Київський національний університет імені Тараса Шевченка

Кононов М.В.

Комп’ютерні технології

Посібник для студентів спеціальності
172 «Телекомунікації та радіотехніка»
факультету радіофізики електроніки
та комп’ютерних систем

Київ – 2019

УДК 004

Комп’ютерні технології. Посібник для студентів спеціальності 172 «Телекомунікації та радіотехніка» факультету радіофізики електроніки та комп’ютерних систем / Кононов М.В. – Київ: ФРЕКС Київського національного університету імені Тