяткові числа набагато легше читати, чим двійкові. А після невеликої практики виявляється, що працювати з ними (з точки зору розробки програмного забезпечення на низькому програмному рівні) набагато зручніше, ніж з десятковими.

Частина 4. Алгоритми виконання арифметичних операцій

4.1. Алгоритми додавання і віднімання двійково-десяткових операндів

У сучасних універсальних ККС переважно використовуються дві форми подання чисел. Одна з них одержала назву *форми з фіксованою комою (fixed point)* або натуральної форми. Інше подання чисел – *форма з плаваючою комою (floating point)*, яка також має й інші назви, а саме: нормальна, експоненціальна, напівлогарифмічна або так званий науковий запис.

Будь-яке числоX в позиційній однорідній системі числення з основою k можна записати як $X = M*k^p$, де M називають мантисою числа X, а P – *порядком* цього числа (наприклад, довжина екватора Землі записують як $X = 0{,}4*10^8$ *метрів,* то $M = 0{,}4$, $k = 10$, $P = 8$). Інакше порядок числа називають також *експонентою.*

Якщо в ККС кожному числу X однозначно відповідає мантиса M, а порядок P фіксований для всіх чисел, то говорять, що число X подане у формі з фіксованою комою. Порядок P в цьому випадку встановлюється заздалегідь при підготовці задачі до розв'язку, при чому число k^{-p}є масштабним коефіцієнтом, на який необхідно помножити вихідні дані для того, щоб уникнути переповнення розрядної сітки.

Якщо ж кожному числу X однозначно відповідає пара M та P, то таке подання називають формою з плаваючою комою.

При поданні чисел у формі з фіксованою комою положення коми залишається незмінним для всіх чисел, з якими оперує ККС. З метою спрощення процедури визначення масштабних коефіцієнтів кому звичайно фіксують перед старшим розрядом мантиси або після її молодшого розряду. У першому випадку ККС оперує з числами, що за абсолютною величиною менше одиниці, тобто, з правильними дробами. Якщо кількість розрядів для подання мантиси дорівнює n, то мінімальне за абсолютною величиною, але не рівне нулю, число, яке можна записати в цьому випадку, визначається одиницею в розряді з найменшою вагою і дорівнює k^{-n}. Максимальне за абсолютною величиною число в цьому випадку має цифри $k-1$ у всіх розрядах і дорівнює $1-k^{-n}$. При використанні такої форми подання чисел зручно виконувати операцію множення, оскільки множення правильних дробів ніколи не приводить до переповнення розрядної сітки.

В другому випадку ККС виконує операції з цілими числами, абсолютні величини яких знаходяться в межах від 1 (мінімального, але не рівного нулю числа) до k^n-1.

Розрядна сітка для подання чисел (*формат даних*) з фіксованою комою звичайно складається з двох полів. Перше поле містить один (у пам'яті) або два (у процесорі) розряди для подання знака операнда, друге поле (nрозрядів) служить для подання мантиси M.

При поданні чисел у формі з плаваючою комою порядок P може бути додатним або від'ємним, але обов'язково цілим числом. Мантиса ж э додатним

або від'ємним правильним дробом, причому $k^{-1}<=M<1$. Це співвідношення називають умовою нормалізації мантиси. Нормалізоване подання мантиси дозволяє найбільш раціонально використовувати розрядну сітку (у старших, найбільш вагомих, розрядах мантиси не можуть знаходитися нульові цифри) і виключає неоднозначність запису всього числа. Знак мантиси визначає знак усього числа. Якщо для запису порядку використовується *m* розрядів, а для запису мантиси*n* розрядів, то у формі з плаваючою комою, може бути подане наступне максимальне (за абсолютною величиною) число

$$(1 - k^{-n})k^{km-1} = k^{km-1} - k^{km-n-1}$$

У свою чергу, мінімальне за абсолютною величиною, але не рівне нулю, число у формі з плаваючою комою , складає

$$(k^{-1})k^{-(km-1)} = k^{-km}$$

В ККС, де використовується подання чисел у форматі з плаваючою комою, арифметичні операції виконуються як над мантисами операндів, так і над їх порядками. При цьому часто необхідно виконувати операцію нормалізації чисел, сутність якої полягає у виконанні умови $k^{-1}<=M<1$.

Тому процесори ККС із плаваючою комою є більш складними і менш швидкодіючими, ніж процесори ККС із фіксованою комою (за однакової швидкодії електронних компонентів цих процесорів). Разом з тим, у ККС із фіксованою комою виникають деякі труднощі через необхідність масштабування даних. Крім того, тут значно вужчий діапазон чисел, які можна подавати.

4.2. Алгоритми додавання і віднімання чисел з фіксованою комою в прямих і доповняльних кодах

У пам'яті ККС операнди звичайно зберігаються у прямому або доповняльному коді. Використання прямих кодів часто забезпечує можливості реалізації або зручності виконання деяких інших операцій, наприклад, множення, з операндами у формі з фіксованою комою. В залежності від того, в яких кодах зберігають операнди в пам'яті, розрізняють операції в прямих і доповняльних кодах. Повний перелік варіантів виконання операцій додавання і віднімання чисел з фіксованою комою містить три позиції: ПК-ОК, ПК-ДК, ДК-ДК (ПК, ОК, ДК – відповідно прямий,обернений та доповняльний коди операндів).

Таблиця 15

Код для зберігання операндів у пам'яті	Код для віднімання операндів у процесорі
ПК	ОК
ПК	ДК
ДК	ДК

Таблиця 16

X_0	Y_0	S_0	Коди операндів, що передаються на суматор
0	0	0	$X_{ПК}$, $Y_{ПК}$
0	0	1	$X_{ПК}$, $Y_{ОК}$
0	1	0	$X_{ПК}$, $Y_{ОК}$
0	1	1	$X_{ПК}$, $Y_{ПК}$
1	0	0	$X_{ОК}$, $Y_{ПК}$
1	0	1	$X_{ОК}$, $Y_{ОК}$
1	1	0	$X_{ОК}$, $Y_{ОК}$
1	1	1	$X_{ОК}$, $Y_{ПК}$

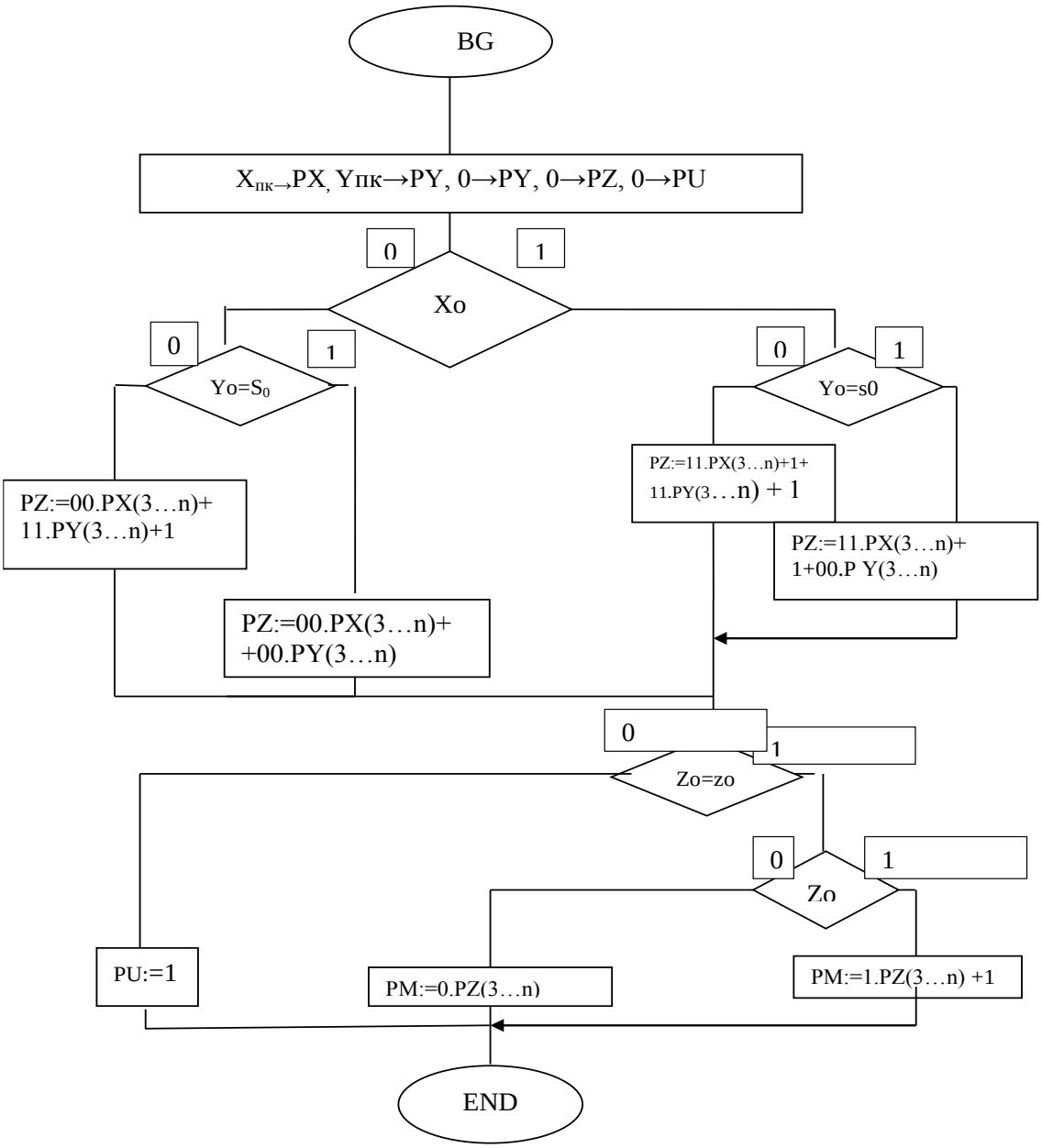


Рис. 28. ГСА додавання-віднімання чисел з фіксованою комою по варіанту ПК-ДК

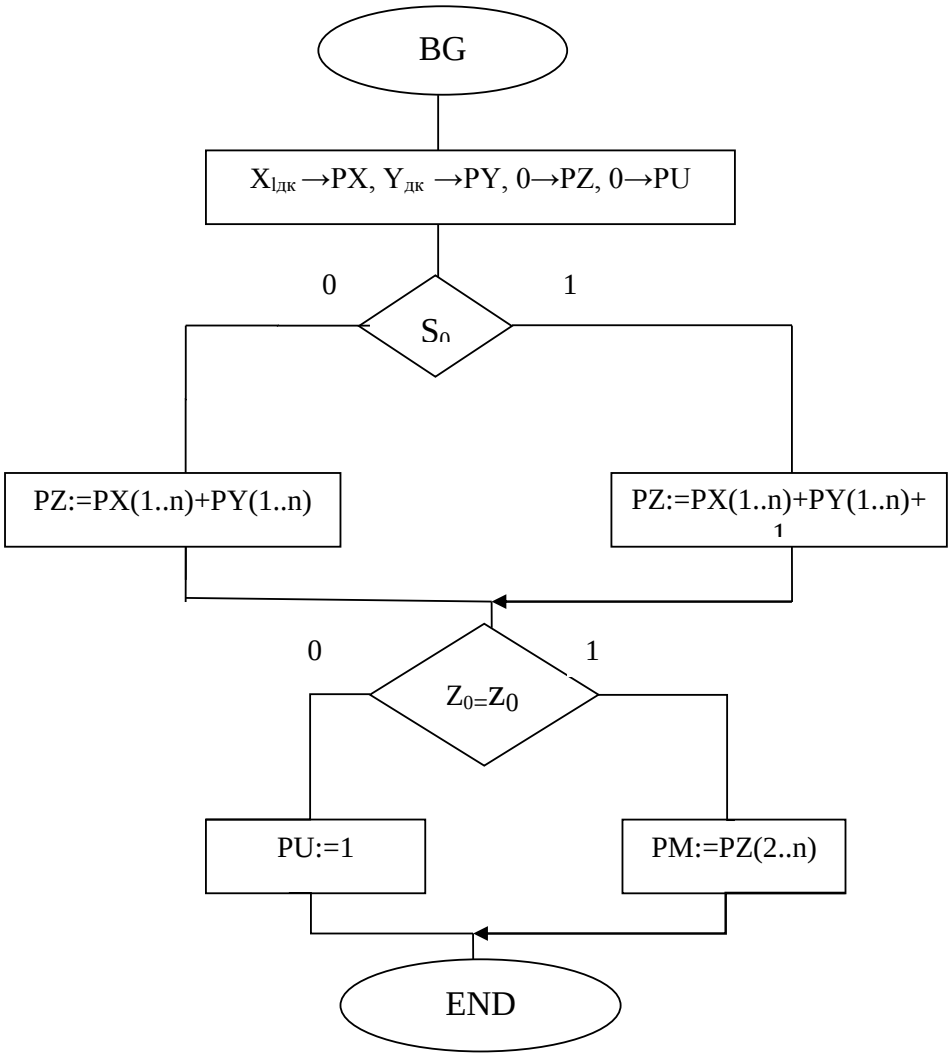


Рис. 29. ГСА додавання-віднімання чисел з фіксованою комою по варіанту ДК-ДК

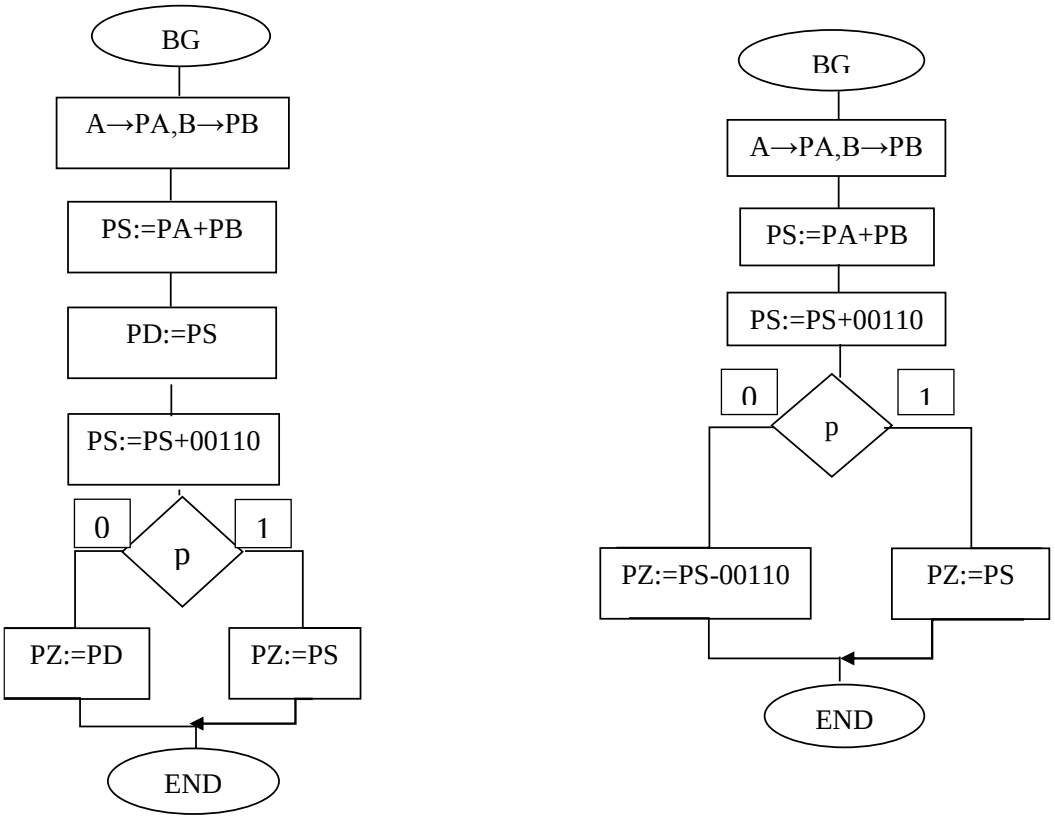


Рис. 30. ГСА додавання двійково-десяткових цифр A і B на лівому рисунку з дублюванням результату першого етапу в регістрі PD до того, як буде обчислене значення *p*, а на правому без дублювання попередньої суми

4.3. Алгоритми додавання і віднімання чисел з плаваючою комою

Для визначеності будемо вважати, що $k=2$. Тоді операнди X і Y у формі з плаваючою комою можна записати як $X = 2^A x$, $Y = 2^B y$, де A і B – порядки чисел X Y, що подані m розрядами, а мантисиxі x і y – n – розрядні правильні нормалізовані дроби, тобто, $1/2 \le |x| < 1$, $1/2 \le |y| < 1$.

Операція додавання-віднімання операндів у формі з плаваючою комою включає такі етапи.

1. Вирівнювання порядків операндів і визначення порядку результату.
2. Додавання-віднімання мантис.
3. Нормалізація результату.
4. Округлення результату.

Вирівнювання порядків необхідно для того, щоб цифри мантис з певною вагою знаходились у відповідних їм розрядах регістрів. Виконується така операція шляхом зсуву вправо мантиси того операнду, порядок якого менше. Алгоритми вирівнювання порядків зводиться до наступного (рис. 31).

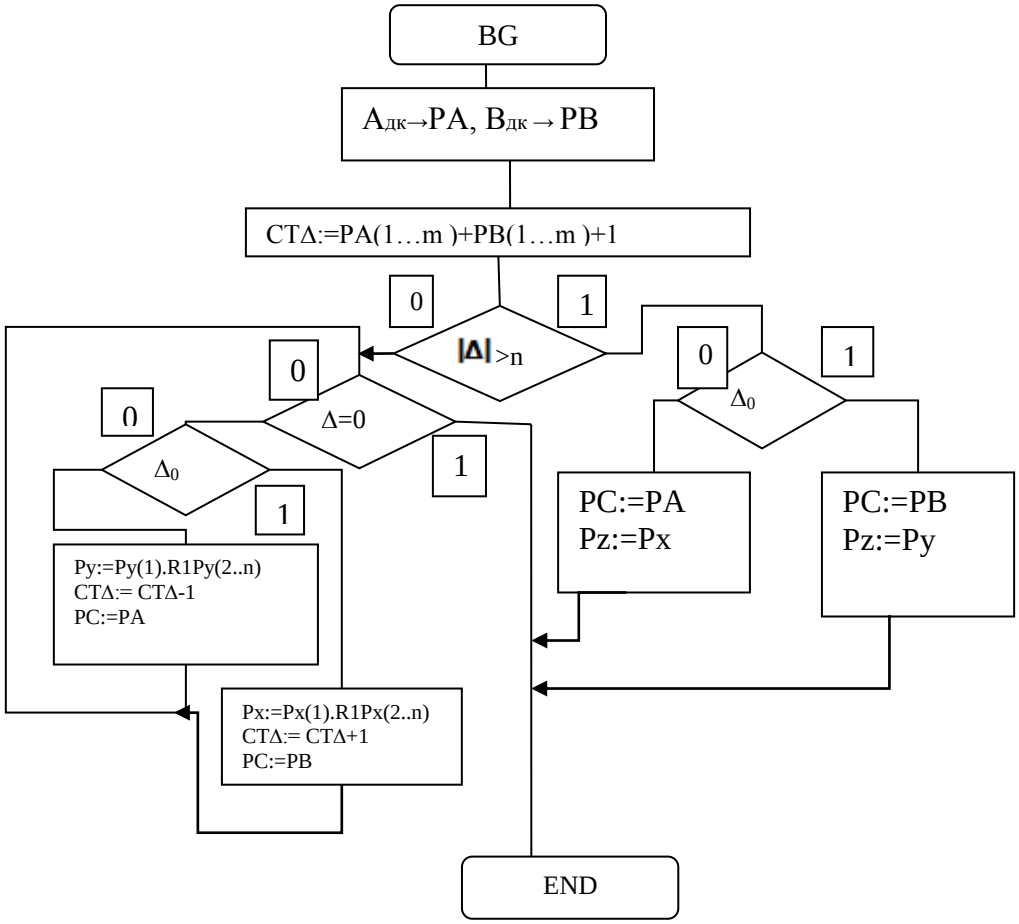


Рис. 31. ГСА додавання-віднімання чисел з плаваючою комою по алгоритму вирівнювання порядків

1. Сформувати різницю порядків $\Delta = A - B$.

2. Якщо $\Delta > 0$, то зсунути на один розряд вправо мантису y і відняти одиницю від різниці порядків Δ; якщо $\Delta < 0$, то зсунути на один розряд вправо мантиссу x і додати одиницю до різниці порядків

3. Пункт 2 алгоритму повторювати доти, поки виконується нерівність $\Delta \neq 0$; при $\Delta = 0$ вирівнювання порядків вважають закінченим.

При множенні і діленні чисел у формі з плаваючою комою над порядками виконуються тільки операції додавання і віднімання. Тому для подання порядків зручно використовувати доповняльний код. Оскільки порядки A і B можуть бути з різними знаками, то для подання їх різниці Δ може знадобитися $m + 1$ розрядів. До різниці порядків у процесі їх вирівнювання або додається, або віднімається одиниця. Тому виконувати таку операцію зручно на спеціальному лічильнику $ЛЧ$ різниці порядків Δ, а ідентифікатором цього лічильника надалі буде служити абревіатура $Лч\Delta$.

Абсолютна величина Δ може перевищувати число розрядів n, що

відводяться для подання мантис. У цьому випадку всі розряди мантиси,

зсуваються вправо, виходять за межі розрядної сітки і далі, при додаванні-відніманні мантис, абсолютна величина мантиси, що залишалася нерухомою, не змінюється. Отже, за умови $|\Delta|>n$ можна припинити вирівнювання порядків і присвоїти остаточному результату значення операнда, мантиса якого не зсувалася. ГСА такого алгоритму приведена на рис. 31, де *PC*-регістр порядку *C* результату операції, P_z – регістр мантиси результату *Z*.

Додавання-віднімання мантис, для подання яких звичайно використовується прямий код, виконується за тими ж алгоритмами, що і чисел у формі з фіксованою комою. Однак переповнення розрядної сітки (*U=1* на рис. 29) тут не приводить до зупинки обчислень, що як правило, має місце при виконанні операцій над числами у формі з фіксованою комою. Наявність переповнення враховується далі при нормалізації результату.

Визначити порядок результату зручно разом з вирівнюванням порядків операндів. Порядок результату дорівнює порядку більшого за абсолютною величиною операнда (тобто того, мантиса якого не зсувалася при вирівнюванні порядків). Дії, що відповідають визначенню порядку результату. Показані на графі алгоритму вирівнювання порядків (рис. 31).

При використанні модифікованого доповняльного коду для віднімання ліве порушення нормалізації виявляється так само, як і в попередньому випадку, а праве порушення – по комбінаціях 00,0... і 111,1... . ГСА нормалізації результату приведена на рис. 32, λ і ρ логічні умови, що відповідають перевірці наявності лівого і правого порушення нормалізації.

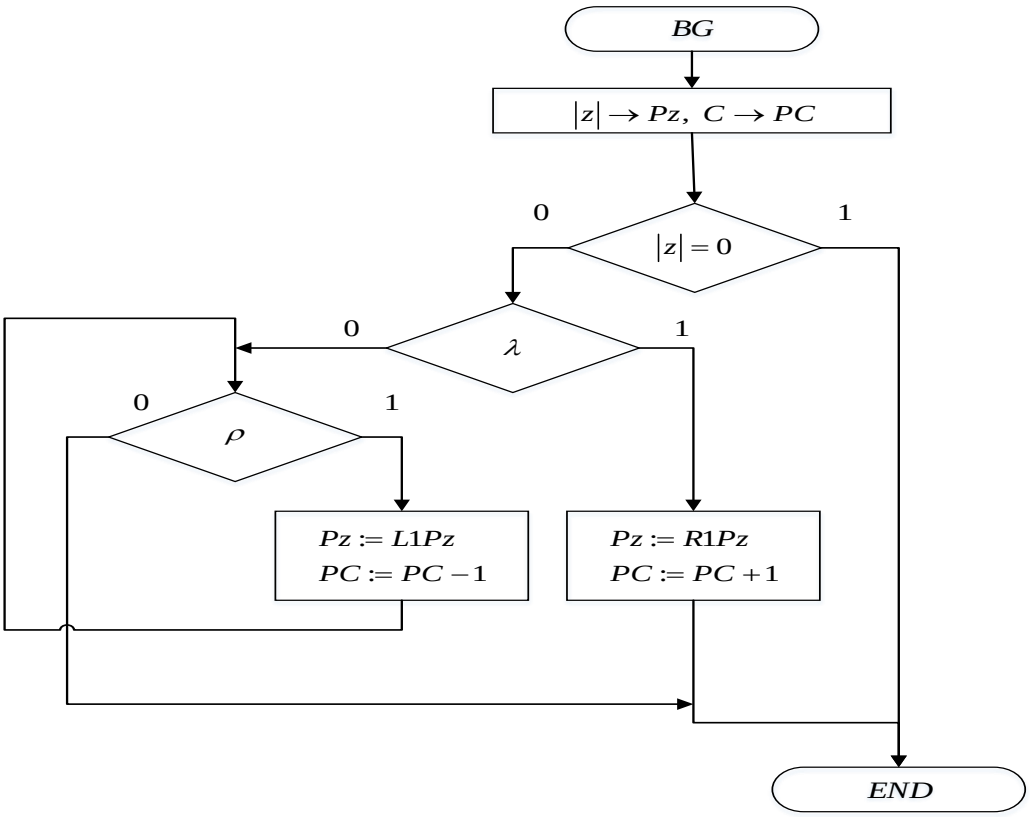


Рис. 32. Схема нормалізації результату для перевірки наявності лівого і правого порушення нормалізації

Як уже відзначалося раніше, необхідність округлення результату викликана тим, що при нормалізації результату може виконуватися правий зсув мантиси результату і молодший її розряд при цьому виходить за межі розрядної сітки. Якщо вважати, що результат поданий прямим кодом, то відкидання цього розряду вносить в абсолютну величину результату або нульову похибку (якщо відкидається цифра 0), або похибку, що дорівнює 1/2 ваги молодшого розряду (якщо відкидається цифра 1).

Для того, щоб похибка абсолютної величини результату була знакозмінною і її середнє значення в досить великій послідовності операцій дорівнювало нулю, округлення слід робити в такий спосіб. Якщо за межі розрядної сітки зсувається цифра 1, то молодший розряд мантиси результату встановлюють в одиницю незалежно від того, яка цифра була в цьому розряді раніше. В іншому випадку значення молодшого розряду не змінюють. Нехай x_n – молодша цифра мантиси результату до округлення, x'_n – та ж цифра після округлення, x_{n+1} – цифра мантиси, що виходить за межі розрядної сітки при округленні, δ- похибка округлення. Тоді всі можливі ситуації при округленні ілюструє табл. 17, з якої видно, що при такому округленні похибка абсолютної величини мантиси результату буде дорівнювати 0 або ±1/2 ваги молодшого розряду, а її середнє значення дорівнює нулю.

Таблиця 17

x_n	x_{n+1}	x'_n	δ
0	0	0	0
0	1	1	$+(1/2)\ 2^{-n}$
1	0	1	0
1	1	1	$-(1/2)\ 2^{-n}$

Викладені міркування щодо алгоритмів виконання операцій додавання-віднімання чисел у формі з плаваючою комою можуть бути легко узагальнені для будь-якої основи *k* системи числення, що використовується в ККС.

4.4. Алгоритми додавання-віднімання двійково-десяткових операндів

Для цього використаємо таблицю двійкових сум, їх зображень після першого етапу додавання, правильних їх зображень у ДДК “8,4,2,1”, десяткових переносів і коригуючих поправок (Таблиця 18). З таблиці видно, що якщо результат, отриманий на першому етапі, знаходиться в межах від 0 до 9, то коригувати його немає необхідності, а двійковий і десятковий перенос у цьому випадку співпадають.

Таблиця 18

Десят-кова сума	Сума після першого етапу					Правильна сума					Корекція				
	S_5	S_4	S_3	S_2	S_1	p	Z_4	Z_3	Z_2	Z_1	C_5	C_4	C_3	C_2	C_1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
2	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
3	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0
4	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	1	0	1	0	0	1	0	1	0	0	0	0	0
6	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0
7	0	0	1	1	1	0	0	1	1	1	0	0	0	0	0
8	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0
9	0	1	0	0	1	0	1	0	0	1	0	0	0	0	0
10	0	1	0	1	0	1	0	0	0	0	0	0	1	1	0
11	0	1	0	1	1	1	0	0	0	1	0	0	1	1	0
12	0	1	1	0	0	1	0	0	1	0	0	0	1	1	0
13	0	1	1	0	1	1	0	0	1	1	0	0	1	1	0
14	0	1	1	1	0	1	0	1	0	0	0	0	1	1	0
15	0	1	1	1	1	1	0	1	0	1	0	0	1	1	0
16	1	0	0	0	0	1	0	1	1	0	0	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1	0	0	1	1	0
18	1	0	0	1	0	1	1	0	0	0	0	0	1	1	0
19	1	0	0	1	1	1	1	0	0	1	0	0	1	1	0

Якщо ж результат першого етапу знаходиться в межах від 10 до 19, то для утворення правильної суми $pz_4z_3z_2z_1$, до попередньої суми $S_5S_4S_3S_2S_1$ необхідно додати коригувальну поправку 00110. Відмітимо, що ознакою знаходження суми першого етапу в діапазоні від 10 до 19 може служити наявність одиничного значення переносу *p*. Таким чином, якщо при додаванні корекції перенос *p = 0*, то за правильну суму необхідно взяти результат першого етапу, якщо ж *p = 1,* то за правильну суму необхідно взяти відкоригований результат. З урахуванням викладеного ГСА додавання двійково-десяткових цифр А і В можна подати рис. 33,а. Тут регістр *PD* необхідний для дублювання результату першого етапу до того, як буде визначене значення p.

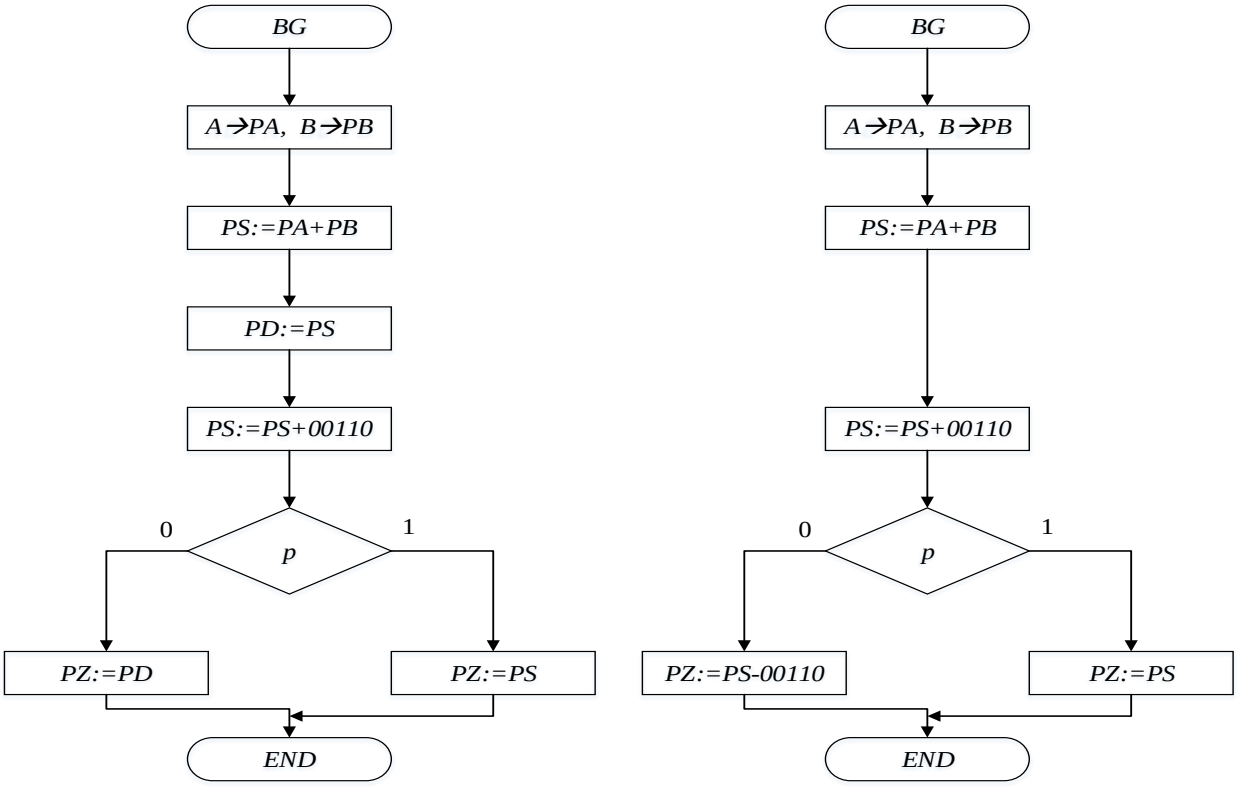


Рис. 33. Схема алгоритма додавання двійково-десяткових цифр

Очевидно, що одержати правильну суму в підсумку можна і без дублювання попередньої суми відповідно до ГСА на рис.33,б. Такий алгоритм містить на одну операторну вершину менше, ніж попередній, однак реалізація оператора відновлення правильного результату $(PZ := PS - 0110)$ може вимагати більших часових витрат, ніж оператора пересилання $(PD := PS)$.

Для додавання-віднімання багаторозрядних десяткових операндів дії, відповідно до алгоритмів на рис. 33, повинні виконуватися по кожному десятковому розряду (у ДДК "8,4,2, 1" − по кожній тетраді двійкових розрядів). Якщо використовуються ДДК, що не мають властивості доповняльності, то при передачі операндів з регістрів інверсним кодом (тобто тоді, коли на ГСА зазначено $\overline{PX}$ або $\overline{PY}$), повинне здійснюватися не просте інвертування двійкових цифр, а перетворення десяткової цифри *A* в цифру *9-A*.

4.5. Основні алгоритми комп'ютерного множення в прямому коді

При множенні чисел у прямих кодах у двійковій системі числення знакові і числові розряди обробляють окремо. Для визначення знаку добутку здійснюють додавання по модулю 2 цифр, записаних у знакових розрядах операндів. Далі будемо вважати, шо множник *X* і множене *У* – правильні двійкові дроби вигляду

$$X = \sum_{i=1}^{n} x_i 2^{-i} \quad \text{і} \quad Y = \sum_{i=1}^{n} y_i 2^{-i}$$

де $x_i \in \{0,1\}$. Тоді добуток абсолютних величин операндів *X* і *Y* можна подати як

$$+x_3 y_1 2^{-4} + \cdots + x_3 y_{n-2} 2^{-n-1} + x_3 y_{n-1} 2^{-n-2} + x_3 y_n 2^{-n-3}$$

$$+ x_n y_1 2^{-n-1} + x_n y_2 2^{-n-2} + x_n y_3 2^{-n-3} + \cdots + x_n y_n 2^{-2n} =$$

Вирази виду Уx_i, називають *частковими добутками*, а вирази виду $x_i y_i$ -

розрядними добутками. Оскільки множення на еквівалентне зсуву вправо, то обчислення добутку *Z* зводиться до формування часткових добутків Yx_i, їхнього зсуву і додаванню з урахуванням ваг, обумовлених величинами

. У свою чергу

$$Yx_i = \begin{cases} Y, \text{якщо } x_i = 1 \\ 0, \text{якщо } x_i = 0 \end{cases}$$

Очевидно, що таку операцію формування часткового добутку Yx_i, можна легко реалізувати як видачу вмісту регістра, що зберігає *Y*, керовану цифрою множника x_i. З вищенаведеного виразу для добутку *Z* також випливає, що розрядні добутки

$$x_i y_j = \begin{cases} y_j, \text{якщо } x_i = 1 \\ 0, \text{якщо } x_i = 0 \end{cases}$$

мають ваги від до , а ненульова цифра добутку з вагою може утворитися як наслідок переносу при додаванні розрядних добутків з меншими вагами. Це значить, що для запису всіх розрядів добутку їх кількість повинна дорівнювати 2n.

Таким чином, множення Y на X може бути реалізоване шляхом виконання певного циклічного процесу, де кожен крок складається з формування чергового часткового добутку, його зсуву відносно вже накопиченої суми часткових добутків і додавання до цієї суми. Правила виконання цих дій залежать від конкретної форми для виразу $Z = X * Y$. Відомо чотири основних алгоритми комп'ютерного множення.

1. **Перший алгоритм множення** використовує подання добутку $Z = X * Y$ у вигляді

$$Z = X * Y = x_n 2^{-n} Y + x_{n-1} 2^{-n+1} Y + \cdots + x_2 2^{-2} Y + x_1 2^{-1} Y =$$

Звідси випливає, що добуток *Z* може бути отриманий за рекурентною формулою

$$Z_i = (Z_{i-1} + x_{n-i+1} Y) 2^{-1}, \qquad i = \overline{1,n}$$

де $Z_0 = 0$, $Z_n = Z$, Z_i – сума *i* часткових добутків. Множення тут починається з молодших розрядів множника, на кожному кроці сума часткових добутків зсувається вправо, число кроків множення дорівнює n, останній крок закінчується зсувом. ГСА такого множення подана на рис. 34, де x_n^*.

- цифра в молодшому розряді регістра множника *РХ* на даному кроці множення ("поточна" цифра множника), *ЛЧК* – лічильник числа кроків множення. Довжина регістрів операндів складає *n* розрядів, регістра результату – *2n* розрядів, лічильника кроків – $]\log_2 n[$ розрядів. Додавання останнього часткового добутку може супроводжуватися формуванням переносу зі старшого розряду суми часткових добутків, який записується в регістр *РZ* в

процесі виконання наступного (останнього) зсуву.

Рекомендується виконати практичне завдання 1.

Складемо цифрову діаграму множення чисел *X=11/16* та *У-9/16, n = 4* (табл. 19). У табл. 19 кожний рядок, за винятком тих, які позначені як “Результат додавання”, відповідає виконанню мікрооперацій, зазначених в операторних вершинах ГСА. При цьому порядок їх виконання визначається ГСА і конкретними значеннями операндів.

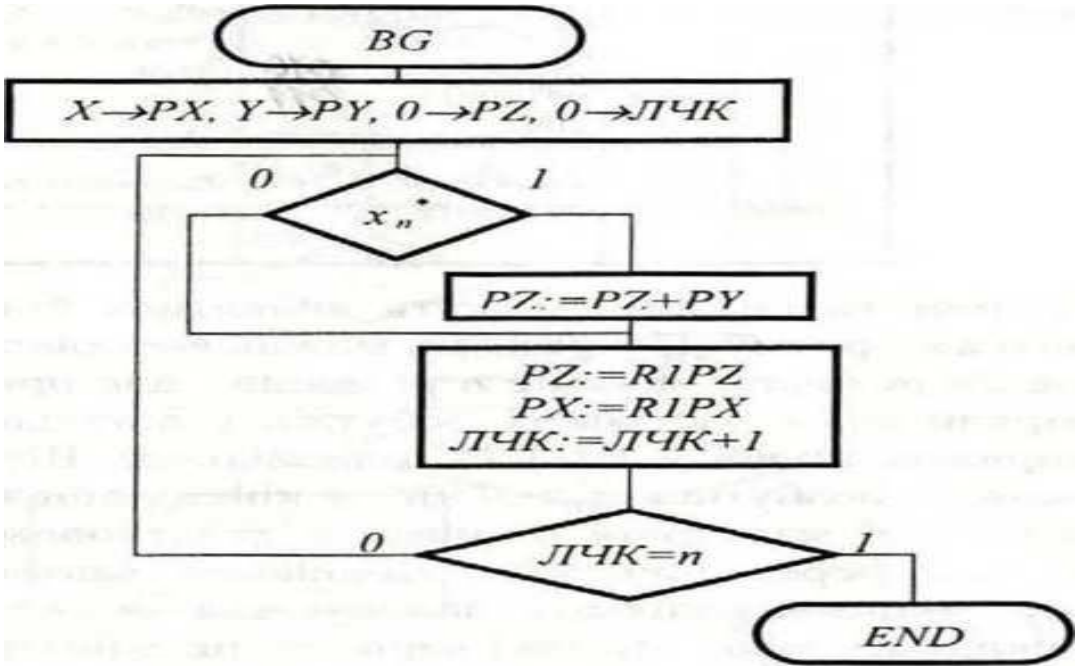


Рис. 34. ГСА першого алгоритму множення: множення з молодших розрядів множника зі зсувом суми часткових добутків вправо

Таблиця 19

$PX\ x_n^*$	PY	PZ	ЛЧК	Пояснення
1011	1001	00000000 1001	000	Вихідний стан +Y
0101		10010000 01001000 1001	001	Результат додавання Зсув +Y
0010 0001		11011000 01101100 00110110 1001	100 101	Результат додавання Зсув Зсув +Y
0000		11000110 01100011	100 END	Результат додавання Зсув

З практичної точки зору можуть виявитися більш зручними інші варіанти цього алгоритму. Наприклад, можна використовувати регістр *PZ* довжиною не *2n* розрядів, а лише *n* розрядів, але при цьому записувати молодші розряди суми часткових добутків у вивільнювані розряди *PX,* тому що напрямок зсувів у *PX* і *PZ* однаковий. Ще в одному варіанті алгоритму можна позбутися від *ЛЧК,* а кінець множення визначати за появою спеціальної маркерної одиниці в додатковому *2n+1-му* розряді регістра *PZ,* яка перед початком виконання множення (одночасно з підготовчими мікрооперaціями) записується в n+1-й розряд цього регістра. Очевидно, однак, що такі варіанти не змінюють суті алгоритму, що зафіксована в його назві, а саме: множення з молодших розрядів множника зі зсувом суми часткових добутків вправо.

Час множення по цьому алгоритму буде складати $T_{M_1} = [n(t]_{зс} + t_+)$, де час зсуву, t_+ – час додавання.

Другий основний алгоритм множення базується на поданні добутку у вигляді

$$Z = X * Y = x_n 2^{-n} Y + x_{n-1} 2^{-n+1} Y + \cdots + x_2 2^{-2} Y + x_1 2^{-1} Y =$$
$$= \left(\ldots\left((0 + x_n Y 2^{-n}) + x_{n-1} Y 2^{-n+1}\right) + \cdots + x_2 Y 2^{-2}\right) + x_1 Y 2^{-1} =$$

,

де $Y_i = Y 2^{-i}$ Множення, таким чином, зводиться до обчислень за рекурентною формулою

$$Z_i = Z_{i-1} + x_{n-i+1} Y_{n-i+1}\ , \qquad i = \overline{1,n}$$

де $Z_0 = 0, Z_n = Z, \quad Y_n = Y2^{-n}, \quad Y_{n-i+1} = 2Y_{n-(i+1)+1}$. Множення тут починається з молодших розрядів множника, у процесі множення множене зсувається вліво, число кроків дорівнює *n*, а закінчується множення додаванням. Регістр множника повинен мати довжину в *n* розрядів, регістри множеного і суми часткових добутків – по 2 *n* розрядів. Перед початком множення множене повинне бути записане у відповідний регістр зі зсувом вправо на *n* розрядів для того, щоб було сформоване значення Y_n. ГСА такого множення показана на рис. 35.

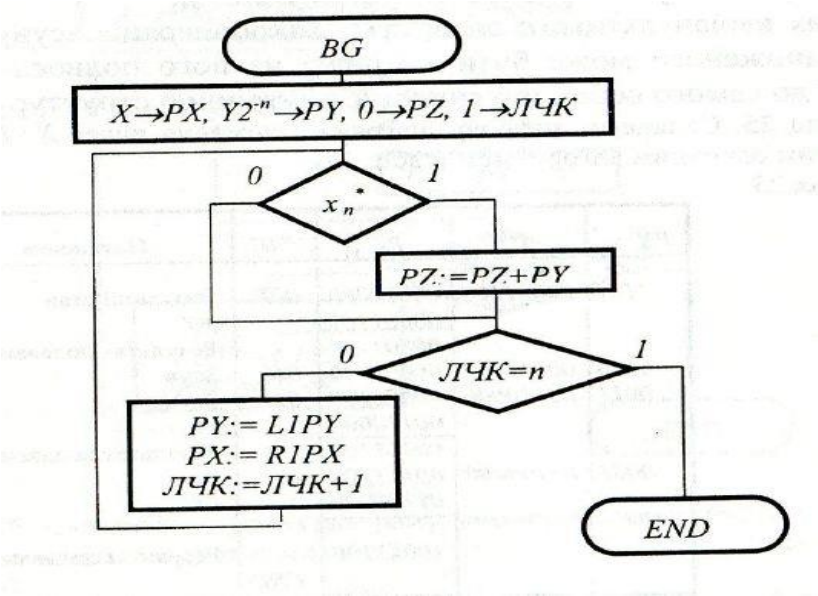


Рис. 35. ГСА другого основного алгоритма множення з молодших розрядів множника з подальшим зсувом вліво множеного на ***n*** кроків

Початкова установка *ЛЧК* в одиницю і викликана тим, шо значення Y_n уже сформоване при запису його в регістр множеного.

Час множення за другим основним алгоритмом буде визначатися тривалістю *n* додавань і *n-1* зсувів,

тобто, . Зменшення числа зсувів тут на одиницю в порівнянні з T_{M1} пояснюється тим, що перший зсув фактично виконується при початковому заповненні регістрів.

У варіантах розглянутого алгоритму для визначення кінця операції може служити, наприклад, не заповнений до *n* лічильник *ЛЧК,* а нульовий вміст регістра *РХ.* Справді, якщо починаючи з деякого x_i виявиться, що $x_i = x_{i-1} = \cdots = x_2 = x_1 = 0$, то, на відміну від першого алгоритму, операцію можна закінчувати.

Рекомендується виконати практичне завдання 2.

Складемо цифрову діаграму множення чисел *X=11/16* та

У-9/16, n = 4 (табл. 19). У табл. 19 кожний рядок, за винятком тих, які позначені як “Результат додавання”, відповідає виконанню мікрооперацій, зазначених в операторних вершинах ГСА. При цьому порядок їх виконання визначається ГСА і конкретними значеннями операндів.

Третій основний алгоритм множення одержимо, коли напишемо добуток Z у вигляді:

$$Z = X * Y = x_1 2^{-1} Y + x_2 2^{-2} Y + \cdots + x_{n-1} 2^{-n+1} Y + x_n 2^{-1-n} Y =$$

Отже, добуток можна отримати за рекурентною формулою:

$Z_i = Z_{i-1} 2 + x_i Y 2^{-n}, i = \overline{1, n}$

де $Z_0 = 0, \quad Z_n = Z$. Відповідно до цієї формули множення починається зі старших розрядів множника, зсувається вліво сума часткових добутків, число кроків множення дорівнює n, закінчується виконання алгоритму додаванням. ГСА для цього випадку приведена на рис. 36. Довжину в 2n розрядів тут повинен мати тільки регістр PZ. Однак оскільки зсуви в PX і PZ виконуються в одну і ту ж сторону, то в варіантах цього алгоритму старші розряди добутку можна поміщати в регістр PX. Час множення за третім основним алгоритмом складає $T_{M3} = (n-1)(t_C + t_+) + t_+$, тобто, такий же. як і за другим основним алгоритмом.

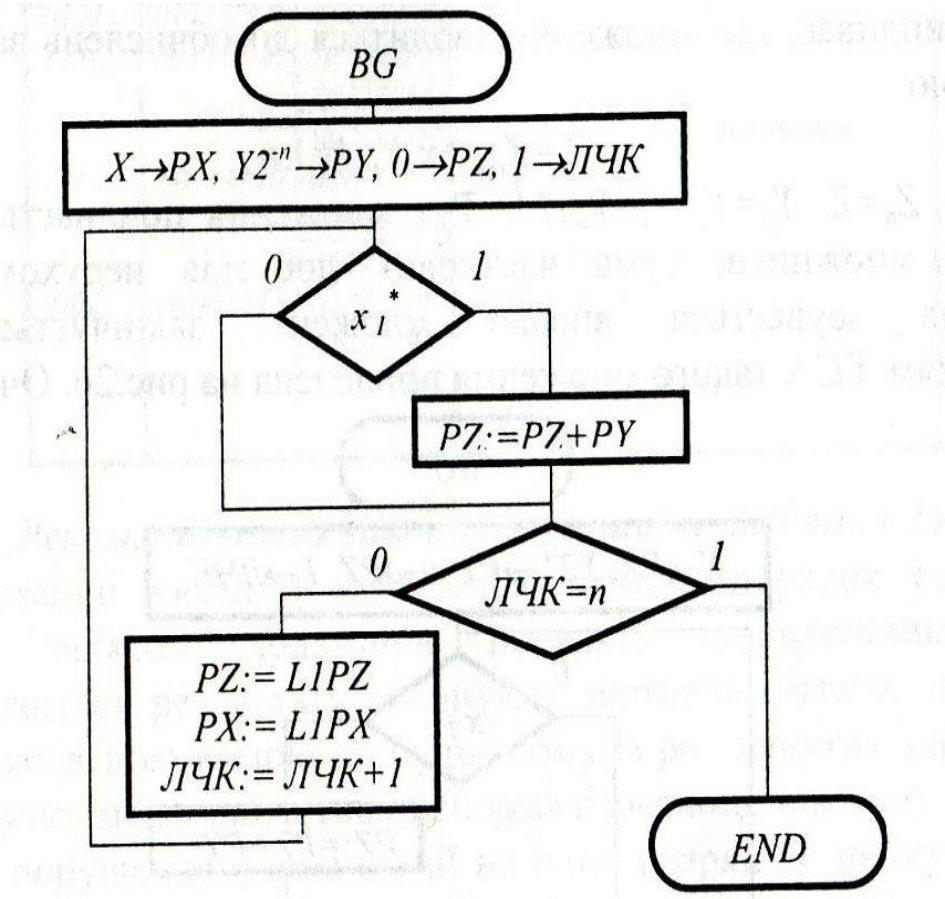


Рис. 36. ГСА третього основного алгоритма множення з початком зі старших розрядів множника, зсувом вліво суми часткових добутків з числом кроків множення ***n***

Рекомендується виконати практичне завдання 3

Складемо цифрову діаграму множення чисел X = 11/16 та Y = 10/16, n = 4 за третім основним алгоритмом (табл. 20).

Таблиця 20

x_i^* РХ	РУ	РZ	ЛЧК	Пояснення
1011	1010	00000000	001	Вихідний стан
		1010		+Y
		00001010		Результат додавання
0110	1010	00010100	010	Зсув
1100	1010	00101000	011	Зсув
		1010		+Y
		00110010		Результат додавання
1000	1010	01100100	100	Зсув
		1010		+Y
		01101110		Результат додавання
			END	

Основні ідеї інших можливих варіантних відмінностей третього алгоритму множення по суті повторюють вже розглянуті раніше.

Четвертий основний алгоритм множення використовує подання добутку у вигляді

$$Z = X * Y = x_1 2^{-1} Y + x_2 2^{-2} Y + \cdots + x_{n-1} 2^{-n+1} Y + x_n 2^{-1-n} Y =$$

Звідси випливає, що множення зводиться до обчислень за рекурентною формулою

$$Z_i = Z_{i-1} + x_i Y_i, \qquad i = \overline{1,n},$$

де $Z_0 = 0, \qquad Z_n = Z$, $Y_0 = Y, \qquad Y_i = Y_{i-1} 2^{-1}$. Тут множення починається зі старших розрядів множника, сума часткових добутків нерухома, у процесі множення зсувається вправо множене, закінчується множення додаванням. ГСА такого множення приведена на рис. 37.

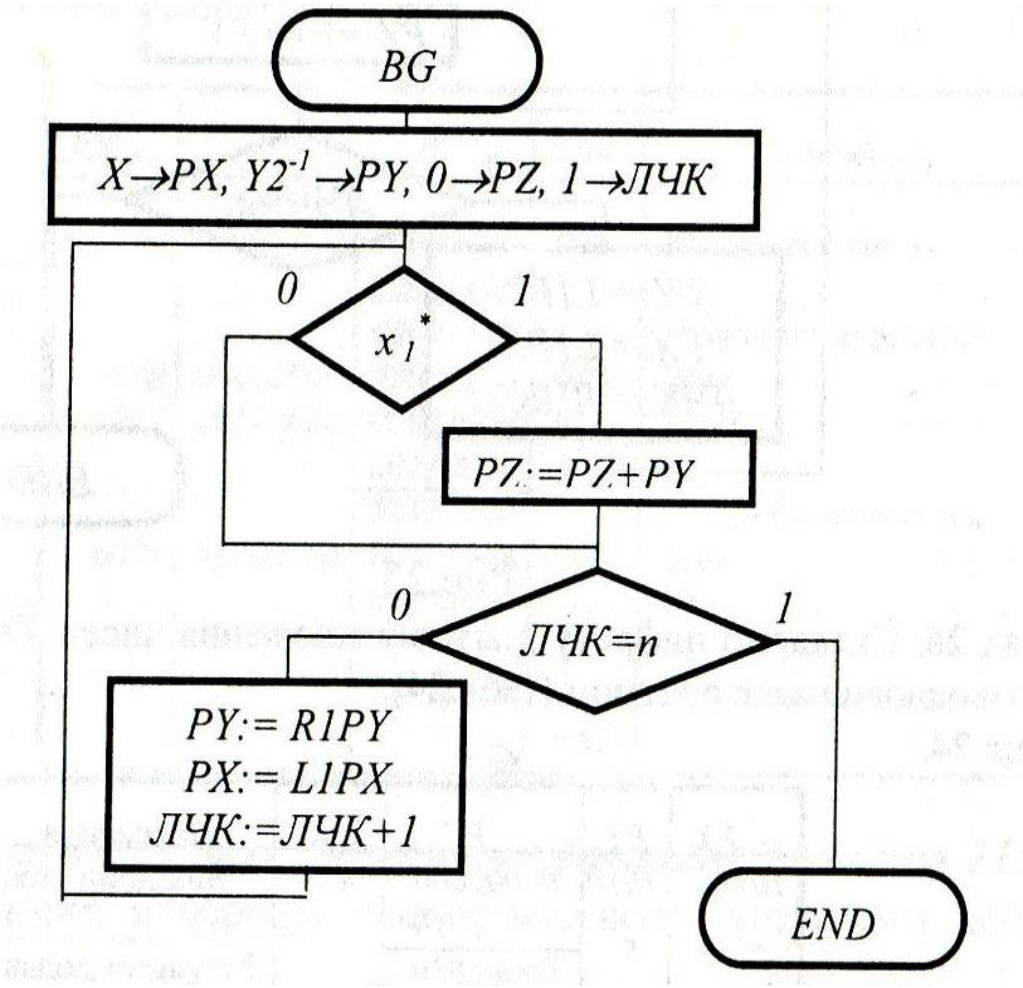


Рис. 37. ГСА множення за четвертим основним алгоритмом

Очевидно, що довжину в 2*n* розрядів тут повинні мати регістри РУ і PZ. Оскільки сума часткових добутків нерухома, то використовувати для збереження її частини вивільнюваний регістр РХ неможливо. Однак закінчення операції можна визначати по нульовому вмісту РХ і, таким чином, виключити ЛЧК. Відмінності на ГСА, як і при множенні за другим основним алгоритмом, будуть полягати в заміні умови ЛЧК = *n* умовою РХ = 0 і виключенні елементарних операцій на ЛЧК.

Час множення за четвертим основним алгоритмом складає

Рекомендується виконати практичне завдання 4.

Складемо цифрову діаграму множення чисел X = 7/16 та У = 15/16, n-4 за четвертим основним алгоритмом (табл. 21).

Таблиця 21

x_1^* РХ	РУ	РZ	ЛЧК	Пояснення
0111	01111000	00000000	001	Вихідний стан
1110	00111100	00000000	010	Зсув
		00111100		+Y
		00111100		Результат додавання
1100	00011110	00111100	011	Зсув
		00011110		+Y
		01011010		Результат додавання
1000	00001111	01011010	100	Зсув
		00001111		+Y
		01101001		Результат додавання
			END	

При множенні чисел з плаваючою комою порядок результату визначається шляхом додавання порядків співмножників. Для визначення мантиси результату виконують множення мантис операндів як чисел з фіксованою комою по будь-якому із розглянутих алгоритмів. Мантиса добутку може виявитися ненормалізованою, причому можливе тільки праве порушення нормалізації на один розряд. У цьому випадку мантису треба подвоїти шляхом зсуву її на один розряд вліво, а з порядку добутку треба відняти одиницю.

4. 6. Комп'ютерне множення з округленням

Розглянуті вище чотири основних алгоритми множення забезпечують одержання *2n* розрядів добутку. Однак на практиці часто немає необхідності в такій їх кількості. Наприклад, якщо операнди подані у формі з фіксованою перед старшим розрядом комою (тобто, коли X<1, Y<1), то вся частина добутку, задана розрядами з *n+1-го* по *2n-й*, буде менше самої молодшої одиниці кожного з операндів, і тому добуток доцільно округляти до *n* розрядів.

Округлення відкиданням *n* молодших розрядів добутку може приводити до від’ємної похибки, абсолютна величина якої знаходиться в межах від 0 до $[-(2]^{\intercal}(-n-1)-2^{\intercal}(-2n-1)$, а її середнє значення дорівнює $[-(2]^{\intercal}(-n-1)-2^{\intercal}(-2n-1)$. Якщо операція множення буде чергуватися з іншими операціями, наприклад, додаванням-відніманням, то може відбуватися накопичення цієї похибки, що істотно спотворює результат послідовності операцій. Тому з метою зменшення такої похибки округлення виконують у такий спосіб: у *2n*-розрядному добутку після закінчення операції додають

одиницю в *n+1*-й розряд (при цьому може відбутися зміна цифр у лівій частині добутку) і далі відкидають усі молодші розряди, що правіші *n*-го.

Таблиця 22

Ваги розрядів							Похибка округлення δ
2^{-n}	2^{-n-1}	2^{-n-2}	2^{-n-3}		2^{-2n+1}	2^{-2n}	
Молодші розряди добутку							
z'_n	$z^{\square}_{n+1}$	$z^{\square}_{n+2}$	$z^{\square}_{n+3}$		$z^{\square}_{2n-1}$	$z^{\square}_{2n}$	
$z^{\square}_n$		0	0		0	0	0
$z^{\square}_n$		0	0		0	1	-2^{-2n}
$z^{\square}_n$		0	0		1	0	-2^{-2n+1}
$z^{\square}_n$		0	0		1	1	$-2^{-2n+1}-2^{-2n}$
$z^{\square}_n$		1	1		1	0	$-2^{-n-1}-2^{-2n+1}$
$z^{\square}_n+1$		1	1		1	1	$-2^{-n-1}+2^{-2n}$
$z^{\square}_n+1$		0	0		0	0	$+2^{-n-1}$
		0	0		0	1	$+2^{-n-1}-2^{-2n}$
$z^{\square}_n+1$		1	1		0	0	$+2^{-2n+2}$
$z^{\square}_n+1$		1	1		0	1	$+2^{-2n+1}+2^{-2n}$
$z^{\square}_n+1$		1	1		1	0	$+2^{-2n+1}$
$z^{\square}_n+1$		1	1		1	1	$+2^{-2n}$

Позначимо як z'_n цифру *n*-го розряду після округлення, а δ -похибку такого округлення. Тоді можливі значення z'_n і δ можна подати табл. 22, що дає можливість оцінити максимальну і середню похибки такого округлення. Як видно з табл. 22, для розглянутого алгоритму округлення похибка буде знакозмінною. При цьому її максимальне за абсолютною величиною значення

$$2^{-n-1},$$

не перевищує тобто, половини ваги молодшого (того, що залишається в розрядній сітці після округлення) розряду. Середнє ж значення похибки δ знайдемо, якщо відмітимо, що сума похибок у першому й останньому рядках таблиці, у другому і передостанньому рядках і т.д., а також

сума похибок у двох середніх рядках таблиці дорівнює . Оскільки

2^n

число рядків у таблиці дорівнює , то сума похибок по всіх рядках складе

. Таким чином, середня похибка округлення буде дорівнювати:

$$\delta = \frac{2^{-n-1}}{2^n} = 2^{-2n-1}$$

тобто, середня похибка округлення дорівнює половині ваги останнього розряду *2n*-розрядного добутку. Очевидно, що такі значення максимальної і середньої похибки округлення часто можуть бути цілком прийнятними з практичної точки зору. При цьому додаткового кроку для саме округлення виконувати не треба – досить записати одиницю в той розряд PZ, який після закінчення операції виявиться старшим розрядом, що відкидається. Однак, розглянутий алгоритм округлення вимагає використання *2n-1* розрядного регістра PZ і суматора такої ж розрядності, незважаючи на те, що після завершення операції *n* молодших розрядів добутку відкидаються.

Зменшити довжину PZ і суматора при множенні можна за рахунок використання округлення в процесі додавання часткових добутків. У цьому випадку на кожному кроці множення зберігають лише *l* молодших розрядів суми часткових добутків. Величина *l* визначається з умови того, щоб похибка округлення не перевищувала половини ваги *n*-го розряду добутку після округлення. Будемо вважати, що довжина регістра PZ складає *n+1* розрядів. Оцінимо максимальну похибку, обумовлену відкиданням *n-l* молодших розрядів добутку, припускаючи, що всі розрядні добутки дорівнюють *l*. Для цього подамо суму часткових добутків у вигляді табл. 23 (*n* = *6*).

Таблиця 23

Ваги розрядів											
2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}
	1	1	1	1	1	1					
		1	1	1	1	1	1				
			1	1	1	1	1	1			
				1	1	1	1	1	1		
					1	1	1	1	1	1	
						1	1	1	1	1	1
n						*l*		*n-l*			

Очевидно, що похибка від втрати *n-1* молодших розрядів буде дорівнювати сумі одиниць (з урахуванням їх ваг), що знаходяться правіше *n+l*-го розряду. Тоді можна записати, що

$$\Delta = 2^{-(n+l+1)} +$$
$$+ 2^{-(n+l+1)} + 2^{-(n+l+2)} +$$
$$+ 2^{-(n+l+1)} + 2^{-(n+l+2)} + 2^{-(n+l+3)} +$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$+2^{-(n+l+1)} + 2^{-(n+l+2)} + 2^{-(n+l+3)} + \ldots\ldots + 2^{-(2n-l)} + 2^{-2n} =$$
$$= 2^{-(n+l)} (2^{-1} +$$
$$+ 2^{-1} + 2^{-2} +$$
$$+ 2^{-1} + 2^{-2} + 2^{-3} +$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$2^{-1} + 2^{-2} + 2^{-3} + \ldots\ldots\ldots\ldots 2^{-n+l+1} + 2^{-n+1}) =$$
$$=2^{-(n+l)} ((1-2^{-1}) +$$
$$+(1-2^{-2}) +$$
$$+(1-2^{-3}) +$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$+(1-2^{-n+l}))= 2^{-(n+l)} (n - l-(1-2^{-n+l}))\approx 2^{-(n+l)} (n - l-1).$$

Поставимо вимогу, щоб ця похибка була не більшою половини ваги молодшого розряду, що залишається, тобто

$$2^{-(n+l)} (n - l-1) \le 2^{-n-1}.$$

Звідки одержимо

$$2^{-(l-1)} (n - l-1) \le 1,$$

$$-l + 1 + \qquad \le 0,$$

$$-l \le -\log_2(n - l - 1) - 1,$$

$$l \ge 1 + \log_2(n - l - 1).$$

З останньої умови встановимо наступну систему відповідностей між числом додаткових розрядів і довжиною операндів:

$l = 2$ для $4 \le n \le 5$,
$l = 3$ для $6 \le n \le 8$,
$l = 4$ для $9 \le n \le 13$,
$l = 5$ для $14 \le n \le 22$,
$l = 6$ для $23 \le n \le 39$,
$l = 7$ для $40 \le n \le 72$.

Зауважимо, що другий та четвертий основні алгоритми множення можуть бути пристосовані для таких обчислень

$$\sum_{i=1}^{N} XiYj,$$

для чого достатньо черговий результат не відсилати в пам'ять, а залишати його на PZ , який в цьому випадку повинен мати додаткові старші розряди, які призначені для зберігання цілої частини накопичуваної суми добутків.

4.7. Основні алгоритми прискоренного комп'ютерного множення чисел у комп'ютерах і комп'ютерних системах.

Як показує аналіз часу виконання окремих операцій у ККС. Часові витрати на множення можуть складати до 80% усього машинного часу при розв'язку деяких класів задач. Тому розробка алгоритмів прискореного, які забезпечують час множення менше тривалості ***n*** кроків додавання і зсуву, має дуже важливе значення. Хоча до цього часу розроблено велику кількість таких алгоритмів, однак Їх основу складають деякі базові способи або підходи.

Способи прискорення множення розділяють на ***логічні, апаратні*** і ***комбіновані.***

Логічними називають такі способи, де ефект прискорення досягається за рахунок ускладнення засобів керування процесом множення без яких-небудь істотних змін у структурі арифметичних засобів. Додаткові апаратурні витрати, обумовлені таким ускладненням. Не залежать від розрядності операндів.

Апаратними називають такі способи, що забезпечують прискорення множення за рахунок введення додаткового обладнання в основні арифметичні засоби, завдяки чому досягається поєднання в часі окремих складових частин процесу множення. При цьому залежно від того, якій величині ***n*** чи n^2 пропорційні додаткові витрати обладнання, розрізняють ***апаратні способи першого*** і ***другого порядку***.

Комбіновані способи використовують як апаратні,так і логічні засоби прискорення множення. Найпростішим логічним способом прискорення множення є ***пропуск кроків додавання*** в тих випадках, коли ***чергова цифра множника дорівнює нулю***. Відзначимо, що завдяки своїй простоті та очевидності ця ідея у неявному вигляді реалізована у всіх розглянутих раніше алгоритмах.

Цей спосіб дозволяє в середньому скоротити число додавань, шо припадають на один розряд множника, до 0,5 (тут і далі передбачається, що 0 і

1 зустрічаються в будь-якому розряді множника однаково часто). Таким чином, середній час множення з пропуском кроків додавання з нулем складе:

.

Більш ефективним є логічний спосіб прискорення множення з використанням операцій *додавання* і *віднімання*, що називають також способом *послідовного перетворення цифр множника*. У цьому випадку множник записують у двійковій модифікованій квазіканонічній системі числення з цифрами {-1, 0, 1} і з можливих подань множника вибирають ті, які містять мінімальне число ненульових цифр. Для цього групу $m \geq 2$ одиниць вигляду 0 1 1…1 0
m

перетворюють у групу вигляду .
M

Множене додають до суми часткових добутків, якщо чергова цифра перетвореного множника дорівнює 1, і віднімають, якщо цифра перетвореного множника дорівнює $\bar{1}$.

Розглянемо цей спосіб докладніше на прикладі першого основного алгоритму множення. Множник перетворюють у цьому випадку молодших розрядів, причому перетворений множник може мати на одну цифру більше, ніж вихідний операнд, що збільшує число кроків алгоритму на одиницю. Однак на останньому кроці зсув виконувати не треба, ГСА такого множення пояснюється рис.38, де буквою φ_1, позначена ознака того, що на *i*-му кроці виконувалося перетворення цифр множника. Якщо $\varphi_i = 0$, а x_{n-1} і x_n утворюють комбінації 00,10 і 01, то множення виконується як звичайно – за першим основним алгоритмом. У випадку $x_{n-1} = x_n = 1$ виконується віднімання множеного і виробляється ознака $\varphi_{i+1} = 1$, що свідчить про те, що перша одиниця даної групи перетворена в 1.

Множене додають до суми часткових добутків, якщо чергова цифра перетвореного множника дорівнює 1, і віднімають, якщо цифра перетвореного

множника дорівнює . У цьому випадку всі інші одиниці групи повинні бути перетворені в нулі, а перший нуль за даною групою одиниць слід замінити на

одиницю. Якщо ж $\varphi_i = 1$. то виконувана дія визначається цифрами множника x_n^* і x_{n-1}^*, отриманими в такий спосіб

$$x_n^* = x_n + \varphi_i (mod\ 2),\ x_{n-1}^* = x_{n-1} + p_n (mod\ 2),$$

де p_n – перенос при двійковому додаванні x_n і φ_i. Ознака $\varphi_{i+1} = 1$ при цьому виробляється не тільки у випадку $x_n^* = x_{n-1}^* = 1$, але й у випадку $p_n = 1$.

Можна показати, що при досить великому *n* середнє число додавань-віднімань, віднесене до одної цифри множника, складає 1/3, тобто $T_{MCP} = (n+1)(t_c + 0{,}3t_+)$.

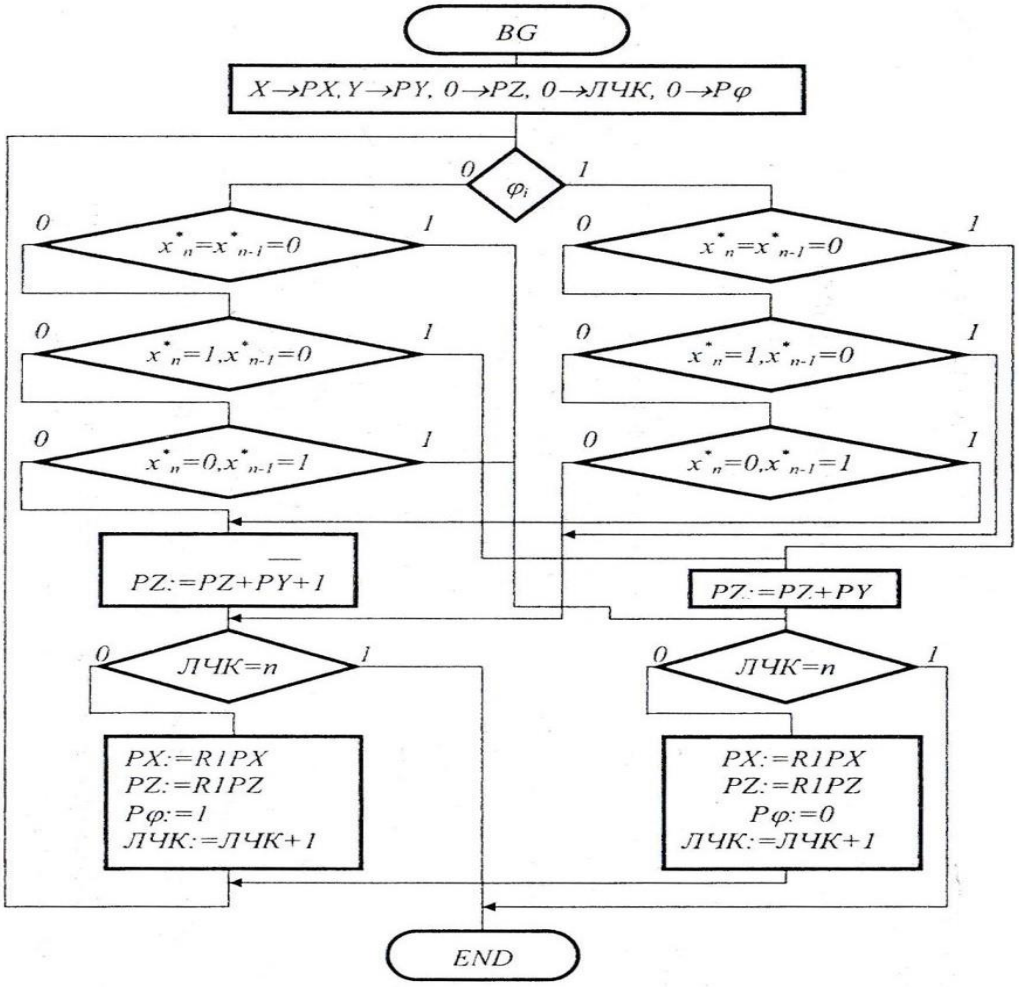


Рис. 38. ГСА логічного способу прискорення множення з використанням операцій додавання і віднімання, тобто способом послідовного перетворення цифр множника.

Рекомендується виконати практичне завдання 5.

Складемо цифрову діаграму множення чисел X = 55/64 і Y = 37/64 за алгоритмом послідовного перетворення цифр множника (табл. 24). У двійковому поданні X = 0,110111, Y = 0,100101, а z^*_0 відповідає вмісту двох знакових розрядів PZ. Як видно з табл.24, неперетворений множник містив 5 одиниць і число додавань при множенні без прискорення також дорівнювало б

5. Після перетворення множника число одиниць і число додавань-віднімань скоротилося до 3.

Таблиця 24

РХ	x_{n-1}^{*}	x_{n}^{*}	РУ	z_{0}^{*}	РZ	ЛЧК	Рφ	Пояснення
1101	1	1	100101	00	000000000000	000	0	Вихідний стан
				11	011011			-Y
				11	011011			Результат додавання
0110	1	1		11	1011011	001	1	Зсув
0011	0	1		11	11011011	010	1	Зсув
0001	1	0		11	111011011	011	1	Зсув
				11	011011			-Y
				11	010110011			Результат додавання
0000	1	1		11	1010110011	100	1	Зсув
0000	0	1		11	11010110011	101	1	Зсув
0000	0	0		11	111010110011	110	1	Зсув
				00	100101			+Y
				00	011111110011			Результат додавання
					Z	END		

Апаратні способи прискорення множення першого порядку базуються на аналізі не однієї цифри множника, а двох, трьох і більше таких цифр. При цьому апаратні способи звичайно комбінують з розглянутим вище логічним способом перетворення цифр множника. Крім того, процесор ККС повинен мати засоби для швидкого зсуву операндів на декілька розрядів і, в ряді випадків, мати можливість підготовки або швидкого формування операндів, кратних множеному, тобто, операндів вигляду 2Y,3Y,4Y і т.д.

Розглянемо спочатку такий варіант апаратного прискорення множення першого порядку, коли регістри РХ і РZ мають вбудовані засоби для зсуву їх вмісту на один і два розряди. Тоді при аналізі двох цифр множника можуть виникнути такі ситуації, обумовлені комбінаціями цих цифр:

1) $x_{n-1} = 0$, $x_n = 0$ – у цьому випадку немає необхідності виконувати додавання, треба лише зсунути множник Х і суму часткових добутків на два розряди вправо;

2) $x_{n-1} = 0$, $x_n = 1$ – необхідно додати Y до суми часткових добутків і далі виконувати зсув як у попередньому випадку;

3) $x_{n-1} = 1$, $x_n = 0$ – у цьому випадку виконується зсув вмісту РХ і РZ на один розряд вправо (відзначимо, що при наявності подвоєного множеного або можливостей для його швидкого одержання в цій ситуації можна було б виконувати додавання +2Y, а наступний зсув виконувати на два розряди);

4) $x_{n-1} = 1$, $x_n = 1$ – ця пара цифр множника перетвориться в тріаду 10 , на даному кроці множення виконується віднімання множеного, зсув на два розряди вмісту РХ і РZ і виробляється ознака $\Phi_{i+1} = 1$, що далі враховується так само, як і при логічному прискоренні множення.

ГСА такого множення приведена на рис. 39. На тих кроках множення, де виконуються операції додавання-віднімання множеного (тобто, +Y або –Y), а також на тих кроках, де черговий частковий добуток дорівнює нулю, але $x_{n-1} = x_n$, зсув в PZ і PX виконується на два розряди, тому що у всіх зазначених випадках наступна цифра перетвореного множника дорівнює нулю. З аналізу ГСА можна зробити висновок, що якщо на *i*-му кроці виконується зсув на два розряди, то $\Phi_{i+1} = x_{n+1}$, а при зсуві на один розряд $\Phi_{i+1} = x_n$. Це дозволяє використовувати в якості P_ϕ додатковий розряд у PX, зв'язаний з іншими розрядами засобами зсуву на 1 і 2 розряди, причому формування ознаки Φ_{i+1} у такому випадку зводиться до простого зсуву.

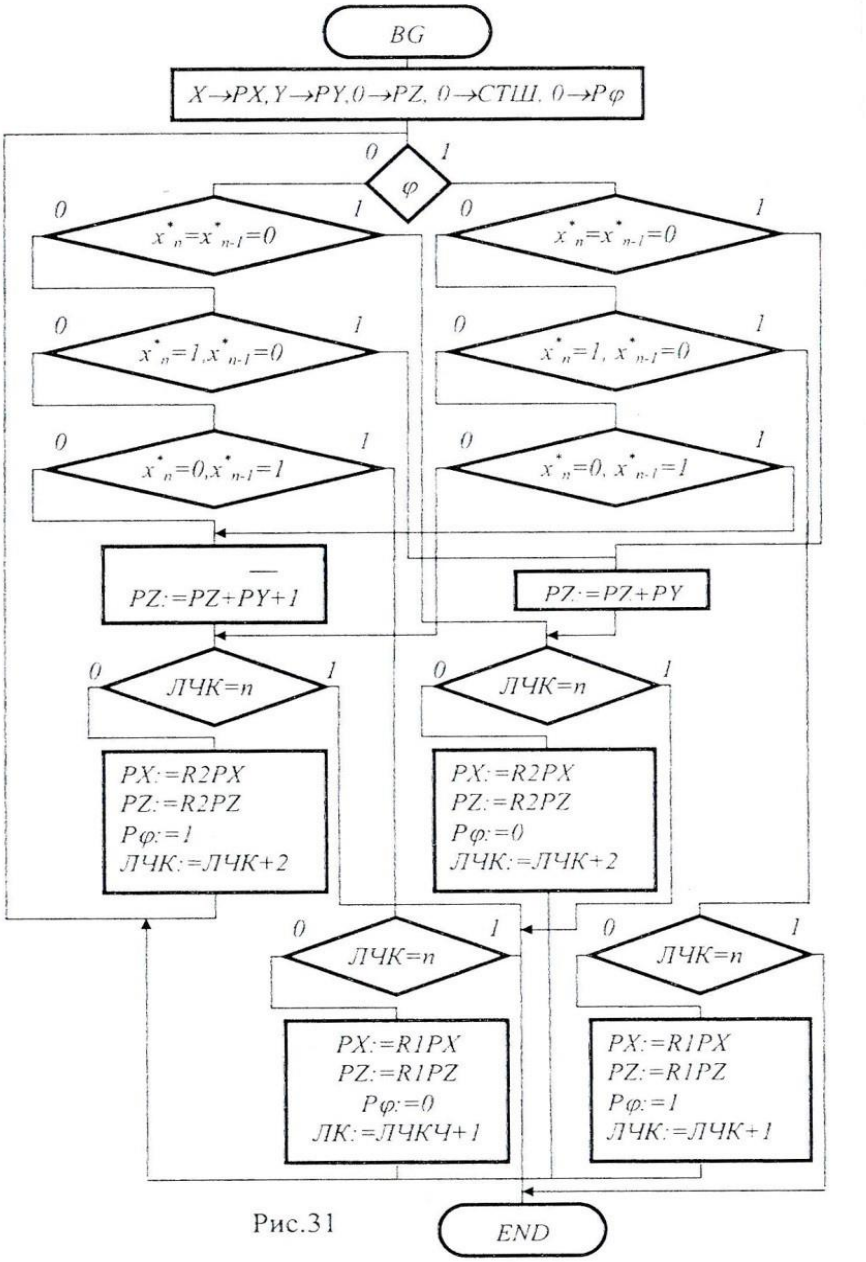


Рис.31

Рис.39. ГСА варіанту апаратного прискорення множення першого порядку, коли регістри РХ і PZ мають вбудовані засоби для зсуву їх вмісту на один і два розряди.

Рекомендується виконати практичне завдання 6.

Побудуємо цифрову діаграму множення чисел X = 45/64 і У = 38/64 за алгоритмом на рис. 39 (табл. 25). У двійковому поданні X = 0,101101, У = 0,100110.

Таблиця 25.

PX	$x_{n-1}^{\bullet}$	$x_n^{\bullet}$	PY	$z_{i_4}^{\bullet}$	PZ	ЛЧК	Рφ	Пояснення
1011	0	1	100110	00	000000000000	000	0	Вихідний стан
				00	100110			+Y
				00	100110000000			Результат додавання
0010	1	1		00	001001 10	010	0	Зсув 2
				11	011010			-Y
				11	100011100000			Результат додавання
0000	1	0		11	111000111000	100	1	Зсув 2
				11	011010			-Y
				11	010010111000			Результат додавання
0000	0	0		11	110100101110	110	1	Зсув 2
				00	100110			+Y
				00	011010101110			Результат додавання
					Z	END		

У цьому прикладі множення звелося до трьох зсувів на 2 розряди і чотирьох додавань (замість 6 зсувів-додавань – без прискорення). Середнє число додавань-віднімань при наявності в регістрах засобів зсуву на один і два розряди залишається таким же, як і при логічному прискоренні, тобто, 1/3 на розряд множника, Однак середнє число зсувів на розряд множника зменшується до 5/9, тому для розглянутого алгоритму

$$T_{МСР} = (n+1)\left(0{,}56t_c + 0{,}33t_+\right)$$

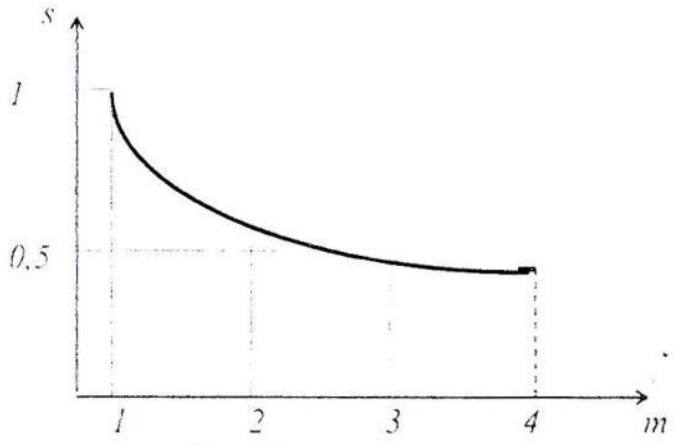


Рис.40.

Слід відзначити, що середнє число *s* зсувів на розряд множника зменшується непропорційно зростанню числа *m* розрядів, на яке можна виконувати зсув (рис. 40). Наприклад, при *m* = 1 середнє число зсувів *s* = 1, при *m* = 2 маємо *s* = 0,56, при *m* = 3 – *s* = 0,44 і т.д. Тому на практиці звичайно вибирають m<=4.

Інший апаратний спосіб прискорення множення першого порядку полягає у використанні чисел, кожне з яких є множене, збільшене в 2^{+i}: разів (інакше такі числа іще називають кратними множеного). Тут +і відповідає множенню з молодших розрядів, -і – множенню зі старших розрядів. а значення | і| залежить від числа аналізованих розрядів множники. Як приклад розглянемо випадок, коли є тільки кратні множеного виду $У2^0$ і $У2^1$. Алгоритм множення пояснюється табл. 25, де x_{n-2}, x_{n-1}, x_n., вміст трьох молодших розрядів множники. S – операція, виконувана на *і*-му кроці множення, а інші позначення відповідають прийнятим раніше. Зсув множника завжди здійснюється на два розряди, однак аналізувати треба три розряди множника, що сприяє зменшенню загального числа додавань-віднімань. Час множення за таким алгоритмом складає $T_{МСР}=(n+1)(0,5t_{зс}+0,33t_{+})$.

4.8. Основні алгоритми комп'ютерного ділення

Вибір алгоритмів виконання операції ділення в ККС залежить, головним чином, від способів і форматів подання операндів, співвідношення їх чисельних значень. використовуваної системи числення. Знак частки звичайно визначають шляхом додавання по модулю 2 цифр знакових розрядів діленого і дільника. При діленні чисел у формі з плаваючою комою порядок частки знаходять як різницю порядків операндів. Ділення мантис або ділення чисел у формі з фіксованою комою виконується над абсолютними величинами операндів, що подані, найчастіше, прямим кодом. Подібно до множення, ділення у ККС реалізується як багатокроковий процес виконання операцій додавання, віднімання, зсуву, інвертування та інших елементарних операцій. На відміну ж від множення цей процес має ітеративний характер, тому що результат ділення лише в досить рідкісних випадках практики обчислень може бути точно поданий числом скінченої довжини в деякій канонічній системі числення.

Відомо два основних алгоритми ***ділення*** в ККС – з ***відновленням*** і ***без відновлення залишку***. Ділення з відновленням залишку є трансформацією відомого десяткового алгоритму ділення "у стовпчик» для двійкової системи числення. Суть його полягає в наступному.

Нехай ділене X і дільник Y є правильними n-розрядними двійковими дробами, поданими прямим кодом. На першому кроці алгоритму дільник Y віднімають від діленого X і визначають знак “нульового» залишку R_0. Термін “нульовий» вказує на те, що відповідна цифра частки відноситься до розряду цілих, тобто має вагу 2^0. Якщо залишок R_0 додатний, тобто $|X| > |Y|$, то в розряді цілих частки встановлюється 1, далі формується ознака переповнення розрядної сітки й операція ділення припиняється. Якщо ж залишок R_0 від`ємний, то в розряді цілих частки встановлюється 0, а потім виконується відновлення діленого шляхом додавання до залишку дільника. Далі виконується зсув відновленого діленого на один розряд вліво і повторне віднімання дільника. Знак отриманого в такий спосіб залишку R_1 визначає першу після коми цифру частки: якщо залишок додатний, то цифра частки z_1, дорівнює 1; якщо залишок від’ємний, то z_1 дорівнює 0. Далі додатний залишок зсувають вліво і з нього знову віднімають дільник для визначення цифри z_2 частки; якщо ж залишок від’ємний, то до нього додають дільник для відновлення залишку, виконують зсув відновленого залишку і віднімання дільника для визначення наступної цифри частки. Ці дії повторюють до одержання потрібної кількості цифр частки з врахуванням додаткового розряду для округлення. Для правильною виконання додавання-віднімання при реалізації такого алгоритму необхідно оперувати з двома знаковими розрядами, тому що при зсуві вліво може відбуватися передача значущої одиниці зі старшого розряду залишку в знаковий і, таким чином. спотворення знака залишку. При наявності ж двох знакових розрядів правильний знак залишку завжди зберігає додатковий знаковий розряд, а одиниця в основному знаковому розряді повинна сприйматися як така, що відноситься до розряду цілих (тобто. у випадку зсуву числа 00,1… одержуємо число 01,…).

Таблиця 26.

Пояснення	Операції, операнди, результати	Цифри частки
X	00,0111	
$-Y$	11,0011	
R_0	11,1010	→
$+Y$	00,1101	$z_0=0,\ Z=0,...$
Відновлення X	00,0111	
Зсув	00,1110	
$-Y$	11,0011	
R_1	00,0001	
Зсув	00,0010	→ $z_1=1,\ Z=0,1...$
$-Y$	11,0011	
R_2	11,0101	→
$+Y$	00,1101	$z_2=0,\ Z=0,10...$
Відновлення додатного залишку	00,0010	
Зсув	00,0100	
$-Y$	11,0011	
R_3	11,0111	→
$+Y$	00,1101	$z_3=0,Z=0,100...$
Відновлення додатного залишку	00,0100	
Зсув	00,1000	
$-Y$	11,0011	
R_4	11,1011	
$+Y$	00,1101	→ $z_4=0,\ Z=0,1000...$
Відновлення додатного залишку	00,1000	
Зсув	01,0000	
$-Y$	11,0011	
R_5	00,0011	→
		$z_5=1,\ Z=0,10001$

Рекомендується виконати практичне завдання 7.

Виконаємо за алгоритмом ділення з відновленням залишку ділення числа X = 7/16 на число Y = 13/16 при n = 4 (табл. 26). В двійковому поданні X = 00,0111, Y = 00,1101, (-Y)дк = 11,0011. Частка з недостачею дорівнює 1/2, тобто 0,1000, а частка з надлишком дорівнює 9/16, тобто, 0,1001

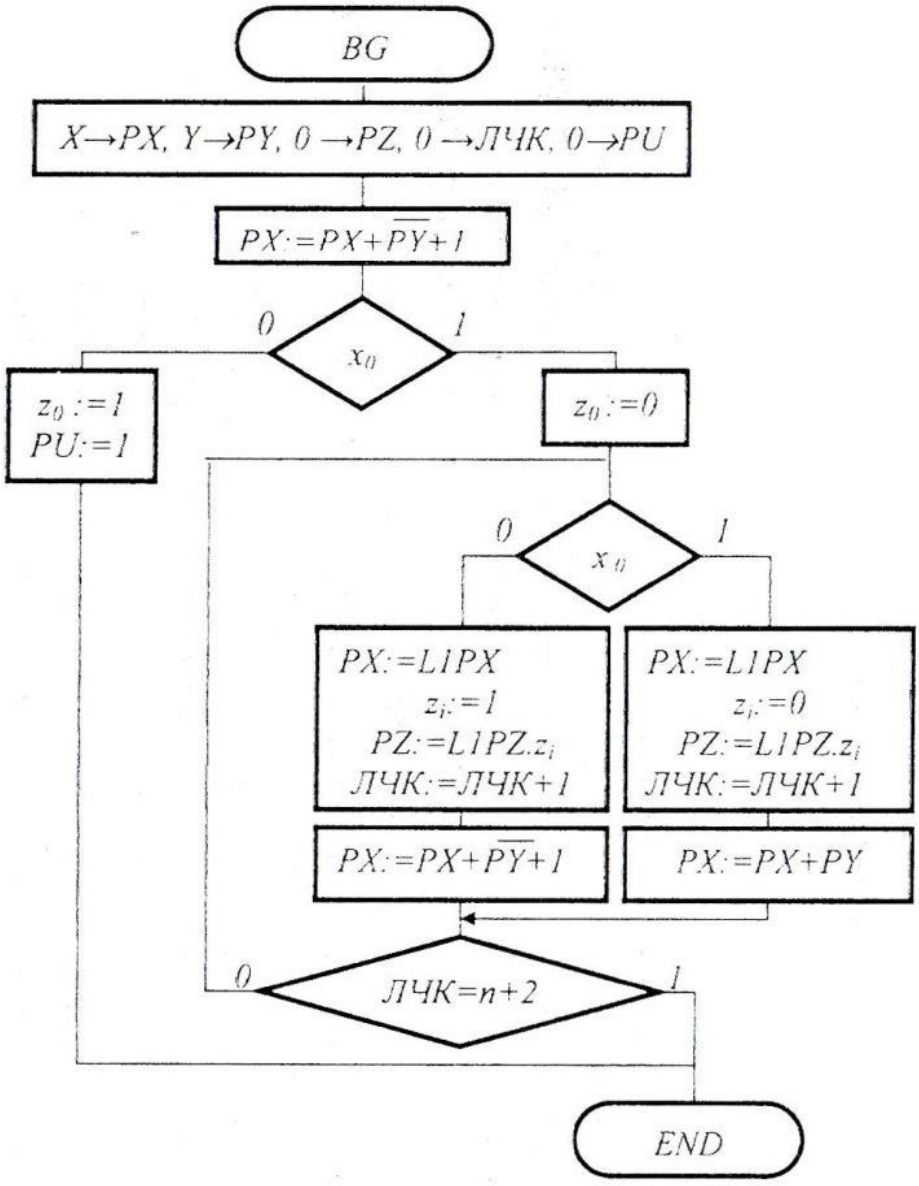


Рис. 41. ГСА алгоритму ділення без відновлення залишку.

Як видно з прикладу, ділення з відновленням залишку є нерегулярним процесом у тому розумінні, що для одержання однієї цифри частки необхідно виконувати або одне віднімання і зсув. або одне віднімання, одне додавання і зсув. У середньому. без врахування часу для одержання додаткового розряду для округлення, тривалість ділення за розглянутим алгоритмом складатиме $T_{д1}=(n+1)(1,5t_{+}+t_{з})$.
Тривалість ділення можна досить суттєво зменшити в порівнянні з вищенаведеною оцінкою (за рахунок виключення “зайвої» операції відновлення залишку), використовуючи алгоритм ділення без відновлення залишку. реалізація якого не вимагає ніяких додаткових апаратурних витрат. Це і послужило причиною того, що в переважній більшості сучасних ККС використовується саме цей алгоритм ділення.

Суть алгоритму ділення без відновлення залишку полягає в наступному (як і раніше, вважаємо. шо операції подано у формі з фіксованою комою).

1. Одержати різницю $R_0 = X - Y$. Якщо $R_0 \geq 0$,, то цифра z_0 частки, що має вагу 2^0, дорівнює 1. Ділення у цьому випадку необхідно припинити, тому шо відбувається переповнення розрядної сітки і порушення форми поданння результату. Якщо ж $R_0<0$, то $z_0 = 0$. ділення можна продовжувати. Різниця R_0, далі називається залишком,

2. Подвоїти залишок (одержати число $2R_i$)або зменшити в два рази дільник.

3. Якщо $2R_i \geq 0$, то відняти, а якщо $2R_i < 0$, то додати до залишку дільникY. Якщо знову отриманий залишок $R_{i+1} \geq 0$, то $z_{i+1}=1$, інакше $z_{i+1}=0$.

4. Повторювати п. п. 2 і 3 цього алгоритму до одержання потрібної кількості цифр частки.

Зауважимо, шо п. 2 алгоритму можна замінити на "Зменшити в два рази дільник". ГСА для розглянутого випадку приведена на рис. 37.

Очевидно, шо відповідно до цього алгоритму для одержання частки, шо має n+1 цифр після коми, потрібен час $T_{Д2}=(n+2)(t_+ + t_{зс})$. У випадку, коли буде вироблена ознака переповнення розрядної сітки, необхідно ділене X зсунути вправо і знову почати ділення. Після завершення операції частку зсувають вліво на те число розрядів, на яке зсувалося вправо ділене.

З алгоритму випливає, що значення частки завжди протилежні: знаку відповідного залишку, тобто, $z_i = \overline{x}_0$.Оскільки перед початком операції в *PZ* записаний 0, то кінець операції можна визначати за допомогою цього регістра. Для цього *PZ* повинен мати додатковий старший розряд, а в молодший його розряд перед початком ділення записується одиниця. Після необхідного числа кроків ця одиниця опиниться в додатковому старшому розряді, що і буде вказувати на закінчення операції. Наявність другої інтерпретації п.2 алгоритму' ділення без відновлення залишку дає другий варіант ГСА на рис.41, де всі оператори виду *PX:=LIPX* слід замінити на оператори *PY:=RIPY*. Однак регістри *PX, PY* і суматор при цьому повинні мати подвійну розрядність,

Рекомендується виконати практичне завдання 8.

Складемо цифрову діаграму ділення X=15/32 на У=24/32 по алгоритму на рис.41 (табл26, де в двійковому поданні X=00,01111, Y=00,11000, $(-Y)_{дк}$ =11,01000) до одержання п`яти цифр результату після коми.

З діаграми (табл.27) видно, що після четвертого додавання - віднімання утворився нульовий залишок і всі наступні цифри частки дорівнюють нулю. Тому після утворення нульового залишку можна виконувати тільки зсуви для того, щоб цифри частки займали відповідні вагові позиції в регістрі PZ. Очевидно, що якщо доповнити ГСА перевіркою умови (PX=0) по кожній з гілок алгоритму, то можна прискорити його виконання.

Таблиця 27.

PX	*PY*	*PZ*	*ЛЧК*	Пояснення
00.01111	*.11000*	*0.000000*	*000*	Вихідний стан
11.01000				*-Y*
11.10111		z_0		R_0
11.01110		*0*	*001*	Зсув
00.11000				*+Y*
00.00110		z_1		R_1
00.01100		*01*	*010*	Зсув
11.01000				*-Y*
11.10100		z_2		R_2
11.01000		*010*	*011*	Зсув
00.11000				*+Y*
00.00000		z_3		R_3
00.00000		*0101*	*100*	Зсув
11.01000				*-Y*
11.01000		z_4		R_4
10.10000		*01010*	*101*	Зсув
00.11000				*+Y*
11.01000		z_5		R_5
10.10000		*010100*	*110*	Зсув
00.11000				*+Y*
11.01000		z_6		R_6
10.10000		*0101000*	*111*	Зсув
00.11000				*+Y*
11.01000			*END*	R_7

Рекомендується виконати практичне завдання 9

Оскільки ділсння мснше часто ‘зустрічається в програмах ніж інші арифмєтичпі опсрації, то всі ККС, де множсння виконується прискорено, часто реалізують ділсння шляхом множення діленого X на величину, обернену ділъннку У,тобто на Q=1/Y. Для одержання досить точного значения Q користуються ітераційними формулами, що не містять операції ділення, напрнклад,

$$Q_{i+1}=Q_i(1+e_i),$$

де Q_{i+1} – i+1 наближення величини Q, зворотної дільнику, Q_i – представлення Q з відносною похибкою e_i.

4.9. Алгоритми і похибки комп'ютерного округлення. Аналіз точності обчислень

Поняття точності виражає деяку узагальнену властивість технічних систем, яку неможливо визначити безвідносно до конкретної технічної області [1]. Необхідність аналізу точності цифрових обчислень в комп'ютерах та комп'ютерних систем (далі ККС) і вивчення основних понять теорії точності обумовлена двома основними причинами [2]. По-перше, точність є однією з найважливіших структурно-функціональних характеристик цифрових обчислювальних засобів широкого призначення поряд з їхньою продуктивністю, вартістю, надійністю та ін. [3]. Для деяких спеціалізованих ККС досягнення високої точності є цільовою функцією їхнього створення, тобто, точність служить і основною характеристикою їхньої ефективності. Друга причина за своїм походженням є внутрішньою для обчислювальної техніки і полягає в тому, що точність у цифрових обчислювальних засобів забезпечується відповідним вибором розрядності операндів або ширини розрядної сітки. Для обґрунтування такого вибору і необхідно вивчати основні поняття теорії точності.

Існування проблем точності для обчислювальних засобів викликане наближеним характером послідовностей операцій і мікрооперацій, скінченністю розрядної сітки, похибками округлений, неточністю вихідної числової інформації. Тому результати обчислення повинні супроводжуватися визначенням або оцінкою похибок обчислень, що подаються у вигляді сумарної (повної) похибки від усіх джерел похибок або окремо по кожному джерелу. Встановлення залежності похибок обчислень від характеру дії різних їх джерел та від параметрів обчислювальних засобів і складає предмет аналізу точності [1].

Далі для визначення характеристик точності обчислювальних засобів будемо вважати, що їх функціонування у векторній формі описується виразами вигляду $Z = F(X)$, де $Z = (Z_1, Z_2, \ldots, Z_m)$ – вектор результатів; $F = (F_1, F_2, \ldots F_m)$ – вектор-алгоритм (набір функцій, які обчислюються, операторів, перетворень, відображень), що виконується над вектором операндів $X = (X_1, X_2, \ldots X_n)$ (аргументів, вихідних даних). Векторна форми відповідає системі виразів вигляду: $Z_1 = F_1(X_1, X_2, \ldots X_n)$, $Z_2 = F_2(X_1, X_2, \ldots, X_n)$, $Z_m = F_m(X_1, X_2, \ldots, X_n)$.

Застосування вектора-алгоритму F до конкретного набору значень X_i називають реалізацією алгоритму F. Кожна реалізація характеризується не тільки конкретним вектором X, але і рядом параметрів, що впливають на точність результату. Ці параметри називають апроксимуючими параметрами обчислень. Завдання цих параметрів разом з вихідною інформацією X визначає єдиний наближений результат Z. Реалізація алгоритму за відсутності апроксимуючих параметрів визначає істинний (еталонний, точний) результат Z_i. Істинний результат і всілякі наближені результати, що отримуються при різних

значеннях апроксимуючих параметрів, утворюють множину можливих результатів *R*. Розуміючи точність як ступінь близькості результатів деякої реалізації алгоритму *Z* до істинного результату Z_i, можна вважати, що задача аналізу точності полягає у визначенні відстані $R = r(Z_i,Z)$ між елементами Z_i і *Z* у множині *R*.

Функція відстані визначається різними способами в залежності від форми задання наближеного результату Z. Так званий точковий результат одержують у випадку, коли він заданий деяким єдиним Z є R, близьким до Zi. Для точкового наближеного результату як функцію R простіше всього використовувати значення різниці Д наближеного і істинного результатів без врахування її знаку, тобто: $\Delta = (\Delta_1, \Delta_2, ..., \Delta_m), \Delta_j = |Z_{ij} - Z_i|,\ j = 1, m.$

Цю функцію називають абсолютною похибкою результату. Однак для засобів обчислювальної техніки абсолютна похибка часто має лише теоретичне значення, тому що при заздалегідь невідомих компонентах вектора *X* і істинне рішення Z_i звичайне невідоме (за винятком спеціальних контрольних або тестових обчислень). Тому на практиці в обчислювальній техніці задають максимальне значення абсолютної похибки, що визначає границі результату [4].

Іншою функцією відстані між точковим наближеним результатом Д і істинним є відносна похибка *∂*, обумовлена відношенням абсолютної похибки до абсолютної величини істинного результату $\partial = (\partial_1, \partial_{2,...,} \partial_m),\ \partial_i = Д_i|Z_{ij}|,\ i = 1,m.$

Як і абсолютна, відносна похибка д може бути задано максимальним значенням. Інтервальний наближений результат вказує границі зміни результату при зміні апроксимуючих параметрів у деякій заданій області *S*. Гранична похибка в такому випадку визначається як верхнє значення функції $r(Z_i,Z)$ по всій області *S*, тобто $R = supr(Z_i,Z)$.

Ймовірнісний наближений результат одержують у випадку, коли відомий закон розподілу ймовірностей результату *Z* у множині *R*. Статистична сукупність, необхідна для визначення такого розподілу, утворюється множиною реалізацій алгоритму *F* для всіх можливих значень апроксимуючих параметрів. Характеристикою відстані *r* служить середнє квадратичне відхилення розподілу наближеного результату та довірча ймовірність.

У загальному випадку ступінь близькості наближеного і істинного результатів залежить від ряду факторів, що впливають на процес обчислень. Тому для аналізу точності засобів обчислювальної техніки виділяють повну похибку результату і її компоненти: методичну, трансформовану й арифметичну похибки**.**

Для визначення методичної похибки відзначимо, що вихідний алгоритм *F* є звичайно математичною моделлю деякого процесу або ж відповідає абстрактній математичній задачі (наприклад, обчисленню визначеного інтеграла). Однак, як для одних алгоритмів, так і для інших характерна наближеність відповідності між реальними процесами і їх математичними описами (обчислення визначеного інтегралу зводиться до обчислення суми

площ прямокутників або трапецій). Похибка, породжувана наближеним характером вихідного алгоритму, називається похибкою алгоритмізації. Така похибка є об'єктивним атрибутом будь-якого процесу обробки інформації. При програмуванні вихідний алгоритм F подається як скінчена послідовність арифметичних і логічних операцій, тобто, вихідний алгоритм F повинен бути перетворений у комп'ютерний алгоритм F_k, сформульований у поняттях і термінах чисельного аналізу. Таким чином, F_k виявляється лише наближеним варіантом F, а відповідна похибка називається похибкою чисельного подання алгоритму. Похибка результату, обумовлена похибками алгоритмізації і чисельного подання, називається методичною похибкою обчислень і оцінюється відстанню $r = r(F(X),F_k(X))$. Тут $F(X)$ має характер точного результату. Аналізом методичних похибок займається теорія наближених обчислень.

Трансформована (внесена, неусунена, успадкована) похибка наближених обчислень виникає, у першу чергу, через похибки у вихідних даних (наприклад, через неточне їх вимірювання або обчислення на попередніх етапах). Може також виявиться, що операнди не можна точно представити скінченим числом цифр у випадках, коли операнди – ірраціональні числа (наприклад, квадратні корені з 2 і 3, числа р, e і т.п.). Крім того, дроби, скінчені в одній системі числення, можуть виявитися нескінченними в іншій системі. Похибки переводу чисел з однієї системи числення в іншу можуть бути і в тому випадку, коли самі вихідні дані абсолютно точні. Застосування машинного алгоритму F_k до приблизно заданої вихідної інформації X_k дає результат $Fk(X_k)$.Значення трансформованої похибки оцінюється відстанню $r = r(F_k(X),F_k(X_k))$. Аналітична форма для оцінки трансформованої похибки має вигляд:

$$\Delta_{mZi} = \sum_{j-1}^{n}\left(\frac{\partial \varphi i}{\partial x_i}\right)\Delta x_j; i = \overline{1,m;}\ j;\ j = \overline{1,n.}$$

$Д_j$ трансформована похибка j-ої компоненти вектора Z; вираз в круглих дужках являє собою часткову похідну від F_i по X_i обчислену за умови $Дx_i = 0$; $Дx_i$ – похибка i-ої компоненти вектора X. Цю форму називають також лінеарізованою формулою для оцінки трансформованої похибки, тому що вона не враховує величини другого та всіх більш високих порядків малості. Зауважимо, що трансформована похибка залежить не тільки від похибок вихідних даних, але і від самих вихідних даних. При цьому можливі ситуації, коли малі похибки вихідних даних приводять до великих похибок результату. Подібні алгоритми і відповідні їм задачі називають некоректно поставленими. Необхідно відзначити, що при реалізації будь-якого обчислювального алгоритму відбувається певне "посилення" помилок вихідних даних, тому точність результату ніколи не може бути вище точності вихідних даних.

В аналізі точності обчислень розрізняють пряму і обернену задачі аналізу помилок. Перша полягає в визначенні трансформованої похибки результату по похибках вихідних даних, друга – у визначенні допустимих похибок вихідних даних по заданій похибці результату.

Арифметична похибка обумовлена обмеженою точністю виконання арифметичних операцій, що, у свою чергу, є наслідком виконання операцій округлення і скінченої довжини подання чисел (як, наприклад, при множенні з округленням до n розрядів і діленні). Реалізація алгоритму F_k в арифметиці скінченої довжини дає наближений результат $F_k^k(X_k\}$. Величина арифметичної похибки оцінюється відстанню $r = r(F_k(X),F_k^k(X_k))$.

Повна (сумарна) похибка наближеного результату обчислень, оцінюється відстанню $r = r(F_i(X),F_k^k(X_k))$ між істинним результатом $F_i(X)$ і результатом, отриманим за одночасної дії усіх факторів, що впливають на точність – $F_k^k(X_k)$.

Трансформовану і арифметичну похибки в сукупності називають також обчислювальними похибками, оскільки вони визначаються властивостями обчислювальних засобів, а не властивостями алгоритмів. Варто зазначити, що практичне визначення повної похибки утруднюється тим, що істинний результат обчислень часто наперед невідомий. Тому повну похибку обчислень одержують як композицію складових, що приймають, наприклад, сенс абсолютних похибок, тобто:

методичної $Д_м = |F(X)-F_k(X)|$,

трансформованої $Д_m = |F_k(X)-F_k(X_k)|$,

та арифметичної $Д_a = |F_k(X_k)-F_k^k(X_k)|$.

Таким чином, повна абсолютна похибка результату становить

$Д = Д_м + Д_m + Д_a$.

В табл. 28 обумовлені певними апроксимуючими факторами розглянуті вище похибки позначені знаком V. При використанні ймовірнісних оцінок для визначення складових повної похибки, наприклад, середніх квадратичних відхилень одержують середнє квадратичне відхилення повної похибки. При використанні ймовірнісних оцінок для визначення складових повної похибки, наприклад, середніх квадратичних відхилень $\partial_т$, $\partial_м$, $\partial_а$, одержують середнє квадратичне відхилення повної похибки:

$$\delta = \sqrt{\delta_{T^2} + \delta_{м^2} + \delta_{a^2}}.$$

Таблиця 28

Апроксимуючі фактори	Похибки (Повна)		
		Похибка обчислювальна	
	Похибка методична	Похибка трансформована	Похибка арифметична
Наближеність алгоритмів обробки інформації	V		
Чисельне подання алгоритмів	V		
Неточність вимірювання		V	
Неточність аналого-цифрових перетворень		V	
Неточність попередніх обчислень		V	
Скінченість подання ірраціональних чисел і дробів		V	
Округлення			V
Обмеженість розрядної сітки			V

Такі оцінки повної похибки наближеного результату виявляються дуже завищеними, тому що в реальних обчисленнях часто виникають складові повної похибки з різними знаками, що приводить до їх часткової або навіть повної компенсації. Однак така компенсація дуже важко піддається врахуванню в загальному вигляді.

Розглянуті складові повної похибки можуть бути віднесені до статичних похибок, тому що вони не є функціями часу. При використанні ККС для управління об’єктами в реальному часі з’являється ще і динамічна похибка обчислень, обумовлена скінченою швидкістю роботи ККС, їх обмеженою швидкодією [5]. Будемо вважати, що при реалізації алгоритму $F(X)$ вихідні дані X і саме наближене рішення Z є функціями часу, тобто, $X = X(t)$, $Z = Z(t)$. Тоді для вихідних даних $X = X(t)$, визначених на момент часу t, рішення буде отримано на момент $t+\theta$, де θ – затримка, що дорівнює часу реалізації алгоритму. Отже, динамічну помилку наближених обчислень можна визначити як $Д_Д = Z(t) – Z(t+\theta)$.

4.10. Похибки подання чисел і арифметичних операцій

Округлення в комп’ютерній арифметиці – операція, яка чисельному значенню операнда визначеної довжини (далі – точному операнду)ставить у відповідність деяке наближене його значення, що має меншу довжину. З чисельного аналізу відомі два основних підходи до трактування змісту цієї операції: округлення з недостачею і округлення з надлишком. Округлення з недостачею означає, що між точним операндом X і результатом округлення Z має місце співвідношення $|X|\geq|Z|$, округлення з надлишком характеризується співвідношенням $|X|\leq|Z|$. Однак ці підходи ще не визначають алгоритмів реалізації округлення стосовно до двозначного кодування операндів і їх форматів. Далі для визначеності будемо розглядати формат з фіксованою перед старшим розрядом комою, що має n розрядів для X і m розрядів для Z, $n>m$. Найбільше поширення для такого формату одержали наступні алгоритми

округлення: а) відкидання молодших розрядів; б) установка молодшого розряду, що залишається у форматі, в той стан, що відповідає його логічній сумі з цифрою, яка виходить за формат (наприклад, при зсуві); в)додавання одиниці до старшого розряду, що виявиться за форматом.

Алгоритм а) найпростіший по реалізації, однак він вносить похибку Д в абсолютну величину *Z*. Значення цієї похибки визначається межами $0 \geq Д \geq -(2^{-m}-2^{-n})$, тобто, такий алгоритм є округленням з недостачею, а середня похибка округлення складає:

$$\Delta_{сер} = 2^{-1}\left(2^{-n} - 2^{-m}\right) = -2^{-1}\left(2^{-m} - 2^{-n}\right) = -2^{-m-1}\left(1 - 2^{m-n}\right) = -2^{-m-1}\left(1 - 2^{-(n-m)}\right).$$

Таким чином, максимальна по абсолютній величині похибка тут не перевищує одиниці молодшого розряду, що залишається у форматі, а середня – одиниці старшого розряду, що виходить за формат при округленні.

Алгоритм б) частіше використовують тоді, коли округлення виконується на один розряд, тобто, коли $n-m=1$. У такому випадку похибка округлення знаходиться в межах $-2^{-n} \leq Д \leq 2^{-n}$, а її середнє значення дорівнює нулю. Якщо ж $n-m>1$, то похибка округлення знаходиться в межах $-2^{-m}(1-2^{-(n-m)}) \leq Д \leq 2^{-m-1}$, а її середнє значення складає:

$$\Delta_{сер} = 2^{-1}\left(2^{-m-1} - 2^{-m}\left(1 - 2^{-(n-m)}\right)\right) = -2^{-m-1}\left(1 - 2^{(-n-m)}\right) + 2^{-m-2} = -2^{-m-1}\left(1 - 2^{-(n-m)} - 2^{-1}\right) =$$

$$= -2^{-m-1}\left(2^{-1} - 2^{-(n-m)}\right).$$

що дещо менше, ніж середня похибка по алгоритму а). Алгоритм б) можна застосовувати і послідовно, тобто, *n*-розрядний операнд округлити до *n-1*-розрядного, потім до *n-2*-розрядного і т.д., поки не буде отриманий *m*-розрядний результат. У цьому випадку межі похибки округлення будуть ширшими, а саме $-2^{\sim m}(1-2^{\sim(n\sim m)}) \leq Д \leq 2^{-m}(1-2^{-(n-m)})$. Однак середнє її значення дорівнює нулю.

Алгоритм в) визначає наступні межі похибок округлення: $-2^{-m}(1-2^{-(n-m)+1}) \leq \Delta \leq 2^{-m-1}$ із середнім її значенням $Д_{сер} = 2^{-n-1}$. Важливою перевагою цього алгоритму є незалежність середньої похибки від *m*. Якщо $n-m=1$, то алгоритм в) дає або нульову, або додатну похибку і в цьому випадку може вважатися округленням з надлишком.

При використанні будь-якого алгоритму абсолютна величина похибки округлення не перевищує одиниці (тобто, половини ваги) молодшого розряду, що залишається в форматі. Однак виконання операцій над операндами, отриманими шляхом округлення, може приводити до результатів, похибка яких більше половини ваги молодшого у форматі розряду. Наприклад, додавання двох операндів, що були округлені за алгоритмами б) і в), уже може приводити до похибки суми, що дорівнює одиниці молодшого розряду. Виконання послідовності операцій з такими операндами може приводити до нагромадження великої трансформованої похибки. Це наочно видно на наступному прикладі в десятковій арифметиці. Нехай необхідно додати чотири числа: 0,116; 0,126; 0,136 і 0,146. їхня сума дорівнює 0,524, а округлена до двох розрядів після коми сума дорівнює 0,52. Якщо ж числа, що додаються, округлити відразу до двох розрядів після коми, то одержимо: 0,12; 0,13; 0,14 і

0,15. Їх сума дорівнює 0,54, тобто, трансформована похибка додавання цих чотирьох операндів склала дві одиниці молодшого у форматі розряду. Очевидно, що чим довша послідовність округлених операндів, які додаються, тим помітніший ефект нагромадження похибок. Через скінченність розрядної сітки ККС множина чисел, які можна подати у ККС, також скінченна. Тому в подання чисел у ККС вноситься похибка, значення якої залежить як від довжини операндів, так і від форми подання чисел. Далі будемо визначати абсолютну похибку операнда як різницю між його точним значенням X та його машинним поданням $X_м$, тобто, $Д=X-X_м$.

Максимальна абсолютна похибка для ККС, що використовують форму подання чисел з фіксованою комою, постійна і дорівнює половині ваги молодшого розряду. Відносна ж похибка для такої форми залежить від значення операнда.

Для ККС, що використовують форму подання чисел з плаваючою комою, абсолютна похибка подання числа визначається як

$$\Delta = \Delta_M 2^P,$$

де $Д_м$ – абсолютна похибка мантиси, що визначається так само, як і для фіксованої коми; P – порядок числа.

Оскільки сам операнд може змінюватися в межах від X_{min} до X_{max} то і відносна похибка подання чисел залежить від кількісного еквівалента чисел. Звідси випливає, що для малих чисел, що наближаються до X_{min}, відносна похибка може досягати 50%.

Максимальна абсолютна похибка для ККС, що використовують форму подання чисел з фіксованою комою, постійна і дорівнює половині ваги молодшого розряду.

У свою чергу порядок P знаходиться в межах $-(2^m-1)\le P\le(2^m-1)$.

Тому мінімальна абсолютна похибка буде при найбільшому від'ємному порядку, а максимальна – при найбільшому додатному порядку, тобто:

$$\Delta_{\min} = 2^{-n-1} \wedge 2^{-(2\wedge m-1)} = 2^{-n-2\wedge m},$$

$$\Delta_{\max} = 2^{-n-1} * 2 \wedge \left(2^{m-1}\right) = 2 \wedge \left(2^m - n - 2\right).$$

Відносна похибка подання чисел для ККС з плаваючою комою визначається за загальним правилом:

$$\delta = I\Delta(X) = \left(\Delta_M 2^P\right) / \left(M(X) 2^P\right) = \Delta_M / M(X),$$

де $M(X)$ – мантиса операнда X, тобто, відносна похибка не залежить від порядку P і знаходиться в межах $д_{min}=Д_M/M_{max}=2^{-n-1}/1\text{-}2^{-n}\approx 2^{-n-1;}$ $д_{max}=Дм/M_{min}=$ $=2^{-n-1}/2^{-1}\approx 2^{-n}$.

Таким чином, у ККС з плаваючою комою відносна похибка подання чисел мало залежить від величини числа (змінюється по всьому діапазону подання чисел лише вдвічі від 2^{-n-1} до 2^{-n}) і тим менше, чим більше n.

Для оцінки трансформованої похибки результату виконаним послідовності операцій необхідно попередньо оцінити похибки результатів кожної з операцій. Одним із джерел цих похибок, у свою чергу, є похибки чисельного подання [6]. Це значить, що кожний операнд X у комп'ютерному

поданні $X_к$ відрізняється від точного його значення на величину абсолютної похибки, тобто, $X= X_к+Д_X$,$Y=Y_к+Д_y$ і т.д. Результат операцій додавання і віднімання можна записати в такому ж вигляді, а також можна оцінити абсолютні і відносні похибки й інших операцій.

Таким чином як висновки аналізу похибок комп'ютерного округлення та похибок подання чисел і арифметичних операцій необхідно підкреслити, що необхідність аналізу точності цифрових обчислень в комп'ютерах і комп'ютерних системах обумовлена двома основними причинами [7]. По-перше, точність є однією з найважливіших структурно-функціональних характеристик цифрових обчислювальних засобів широкого призначення поряд з їхньою продуктивністю, вартістю, надійністю.

Друга причина полягає в тому, що точність у цифрових обчислювальних засобів забезпечується відповідним вибором розрядності операндів або ширини розрядної сітки.

В аналізі точності обчислень розрізняють пряму і обернену задачі аналізу помилок. Перша полягає в визначенні трансформованої похибки результату по похибках вихідних даних, друга – у визначенні допустимих похибок вихідних даних по заданій похибці результату.

Трансформовану і арифметичну похибки в сукупності називають також обчислювальними похибками, оскільки вони визначаються властивостями обчислювальних засобів, а не властивостями алгоритмів [8]. Варто зазначити, що практичне визначення повної похибки утруднюється тим, що істинний результат обчислень часто наперед невідомий. Тому повну похибку обчислень одержують як композицію складових, що приймають сенс абсолютних похибок:

- методичної $Д_м= |F(X)-F_k(X)|$;
- трансформованої $Д_m=|F_k (X)-F_k (X_k)|$;
- арифметичної $Д_a= |F_k (X_k)-F_k^k (X_k)|Д =Д_м+ Д_m+ Д_a$.

Остання формула виражає повну абсолютну похибку результату.

Максимальна абсолютна похибка для ККС, що використовують форму подання чисел з фіксованою комою, постійна і дорівнює половині ваги молодшого розряду. Відносна ж похибка для такої форми залежить від значення операнда $∂=|Д/X|=|2^{-n-1}/X|$.

Для ККС, що використовують форму подання чисел з плаваючою комою, абсолютна похибка подання числа визначається як $Д=Д_M2^P$, де $Д_м$ – абсолютна похибка мантиси, що визначається так само, як і для фіксованої коми; P – порядок числа.

Відносна похибка подання чисел для ККС з плаваючою комою визначається за загальним правилом $∂=|Д/X| =(Д_м2^p)/(M(X)2^P) = Д_м/M(X)$, де $M(X)$ – мантиса опреранда X, тобто, відносна похибка не залежить від порядку P і знаходиться в межах $∂_{min}=Д_M/M_{max} = 2^{-n-1}/1-2^{-n}≈2^{-n-1}$.

Таким чином, у ККС з плаваючою комою відносна похибка подання чисел мало залежить від величини числа (змінюється по всьому діапазону

подання чисел лише вдвічі від 2^{-n-1} до 2^{-n}) і тим менше, чим більше *n*.

КОНТРОЛЬНІ ПИТАННЯ

1. Інформація та загальні принципи її перетворення.
2. Сигнал, як засіб передачі повідомлення.
3. Дискретний сигнал як форма передачі інформації на відстані.
4. Керуючі автомати з схемною логікою. Особливості структурної побудови.
5. Структурний синтез керуючого автомата з схемною логікою.
6. Етапи структурного синтезу МПА з схемною логікою.
7. Синтез мікропрограмного автомата з програмованою логікою.
8. Центральний пристрій керування комп'ютером.
9. Закони булевої алгебри.
10. Канонічні форми булевих функцій. Проблема розв'язуваності.
11. Нормальні та досконалі диз'юнктивні нормальні форми логічних функцій.
12. Основні параметри комп'ютера.
13. Конфігурація комп'ютера.
14. Нові технології обробки інформації в сучасних комп'ютерах.
15. Процесор, типова структура процесора.
16. Приклади виразів булевої алгебри.
17. Структура персонального комп'ютера.
18. Основні режими роботи комп'ютера.
19. Конфігурація сучасних комп'ютерів та порядок представлення основних пристроїв і їх параметрів.
20. Режими та технології роботи процесора.
21. Використання BIOS в програмному забезпеченні автоматизованого робочого місця користувача.
22. Загальна характеристика пам'яті в сучасних комп'ютерах.
23. Цифрові керуючі автомати зі схемною логікою.
24. Цифрові керуючі автомати із програмованою логікою.
25. Синтез цифрових керуючих автоматів зі схемною логікою.
26. Синтез цифрових керуючих автоматів із програмованою логікою.
27. Цифрові керуючі автомати із програмованою логікою як основа побудови процесорів сучасних комп'ютерів.
28. Арифметико-логічні пристрої сучасного персонального комп'ютера.
29. Пристрої управління сучасного персонального комп'ютера та їх класифікація.
30. Дати пояснення спрощеної типової структурної схеми процесора сучасного персонального комп'ютера.
31. Основні параметри пам'яті сучасного персонального комп'ютера.

32. Структура та призначення центрального пристрою управління сучасного персонального комп’ютера.
33. Структурний синтез керуючого автомата з схемною логікою.
34. Синтез мікропрограмного автомата з програмованою логікою.
35. Структура МПА з програмованою логікою.
36. Послідовності дій виконання кожної мікропрограми мікропрограмним автоматом з програмованою логікою.
37. Ознаки класифікації мікропрограм них автоматів з програмованою логікою.
38. Переваги використання МПА з програмованою логікою порівняно з схемною.
39. Центральний пристрій керування, його структура, типові дії у кожному робочому циклі.
40. Алгоритми додавання чисел з фіксованою комою в прямих кодах. Навести приклад.
41. Алгоритми віднімання чисел з фіксованою комою в прямих кодах.
42. Комп’ютерні аспекти теорії систем числення.
43. Системи числення з безпосереднім представленням цифр.
44. Вимоги до систем числення в обчислювальній техніці: однозначність; кінцевість; ефективність.
45. Двійково – кодовані системи числення.
46. Форми комп’ютерного представлення чисел.
47. Форми представлення чисел з фіксованою комою. Навести приклад.
48. Використання чисел з фіксованою комою в спеціалізованих ККС для обробки статистичної інформації та результатів вимірів.
49. Двійково – кодовані системи числення. Навести приклад.
50. Класифікація систем числення в комп’ютерній арифметиці, основні класи мікрооперацій в ККС
51. Алгоритм безпосереднього заміщення числа позиційної однорідної системи числення для переводу чисел з однієї основи до іншої.
52. Алгоритми переводу цілої частини числа та переводу дробів за рекурентними формулами. Навести приклад.
53. Алгоритми переводу цілих чисел із системи числення з основою k1 у систему з основою k2 при k1 > k2.
54. Різновиди й алгоритми виконання операцій зсуву комп’ютерної арифметики.
55. Алгоритми додавання і віднімання двійково – десяткових операндів. Алгоритми додавання і віднімання чисел з плаваючою комою. Навести приклад.
56. Основні алгоритми комп’ютерного множення в прямому коді та множення з округленням.
57. Основні алгоритми комп'ютерного множення в прямому коді. Навести приклад.

58. Алгоритми комп’ютерного множення в доповняльному коді. Навести приклад.
59. Похибки подання чисел і арифметичних операцій.
59. Алгоритми і похибки комп’ютерного округлення
60. Аналіз точності обчислень на ПК.

Частина 5. Практичні завдання

Завдання 1(33). Скласти цифрову діаграму множення чисел X = 55/64 і Y = 3 7/64 за алгоритмом послідовного перетворення цифр множника. У двійковому поданні X = 0,110111, Y = 0,100101, а z^*_0 відповідає вмісту двох знакових розрядів PZ.

При досить великому *n* середнє число додавань-віднімань, віднесене до одної цифри множника, складає 1/3, тобто

$$T_{MCP} = (n+1)(t_C + 0{,}3t_+).$$

Завдання 2(24). Скласти цифрову діаграму множення чисел *X=11/16* та *У-9/16, n = 4* (табл. 19), коли кожний рядок, за винятком тих, які позначені як “Результат додавання”, відповідає виконанню мікрооперацій, зазначених в операторних вершинах ГСА. При цьому порядок їх виконання визначається ГСА і конкретними значеннями операндів. ГСА першого алгоритму множення: множення з молодших розрядів множника зі зсувом суми часткових добутків вправо.

Завдання 3(20).

Скласти цифрову діаграму множення чисел X = 11/16 та Y = 10/16, n = 4 за третім основним алгоритмом (табл. 20) з початком зі старших розрядів множника, зсувом вліво суми часткових добутків з числом кроків множення ***n.***

Час множення за третім основним алгоритмом розраховується за формулою

$$T_{Ma} = (n-1)(t_C + t_+) + t_+.$$

Завдання 4(21).

Скласти цифрову діаграму множення чисел X = 7/16 та У = 15/16, n-4 за четвертим основним алгоритмом (табл. 21). Множення починати зі старших розрядів множника, сума часткових добутків нерухома, у процесі множення зсувається вправо множене, закінчується множення додаванням.

Час множення за четвертим основним алгоритмом складає

Завдання 5(34).

Скласти цифрову діаграму множення чисел X = 55/64 і Y = 3 7/64 за алгоритмом послідовного перетворення цифр множника (табл. 34). У двійковому поданні X = 0,110111, Y = 0,100101, а z^*_0 відповідає вмісту двох знакових розрядів PZ. Оцінити число одиниць і число додавань-віднімань після перетворення множника. Час множення за прискореним алгоритмом складає

$$T_{МСР} = (n+1)\left(0{,}56t_c + 0{,}33t_+\right).$$

Завдання 6(35).

Виконати за алгоритмом ділення з відновленням залишку ділення числа X = 7/16 на число Y = 13/16 при n = 4 (табл. 39). В двійковому поданні X = 00,0111, Y = 00,1101, (-Y)дк = 11,0011. Частка з недостачею дорівнює 1/2, тобто 0,1000, а частка з надлишком дорівнює 9/16, тобто, 0,1001.

У середньому, без врахування часу для одержання додаткового розряду для округлення, тривалість ділення складатиме

$$T_{Д1}=(n+1)(1{,}5t_+ + t_{зс}).$$

Завдання 7(36).

Скласти цифрову діаграму ділення X = 15/32 на Y = 24/32 по алгоритму на рис. 37, де в двійковому поданні X = 00,01111, Y = 00,11000, $(-Y)_{дк}$ = 11,01000) до одержання п`яти цифр результату після коми. Після четвертого додавання – віднімання, якщо утвориться нульовий залишок і всі наступні цифри частки будуть дорівнювати нулю, тому можна виконувати тільки зсуви для того, щоб цифри частки займали відповідні вагові позиції в регістрі PZ.

Завдання 8.

Скласти цифрові діаграми множення двійкових чисел X на Y і V на W за першим основним алгоритмом. (Примітка: у завданнях 13…20) у кожній парі операндів першим вказаний множник, другим – множене).

Завдання 9.

Скласти цифрові діаграми множення двійкових чисел X на V і Y на W за другим основним алгоритмом.

Завдання 10.

Скласти цифрові діаграми множення двійкових чисел X на W і Y на V за третім основним алгоритмом.

Завдання 11.

Скласти цифрові діаграми множення двійкових чисел V на X і W на Y за четвертим основним алгоритмом.

Завдання 12.

Скласти цифрові діаграми множення двійкових чисел X на –W і –Y на –V у доповняльному коді з обробкою знаків чисел і корекцією результату наприкінці операції.

Завдання 13.

Скласти цифрові діаграми прискореного множення двійкових чисел X на Y і V на W за алгоритмом з послідовним перетворенням цифр множника, вибираючи як множник із більшим числом одиниць.

Завдання 14.

Скласти цифрові діаграми прискореного множення двійкових чисел X на V і Y на W за алгоритмом з використанням зсувів на 1 і 2 розряди, вибираючи як множник операнд із більшим числом одиниць.

Завдання 15.

Переведення чисел з однієї системи числення в іншу.

Мета заняття: вивчити та засвоїти алгоритми переведення чисел з однієї системи числення в іншу.

Відпрацьовувані питання:

1. Переведення десяткових чисел в двійкову, вісімкову та шістнадцяткову системи числення.
2. Переведення двійкових, вісімкових та шістнадцяткових систем числення.
3. Переведення вісімкових та шістнадцяткових чисел в двійкові.
4. Переведення двійкових чисел в вісімкову та шістнадцяткову системи числения.

ДОДАТКИ

Додаток 1. Таблиця законів булевої алгебри.

Булеві операції мають властивості, які називаються законами булевої алгебри, перелічені в табл.1.1 та потребують доведення за допомогою таблиць істиності.

№ №пор.	Тотожність	Найменування законів
1	*2*	*3*
1	а) $x_1 \wedge x_2 = x_2 \wedge x_1$ або $x_1 x_2 = x_2 x_1$ б) $x_1 \vee x_2 = x_2 \vee x_1$ або $x_1 + x_2 = x_2 + x_1$	Комутативні закони для кон'юнкції та диз'юнкції
2	а) $(x_1 x_2) x_3 = x_1 (x_2 x_3)$ б) $(x_1 + x_2) + x_3 = x_1 + (x_2 + x_3)$	Асоціативні закони
3	а) $(x_1 + x_2) x_3 = x_1 x_3 + x_2 x_3$ б) $x_1 + x_2 x_3 = (x_1 + x_2)(x_1 + x_3)$	Дистрибутивні закони: а) кон'юнкція відносно диз'юнкції б) диз'юнкція відносно кон'юнкції
4	а) $xx = x$ б) $x + x = x$	Закони повторення (тавтології)
5	а) $x_1 (x_1 + x_2) = x_1$ б) $x_1 + x_1 x_2 = x_1$	Закони поглинання (абсорбції)
6	$x\bar{x} = 0; x + \bar{x} = 1$	Закони доповнення
7	а) $\overline{x_1 x_2} = \bar{x}_1 + \bar{x}_2$ б) $\overline{x_1 + x_2} = \bar{x}_1 \bar{x}_2$	Правила де Моргана
8	$\bar{\bar{x}} = x$	Закон подвійного заперечення
9	а) $x_1 x_2 + x_1 \bar{x}_2 = x_1$ б) $(x_1 + \bar{x}_2)(x_1 + x_2) = x_1$	Закони склеювання
10	а) $x \cdot 1 = x$ б) $x + 1 = 1$	Закони універсальної множини
11	а) $x \cdot 0 = 0$ б) $x + 0 = x$	Закони нульової множини
12	$x_1 + \bar{x}_1 x_2 = x_1 + x_2$	

Диз'юнкція та кон'юнкція мають властивості, аналогічні властивостям арифметичних операцій додавання і множення:

1. Властивість комутативності (переставний закон):

$x_1 + x_2 = x_2 + x_1$, $x_1 x_2 = x_2 x_1$.

2. Властивість асоціативності (сполучний закон):

$x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3$, $x_1(x_2 x_3) = (x_1 x_2) x_3$.

3. Властивість дистрибутивності (розподільний закон):

а) для кон'юнкції відносно диз'юнкції:

$x_1(x_2 + x_3) = x_1 x_2 + x_1 x_3$;

б) для диз'юнкції відносно кон'юнкції:

$x_1 + x_2 x_3 = (x_1 + x_2) \& (x_1 + x_3)$.

Властивість дистрибутивності фактично означає правила розкриття дужок і взяття в них логічних виразів. Правильність наведених властивостей доводиться за допомогою викладених вище законів.

Функція додавання за модулем 2 (нерівносильність) має такі властивості:

1. Комутативність (сполучний закон) $x_1 \oplus x_2 = x_2 \oplus x_1$.

2. Асоціативність (переставний закон) $x_1 \oplus (x_2 \oplus x_3) = (x_1 \oplus x_2) \oplus x_3$.

3. Дистрибутивність (розподільний закон) $x_1(x_2 \oplus x_3) = (x_1 x_2) \oplus (x_1 x_3)$.

4. $x \oplus x = 0$; 5. $x \oplus \bar{x} = 1$; 6. $x \oplus 1 = \bar{x}$; 7. $x \oplus 0 = x$.

На основі властивостей можна вивести правила подання функцій І, АБО, НІ через функцію додавання за модулем 2 і навпаки:

$\bar{x}_1 = x_1 \oplus 1$;

$x_1 + x_2 = x_1 \oplus x_2 \oplus x_1 x_2$;

$x_1 x_2 = (x_1 \oplus x_2) \oplus (x_1 \oplus x_2)$.

Функція імплікації $(\to)$ – це функція, що виражається як $x_1 \to x_2 = \bar{x}_1 + x_2$. Для цієї функції справджуються такі властивості:

1. $x \to x = 1$;	3. $x \to 1 = 1$;	5. $0 \to x = 1$;
2. $x \to \bar{x} = \bar{x}$;	4. $x \to 0 = \bar{x}$;	6. $1 \to \bar{x} = x$.

Функції І, АБО, НІ через імплікацію виражаються таким чином:

$x_1 + x_2 = \bar{x}_1 \to x_2$;

$x_1 x_2 = \overline{x_1 \to x_2}$;

$\bar{x}_1 = x_1 \to 0$.

Функція Шеффера – це функція, що може бути виражена співвідношенням $x_1 / x_2 = \overline{x_1 x_2}$.

Для цієї функції справджуються такі властивості:

1. $x / x = \bar{x}$;	3. $x / \bar{x} = 1$;	5. $x / 0 = 1$;
2. $x / 1 = \bar{x}$;	4. $\bar{x} / 0 = 1$;	6. $\bar{x} / 1 = x$.

Для функції Шеффера справджується тільки властивість комутативності

$x_1 / x_2 = x_2 / x_1$

і не справджується властивість асоціативності:

$x_1 /(x_2 / x_3) \neq (x_1 / x_2) / x_3$.

З основних властивостей функції Шеффера можна дістати такі формули перетворення:

$$\begin{cases} x_1 x_2 = \overline{x_1 / x_2} = x_1 / x_2 / x_1 / x_2; \\ \bar{x} = x / x; \\ x_1 + x_2 = \overline{\bar{x}_1 \bar{x}_2} = \bar{x}_1 / \bar{x}_2 = x_1 / x_1 / x_2 / x_2. \end{cases}$$

Функція стрілка Пірса-Вебба – це функція, що описується виразом $x_1 \downarrow x_2 = \overline{x_1 + x_2} = \bar{x}_1 \bar{x}_2$

Для цієї функції легко показати справедливість таких аксіом:

1. $x \downarrow x = \bar{x}$;	3. $x \downarrow \bar{x} = 0$;
2. $x \downarrow 0 = x$;	4. $x \downarrow 1 = 0$.

На підставі цих аксіом можна встановити, що для функції стрілка Пірса-Вебба справджується тільки властивість комутативності: $x_1 \downarrow x_2 = x_2 \downarrow x_1$.

Функції І, АБО, НІ виражаються через функцію стрілка Пірса-Вебба таким чином:

$$\begin{cases} x_1 x_2 = (x_1 \downarrow x_1) \downarrow (x_2 \downarrow x_2); \\ x_1 + x_2 = (x_1 \downarrow x_2) \downarrow (x_1 \downarrow x_2); \\ \bar{x} = x \downarrow x. \end{cases}$$

Логічний добуток будь-якої кількості різних змінних (символів), що входять із запереченням або без нього, називається елементарною кон'юнкцією.

Звідси випливає, що $x_1 x_2 \bar{x}_3$, наприклад, є, а $x_1 x_2 + \bar{x}_3$ – не є елементарною кон'юнкцією.

Якщо функція $F = x_1 x_2 \ldots x_r$ – елементарна кон'юнкція, то r називається її рангом.

Якщо функцію подано формулою у вигляді диз'юнкції елементарних кон'юнкцій, то кажуть, що її подано диз'юнктивною нормальною формою (ДНФ).

Наприклад, нехай деяка функція $f(x_0, x_1, x_2, x_3)$ реалізована формулою в вигляді диз'юнкції елементарних кон'юнкцій

$F_1(x_0, x_1, x_2, x_3) = x_0 x_2 \bar{x}_3 + x_0 \bar{x}_1 \bar{x}_2 x_3 + x_0 x_3$.

Очевидно, що будь-яку логічну функцію можна представити у вигляді ДНФ, тому що ДНФ утворюється за допомогою системи операцій $\{\wedge, \vee, \neg\}$, яка є повною. Будь-яка логічна функція має не єдину ДНФ.

Досконалою диз'юнктивною нормальною формою (ДДНФ) формули, що містить n різних змінних, називається ДНФ, яка має такі властивості:

1) в ній немає однакових доданків;

2) жоден із доданків не містить двох однакових співмножників;

3) жоден із доданків не містить змінну разом з її запереченням;

4) в кожному окремому доданку є як співмножник або змінна або її заперечення для будь-якого $i = 1,2,\ldots,n$.

Будь-яка логічна функція, за виключенням константи «0», має одну ДДНФ і кілька ДНФ. Будь-яка ДНФ утворюється внаслідок більшого або меншого скорочення ДДНФ, причому від будь-якої ДНФ можна перейти до ДДНФ. Такий перехід називається «розгортанням».

Принцип «розгортання» ДНФ до вигляду ДДНФ деякої функції $f(x_1, x_2, ..., x_n)$, що залежить від n змінних $x_1, x_2, ..., x_n$.

полягає в наступному:

1. Виділити всі доданки (елементарні кон'юнкції), в яких менше за n співмножників.

2. Кожний з цих доданків помножити на співмножник ($x_i + \bar{x}_i$), який дорівнює одиниці, де x_i – відсутня в доданку змінна.

3. Повторювати п.2, доки всі доданки не задовольнятимуть умові наявності n співмножників.

Підкреслимо, що цей принцип отримання ДДНФ застосовний тільки до формул, які мають вигляд ДНФ.

Існує ще один спосіб отримання ДДНФ, причому він теж застосовний для будь-якого вигляду формули, яка реалізує деяку логічну функцію. Це спосіб побудови ДДНФ функції $f(x_1, x_2, ..., x_n)$, виходячи з її табличного подання. Він містить кроки:

1. Побудова таблиці істинності функції $f(x_1, x_2, ..., x_n)$.

2. Для кожного набору значень аргументів (кортежу), на якому $f(x_1, x_2, ..., x_n) = 1$, в ДДНФ додається елементарна кон'юнкція $\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_n$, де

$$\tilde{x}_i = \begin{cases} x_i, & \text{якщо в кортежі } x_i = 1; \\ \bar{x}_i, & \text{якщо в кортежі } x_i = 0, \ i = 1,2,...,n. \end{cases}$$

Тобто ДДНФ має стільки доданків у вигляді елементарних кон'юнкцій, скільки одиниць є в стовпці функції.

Додаток 2. Булеві функції однієї змінної

Загальна таблиця істинності (відповідності) для булевих функцій однієї змінної має вигляд табл. 2.1.

Таблиця 2.1

x	$y_1 = f_1(x)$	$y_2 = f_2(x)$	$y_3 = f_3(x)$	$y_4 = f_4(x)$
0	1	0	1	0
1	1	0	0	1

Функції y_1 та y_2 є функціями - константами: $f_1(x)$ – абсолютно істинна (константа одиниці); $f_2(x)$ – абсолютно хибна (константа нуль), $f_3(x)$ – логічне заперечення, або НІ, або інверсія x (читається як «не x», зображається як $\bar{x}$ або $\neg x$), це єдина нетривіальна функція; $f_4(x)$ – змінна x (повторює значення змінної x і тому збігається з нею).

Додаток 3. Елементарні функції алгебри логіки

У табл.2.2. подано 16 функцій двох змінних, з яких шість $f_{16}(a,b)=0$, $f_{11}(a,b)=a$, $f_{13}(a,b)=b$, $f_{12}(a,b)=\bar{a}$, $f_{14}(a,b)=\bar{b}$, $f_{15}(a,b)=1$, є константами або функціями одного аргументу. Інші 10 функцій залежать від двох аргументів і мають свої загальноприйняті позначення та назви, зазначені в табл. 2.2.

Таблиця 2.2

Позначення функції	Найменування функції	a			
		0	0	1	1
		b			
		0	1	0	1
1	2	3	4	5	6
$f_1 = a \wedge b = ab = a \,\&\, b$	Кон'юнкція (логічне множення)	0	0	0	1
$f_2 = a \vee b = a + b$	Диз'юнкція (логічне додавання)0	0	1	1	1
$f_3 = a \to b$	Імплікація (від a до b)	1	1	0	1
$f_4 = a \leftarrow b$	Обернена імплікація (від b до a)	1	0	1	1
$f_5 = a \sim b = a \Leftrightarrow b$	Рівносильність	1	0	0	1
$f_6 = a \not\sim b = a \oplus b$	Нерівносильність (додавання за модулем 2)	0	1	1	0
$f_7 = a \overline{\wedge} b = a / b$	Функція Шеффера (інверсія кон'юнкції)	1	1	1	0
$f_8 = a \nabla b = a \downarrow b$	Функція стрілка Пірса – Вебба (інверсія диз'юнкції)	1	0	0	0
$f_9 = a \overline{\to} b$	Інверсія імплікації (функція заборони за b)	0	0	1	0
$f_{10} = a \overline{\leftarrow} b$	Інверсія оберненої імплікації (функція заборони за a)	0	1	0	0
$f_{11} = a$	Повторення a (змінна a)	0	0	1	1
$f_{12} = \overline{a}$	Інверсія a	1	1	0	0
$f_{13} = b$	Повторення b (змінна b)	0	1	0	1
$f_{14} = \overline{b}$	Інверсія b	1	0	1	0
$f_{15} = 1$	Одинична функція (константа 1)	1	1	1	1
$f_{16} = 0$	Нульова функція (константа 0)	0	0	0	0

Розглянемо найважливіші функції від двох змінних.

Функція $f_1(a,b)$ – **кон'юнкція** (логічне множення) істинна тоді і тільки тоді, коли a і b – істинні. Кон'юнкцію називають також функцією І; умовно її позначають $f_1(a,b) = ab = a \,\&\, b = a \wedge b$.

Функція $f_2(a,b)$ – **диз'юнкція** (логічне додавання) істинна тоді і тільки тоді, коли істинними є або a, або b, або обидві змінні. Диз'юнкцію часто називають також функцією АБО й умовно позначають так: $f_2(a,b) = a + b = a \vee b$.

Від диз'юнкції потрібно відрізняти функцію $f_6(a,b)$ яка називається функцією **додавання за модулем 2** (функцією нерівнозначності або

різнойменності) і є істинною тоді і тільки тоді, коли істинні або a, або b окремо. Умовне позначення цієї функції $f_6(a,b) = a \oplus b = a \approx b$.

Додаток 4

Задачі комп'ютерної логіки

1. У XIX столітті один вчитель задав своїм учням обчислити суму всіх цілих чисел від одиниці до ста. Комп'ютерів і калькуляторів тоді ще не було, і учні почали сумлінно складати числа. І тільки один учень знайшов правильну відповідь всього за кілька секунд. Ним виявився Карл Фрідріх Гаус – майбутній великий математик. Як він це зробив?

2. При виданні книги треба було 2 775 цифр того, щоб пронумерувати її сторінки. Скільки страниць в книзі?

3. Назвіть два числа, у яких кількість цифр дорівнює кількості букв, що складають назву кожного з цих чисел.

4. Яким чином можна з максимальною точністю виміряти діаметр тонкого дроту, маючи в наявності тільки вимірювальну лінійку і олівець?

5. Як ви думаєте, якщо півстоліття розділити на половину, то скільки в підсумку вийде?

6. Як ви думаєте, який знак слід поставити між 0 і 1, щоб було отримано число більше 0, але менше 1?

7. Чи можете ви записати число 1000 при допомозі тільки восьми вісімок і арифметичних знаків суми?

8. Як так могло статися, що половина числа 12 стало одно 7?

9. Деяке число було збільшено на 25 відсотків. На скільки відсотків потрібно зменшити нове число, щоб отримати початкове число?

10. Деякий число було зменшено на 25 відсотків. На скільки відсотків треба збільшити результат зменшення, щоб отримати початкове число?

11. Всі знають, що 2 в квадраті = 4, 10 в квадраті = 100, половина в квадраті = 1/4, третина в квадраті = 1/9. Як ви думаєте, чому дорівнює кут в квадраті?

12. Є число 188. Як ви думаєте, як його розділити навпіл так, щоб в результаті вийшла одиниця?

13. Чи можете ви назвати десятизначні число, що складається з десяти різних цифр (цифри: 0,1,2,3,4,5,6,7,8,9), яке б при множенні на 2 давало інше десятизначні число, також що складається з десяти різних цифр.

14. Як ви думаєте, на скільки сума всіх парних чисел наявних в послідовності від 1 до 100 більше суми всіх непарних чисел цієї ж послідовності?

15. У мові запитів пошукового сервера для позначення логічної операції «АБО» використовується символ «|», а для логічної операції «І» – символ «&». У таблиці наведено запити і кількість знайдених по ним сторінок деякого сегмента мережі Інтернет.

Запит	Знайдено сторінок (в тисячах)
Фрегат | Есмінець	3400
Фрегат & Есмінець	900
Фрегат	2100

Яка кількість сторінок (в тисячах) буде знайдено за запитом Есмінець? Вважається, що всі запити виконувалися практично одночасно, так що набір сторінок, що містять всі шукані слова, що не змінювався за час виконання запитів.

16. У таблиці наведено запити до пошукового сервера. Розмістіть номера запитів в порядку зростання кількості сторінок, які знайде пошуковий сервер по кожному запиту. Для позначення логічної операції "АБО" в запиті використовується символ |, а для логічної операції "І" - &.

№	Запит
1	принтери & сканери & продажа
2	принтери & продажа
3	принтери | продажа
4	принтери | сканери | продажа.

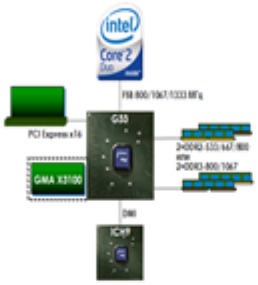
17. Аркуш паперу прямокутної форми перегнули навпіл шість разів. У середній частині цього складеного листа просвердлили наскрізь два отвори. Скільки отворів можна буде нарахувати на аркуші після його розгортання в початкове положення?

18. Багато хто знає, що один квадратний метр складається з одного мільйона квадратних міліметрів (1000x1000 = 1000000). Але ось знайшовся один хлопчик, який ніяк не міг в це повірити. "Ніколи не повірю, що в цьому аркуші паперу вміститися мільйон квадратних міліметрів, поки особисто сам не порахую всі клітини!" – Говорив він, тримаючи в руках квадратний метр спеціальної креслярського паперу, вже розкресленій на міліметрові клітини. І ось одним рано вранці він прокинувся і почав прискіпливо перераховувати на папері клітини, сумлінно відзначаючи олівцем кожну з порахованих клітин. Як Ви вважаєте – чи зміг він в цей день переконатися в тому, що квадратний метр дійсно містить в собі мільйон квадратних міліметрів?

19. На питання, хто з трьох учнів вивчав логіку, була отримана відповідь: «Якщо вивчав перший, то вивчав і другий, але не так, що якщо вивчав третій, то вивчав і другий». Хто з учнів вивчав логіку?

Додаток 5. Таблиця компонентів сучасного комп’ютера

№	Тип пристрою.	Модель	Характеристики
1	Монітор	LG Flatron W1943C 	• ЖК-монітор з діагоналлю 18.5" • тип ЖК-матриці TFT TN • розрішення 1366x768 (16:9) • підключення: VGA • яскравість 200 кд/м2 • контрастність 600:1 • час відгуку 5 мс
2	Системна плата	ASUS P5KPL-AM SE 	• материнська плата форм-фактору microATX • сокет LGA775 • чипсет Intel G31 • 2 слоти DDR2 DIMM, 667-1066МГц • відеоадаптер Intel GMA 3100 • роз’еми SATA: 3 Гбіт/с - 2
3	ОЗП	DDR2 2GB 800 MHz Transcend 	Призначення: Для настільного комп'ютера Об'єм пам'яті 2 GB Тип пам'яті DDR 2 Частота шини 800 MHz Пропускна можливість шини PC2-6400 Латентність (таймінги) CL6 Кількість модулів в комплекті (Kit: 1x2GB) Охолодження немає

4	Центральний процесор	DualCore Intel Pentium E5300, 2633 MHz (13 x 203) 	Сімейство процесора Intel Pentium Dual-Core Тип роз'єму Socket 775 Кількість контактів 775 Внутрішня тактова частота 2600 МГц Частота шини даних 800 МГц Кількість ядер 2 Об'єм кеш пам'яті 2 рівня 2 МБ Напруга живлення 0.85 В – 1.3625 В Гарантія 36 месяцев
5	Чіпсет	**Intel G33 Express** 	• підтримка «нових» процесорів сімейств Celeron та Pentium, а також процесорів Core 2 Duo/Quad із частотою системної шини 800/1066 МГц, включаючи майбутні моделі з частотою системної шини 1333 МГц; • двухканальний контролер пам'яті DDR2-533/667/800 або DDR3-800/1067 з підтримкою до 4 модулів DIMM сумарним об'ємом до 8 ГБ (без ECC) та технологіями Fast Memory Access и Flex Memory; • графічний інтерфейс PCI Express x16; • інтегроване графічне ядро GMA X3100 з підтримкою технології Clear Video; • Шина DMI (з пропускною можливістю ~2 ГБ/с) до нового південного мосту ICH9/R/DH.
6	Відеокарта	Intel(R) GMA 3100	Виконавець: Intel Серія: Graphics Media Accelerator (GMA) X3100 Код: Crestline Потоки: 8 – unified Тактова частота: 500* МГц Частота шейдеров: 500* МГц Тип пам'яті: mit Hauptspeicher geteilt (shared), max 384 MB Максимум пам'яті: 384 Мб DirectX: DirectX 10, Shader 3.0 Енергоспоживання: 13.5 Вт
7	Жорсткий диск	WDC WD2500AAJS-00YZCA0 	Лінійка WD Caviar Blue Тип HDD Призначення для настільного комп'ютера Форм-фактор HDD 3.5" Характеристики накопичувача Об'єм 250 Гб Об'єм буферної пам'яті 8 Мб Швидкість обертання 7200 rpm Інтерфейс Підключення SATA 3Gb/s Зовнішня швидкість передачі даних 300 Мб/с Внутрішня швидкість передачі даних 972 Мбіт/с

У центрі сучасного центрального мікропроцесора (CPU – скор. з англ. Central processing unit – центральний обчислювальний пристрій) находиться ядро (core) – кристал кремнію площею приблизно один квадратний сантиметр, на якому за допомогою мікроскопічних логічних елементів реалізована принципова схема процесора, так звана архітектура (chip architecture). Ядро пов'язано з іншою частиною чіпа (званої «упаковка», CPU Package) за технологією «фліп-чіп» (flip-chip, flip-chip bonding – перевернуте ядро, кріплення методом перевернутого кристала). Ця технологія отримала таку назву тому, що звернена назовні – видима – частина ядра насправді є його «дном», – щоб забезпечити прямий контакт з радіатором кулера для кращої тепловіддачі. Зі зворотнього (невидимого) боку знаходиться сам «інтерфейс» - з'єднання кристала і упаковки. З'єднання ядра процесора з упаковкою виконано за допомогою стовпчикових виводів (Solder Bumps). Ядро розташоване на текстолітовій основі, по якій проходять контактні доріжки до «ніжок» (контактних площадок), це все залито термічним інтерфейсом і закрито захисною металевою кришкою.

Багатоядерний процесор – це центральний мікропроцесор, що містить 2 і більше обчислювальних ядра на одному процесорному кристалі або в одному корпусі. Для чого потрібна багатоядерність? Перший (природно, одноядерний!) мікропроцесор Intel 4004 був представлений 15 листопада 1971 р корпорацією Intel. Він містив 2300 транзисторів, працював на тактовій частоті 108 кГц і коштував $300. Вимоги до обчислювальної потужності центрального процесора постійно росли і продовжують рости. Але якщо раніше виробникам процесорів доводилося постійно підлаштовуватися під поточні нагальні (вічно зростаючі!) запити користувачів ПК, то тепер чипмейкери йдуть з великим випередженням! Довгий час підвищення продуктивності традиційних одноядерних процесорів в основному відбувалося за рахунок послідовного збільшення тактової частоти (близько 80% продуктивності процесора визначала саме тактова частота) з одночасним збільшенням кількості транзисторів на одному кристалі. Однак подальше підвищення тактової частоти (при тактовій частоті понад 3,8 ГГц чіпи просто перегріваються!) впирається в ряд фундаментальних фізичних бар'єрів (оскільки технологічний процес майже впритул наблизився до розмірів атома: сьогодні процесори випускаються по 45-нм технології, а розміри атома кремнію – приблизно 0,543 нм):

• по-перше, зі зменшенням розмірів кристала і з підвищенням тактової частоти зростає струм витоку транзисторів. Це веде до підвищення споживаної потужності і збільшення викиду тепла;

• ринок програмного забезпечення повинен бути забезпечений програмами, здатними ефективно розбивати алгоритм розгалуження команд на парне (для процесорів з парною кількістю ядер) або на непарне (для процесорів з непарною кількістю ядер) кількість потоків.

Переваги багатоядерних процесорів

• можливість розподіляти роботу програм, наприклад, основних завдань додатків і фонових завдань операційної системи, за кількома ядер;

• збільшення швидкості роботи програм;

• процеси, що вимагають інтенсивних обчислень, протікають набагато швидше;

• більш ефективне використання вимогливих до обчислювальних ресурсів мультимедійних додатків (наприклад, відеоредакторів);

• зниження енергоспоживання;

• робота користувача ПК стає більш комфортною.

Недоліки багатоядерних процесорів

• зросла собівартість виробництва багатоядерних процесорів (в порівнянні з одноядерними), що змушує чипмейкерів збільшувати їх вартість, а це частково стримує попит;

• так як з оперативною пам'яттю одночасно працюють відразу два і більше ядра, необхідно «навчити» їх працювати без конфліктів;

• зросле енергоспоживання, що вимагає застосування потужних схем живлення;

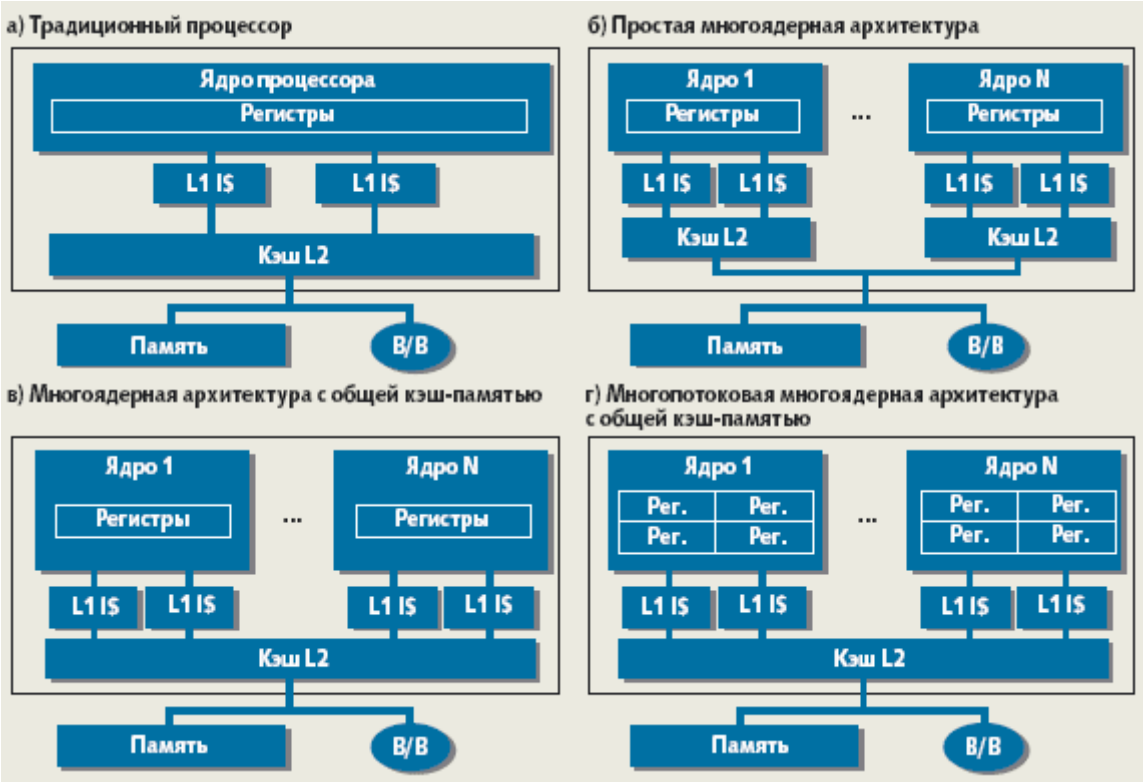
• потрібно більш потужна система охолодження;

• кількість оптимізованого під багатоядерність програмного забезпечення мізерно мало (більшість програм розраховані на роботу в класичному одноядерному режимі, тому вони просто не можуть задіяти обчислювальну потужність додаткових ядер) (зараз не актуально);

• операційні системи, що підтримують багатоядерні процесори (наприклад, Windows XP SP2 і вище) використовують обчислювальні ресурси додаткових ядер для власних системних потреб.

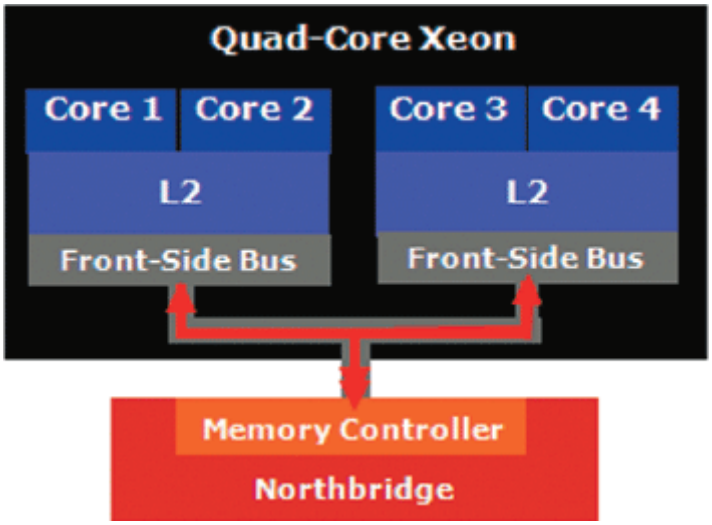
Слід визнати, що в даний час багатоядерні процесори використовуються вкрай неефективно. Окрім того, на практиці n-ядерні процесори не виробляють обчислення в n разів швидше одноядерних: хоча приріст швидкодії і виявляється значним, але при цьому він багато в чому залежить від типу програми. У програм, які не розраховані на роботу з багатоядерними процесорами, швидкодія збільшується всього на 5%. А ось оптимізовані під багатоядерні процесори програми працюють швидше вже на 50%.

Схеми роботи багатоядерних процесорів:



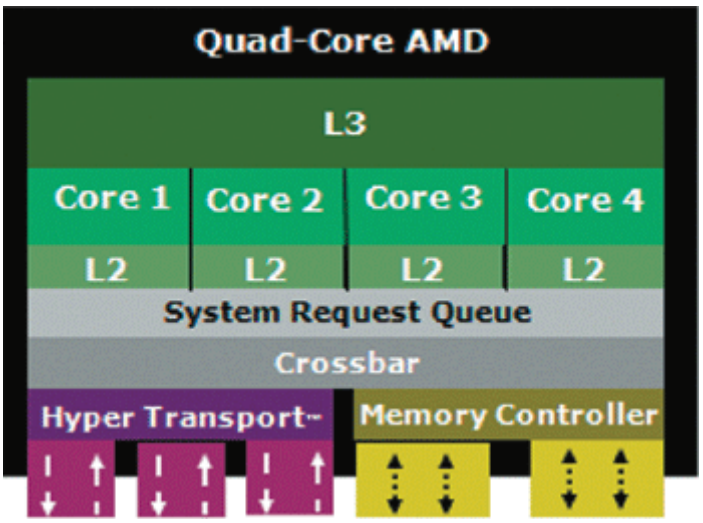
Приклади багатоядерніх процесорів:

Чотірьохядерний процесор Intel представляє собою два двохядерні процесори, що розміщені на одному кристалі. Іншими словами, розроблений фахівцями Intel чотирьохядерний пристрій Xeon є не власне чотирьохядерний процесор, а здвоєний двоядерний процесор. Але хоча така архітектура дозволила компанії Intel швидко довести виріб до ринку, запропонована структура не є оптимальною. Коли процесори, розміщені на окремих ядрах, обмінюються даними, ці дані доводиться пересилати по системній шині і через контролер пам'яті, а це не concept (не самий кращий) спосіб. Крім того, як і в більш ранніх структурах Intel, при подібному підході загальна швидкодія системи ставиться в залежність від швидкодії системної шини. Але, незважаючи на зазначені недоліки і завдяки вдосконаленням в мікроархітектурі Intel Core, а також підключення додаткових процесорів, чотириядерні кристали Intel були найшвидшими з представлених на ринку x64-сумісних процесорів.



Розробники процесора Barcelona від AMD, що готується до випуску, пішли іншим шляхом: вони розмістили на одній напівпровідниковій пластині чотири незалежних процесора. У чотирьохядерному кристалі AMD, як і в випущених раніше моделях Opteron, реалізована фірмова архітектура Direct Connect Architecture. При виготовленні процесорів Barcelona буде використовуватися технологічний процес 65 нм. Передбачено випуск версій потужністю 68, 95 і 120 Вт. Відповідно до чотирьохядерної природи моделі всі чотири ядра функціонують незалежно один від одного. Теоретично справді чотирьохядерна модель забезпечує також більш ефективне енергоспоживання, оскільки кожне ядро може підвищувати і знижувати свою тактову частоту відповідно до навантаження.

У конструкцію процесора Barcelona був внесений ряд інших важливих удосконалень. Він забезпечує 128-розрядну обробку даних з плаваючою комою і оснащений новим кешем третього рівня ємністю 2 Мбайт, використовуваним усіма ядрами. Оскільки кожен процесор виконує протягом одного такту більший обсяг роботи, підвищення його продуктивності (що становить близько 15%) забезпечує загальне підвищення швидкодії приблизно на 40%. Важлива особливість чотириядерних процесорів AMD полягає в тому, що на рівні гнізд вони сумісні з існуючими двоядерними процесорами Socket F. Отже, сучасні двоядерні системи, побудовані з використанням технології AMD Socket F, можуть бути модернізовані до рівня чотирьох ядер шляхом звичайної заміни процесорів і подальшої модернізації системи BIOS. Модель Barcelona повинна перевершити чотириядерні процесори Intel і по масштабованості. Кожне ядро чотирьохядерної пластини процесора Barcelona в майбутньому теоретично може бути модернізовано до рівня двоядерного кристала. По суті, це означає можливість розміщення на одній чотирьохядерній пластині чотирьох двоядерних процесорів.



Додаток 6

Приклади переведення чисел та математичних операцій над числами різних систем числення

Переведемо 2016_{10} в двійкову систему ось як:

Ціла частина числа отримується діленням на основу нової

2016	2										
-2016	1008	2									
0	-1008	504	2								
	0	-504	252	2							
		0	-252	126	2						
			0	-126	63	2					
				0	-62	31	2				
					1	-30	15	2			
						1	-14	7	2		
							1	-6	3	2	
								1	-2	**1**	
									1		

Отримано:$2016_{10} = 11111100000_2$

Перевірка;

$11111100000_2 = 1\cdot2^{10}+1\cdot2^9+1\cdot2^8+1\cdot2^7+1\cdot2^6+1\cdot2^5+0\cdot2^4+0\cdot2^3+0\cdot2^2+0\cdot2^1+0\cdot2^0 =$
$1024+512+256+128+64+32+0+0+0+0+0 = 2016_{10}$

Отримано: 2016_{10}

Ціла частина числа отримується діленням на основу нової системи числення, до якої виконується перехід.

Переведемо 2016_{10} в вісімкову систему ось як основа нової

2016	8		
-2016	252	8	
0	-248	31	8
	4	-24	**3**
		7	

Отримано: $2016_{10} = 3740_8$

Перевірка:

$3740_8 = 3 \cdot 8^3 + 7 \cdot 8^2 + 4 \cdot 8^1 + 0 \cdot 8^0 = 1536 + 448 + 32 + 0 = 2016_{10}$

Переведемо 2016_{10} в шістнадцатирічну систему ось як:

Ціла частина числа отримується діленням на основу нової системи числення

2016	16	
-2016	126	16
0	-112	7
	14=E	

Отримано: $2016_{10} = 7E0_{16}$

Перевірка:

$7E0_{16} = 7 \cdot 16^2 + 14 \cdot 16^1 + 0 \cdot 16^0 = 1792 + 224 + 0 = 2016_{10}$

Отримано: 2016_{10}

```
  23540
      3740
1  3  640
-------------
1  4  511200
```

Отримано: $3740_8 * 3154_8 = 14511200_8$

Виконаємо множення $66C_{16} * 7E01_{16}$

```
x      66C
     7E01
   -------
      66C
    000
+ 59E8
  2CF4
 -------
 3292E6C
```

Отримано: $66C_{16} * 7E01_{16} = 3292E6C_{16}$

Складання:

Виконаємо складання $11111100000_2 + 11001101100_2$ (**1644**)

```
. . . . . .
 11111100000
+11001101100
------------
111001001100
```

Отримано: $11111100000_2 + 11001101100_2 = 111001001100_2$

Виконаємо складання $\mathbf{3740_8 + 3154_8}$

```
  . .
 3740
+3154
------
 7114
```

Отримано: $3740_8 + 3154_8 = 7114_8$

Виконаємо складання $\mathbf{66C_{16} + 7E01_{16}}$

```
  .
  66C
+7E01
------
 846D
```

Отримано: $66C_{16} + 7E01_{16} = 846D_{16}$

Множення:

Виконаємо множення $11111100000_2 * 11001101100_2$

```
                 11111100000
x
                 11001101100
  ------------------------------
                 00000000000
                00000000000
               11111100000
              11111100000
             00000000000
+           11111100000
           11111100000
          00000000000
         00000000000
        11111100000
       11111100000
  ------------------------------
  1100101001001010000000
```

Отримано: $11111100000_2 * 11001101100_2 = 1100101001001010000000_2$

Виконаємо множення $3740_8 * 3154_8$

```
      3740
x
      3154
  ------------
+   17600
```

Виконаємо множення $3740_8 * 3154_8$

```
x      3740
       3154
     -------
      17600
+    23540
     3740
   13640
  --------
  14511200
```

Отримано: $3740_8 * 3154_8 = 14511200_8$

Виконаємо множення $66C_{16} * 7E01_{16}$

```
x      66C
       7E01
     -------
       66C
+      000
     59E8
   2CF4
  --------
  3292E6C
```

Отримано: $66C_{16} * 7E01_{16} = 3292E6C_{16}$

Ділення:

Виконаємо ділення $11111100000_2 \div 11001101100_2$

```
 11111100000 | 11001101100
-11001101100 | 1.00111001111
  101110100000
-   11001101100
  101001101000
-   11001101100
    11111111000
-   11001101100
    110001100000
-     11001101100
      101111101000
-     11001101100
      101011111000
-     11001101100
      100100011000
-     11001101100
       1010101100
```

Отримано: $11111100000_2 \div 11001101100_2 = 1.00111001111_2$

Виконаємо ділення $3740_8 \div 3154_8$

```
 3740 | 3154
-3154 | 1.16366520263
 5640
-3154
 24640
-23210
  14300
 -11504
   25740
  -23210
    25300
   -23210
     20700
    -20034
       6440
      -6330
        11000
       -6330
         24500
        -23210
          12700
         -11504
            1174
```

Отримано: $3740_8 \div 3154_8 = 1.16366520263_8$

```
 66C  | 7E01
-666 | 0.0d0c1640d6d9b
  5F300
 -5E80C
    AF40
   -7E01
    31 3F0
   -2F406
     1FEA0
    -1F804
       69C00
      -6660D
        35F30
       -2F406
         6B2A0
        -6660D
          4C930
         -46E09
           5B270
          -56A0B
            48650
```

Отримано: $66C_{16} \div 7E01_{16} = 0.0d0c1640d6d9b_{16}$

Література

1. Абрамов В.О. Архітектура електронно-обчислювальних машин: навч. посіб. – К.: КМПУ імені Б.Д. Грінченка, 2007. – 84 с.
2. Абрамов В.О. Фізичні основи комп'ютерних систем: навч. посіб. – К.:КМПУ ім. Б.Д. Грінченка, 2007. – 140 с.
3. Савельев А.Я. Арифметические и логические основи цифрових автоматов: уч. – М.: Высшая школа, 1980. – 272 с.
4. Жабин В.И. и др. Логические основы и схемотехника ЭВМ: Практикум. – К.: ВЕК+, 1999. – 128 с.
5. Самофалов К.Г., Корнейчук В.И., Тарасенко В.П. Цифровые ЭВМ. Теория и проэктирование. – К.: Вища шк., 1989. – 424 с.
6. Самофалов К.Г., Корнейчук В.И., Тарасенко В.П., Жабин В.И. Цифровые ЭВМ. Практикум. – К.: Вища шк., 1990. – 215 с
7. Чегренець В.М. Обчислювальні методи: навч. посіб. – К.: КМПУ ім. Б.Д. Грінченка, 2009. – 86 с.
8. Абрамов В.О., Чегренець В.М. Комп'ютерні мережі: навч. посіб. – К.: ун-т ім. Б. Грінченка, 2010. – 128 с.
9. Кузьмін Е.О., Петров В.К., Чегренець В.М. та ін. Обробка та аналіз статистичних і експертних даних: навч. посіб. – К.: НУОУ, 2011. – 122 с.
10. Чегренець В.М. Операційні системи та системне програмування: Навч. посіб. – К.: ун-т ім. Бориса Грінченка, 2011. – 164 с.
11. Кузьмін Е.О., Петров В.К., Чегренець В.М. та ін. Інформатизація штабів: навч. посіб. – К.: НУОУ, 2012.– 235 с.
12. Абрамов В.О., Чегренець В.М. Основи баз даних та робота в СУБД Access: навч. посіб. – К.: ун-т ім. Б. Грінченка, 2013. – 120 с.
13. Корнійчук В.І., Тарасенко В.П., Тарасенко-Клятченко О.В. Основи комп'ютерної арифметики. – К.: Корнійчук, 2014. – 170 с.
14. Основи інформаційно-аналітичного забезпечення органів військового управління: навч. посіб. / Кузьмін Е.О., Петров В. К., Чегренець В.М.– К.: Київ, НУОУ ім. Івана Черняховського, 2014. – 104 с
15. Чегренець В.М. Аналіз точності обчислень у сучасних телекомунікаційних системах із використанням ККС. Матеріали міжнародної науково-технічної конференції «Сучасні інформаційно-телекомунікаційні технології», Т.3. Розвиток інформаційних технологій. – Київ. 17-20 листопада 2015р. – С.102-103.
16. В.М.Чегренець, Н.В.Руденко. Аналіз точності обчислень в сучасних телекомунікаційних системах з використанням комп'ютерних технологій. Наукові записки Українського науково-дослідного інституту зв'язку. – 2016. – №1(41). – Київ. – С. 58-64.
17. В.М.Чегренець, В.М. Ахрамович. Хмарні технології та можливості їх використання в комп'ютерних телекомунікаційних системах. Проблеми

інформатизації. Тези доповідей четвертої міжнародної науково-технічної конференції 11-12 квітня 2016 року.

18. Чегренець В.М.. Ахрамович В.М.,Руденко Н.В. Хмарні технології та можливості їх використання в комп'ютерних телекомунікаційних системах. Наукові записки Українського науково-дослідного інституту зв'язку, 2016. – № 2 (42). – С. 121-129.

19. Учебник / Под ред. проф. Н.В.Макаровой. – М.: Финансы и статистика, 2001.

20. Информатика: Базовый курс / С.В.Симонович и др. – СПб.: Питер, 2013.

Предметний покажчик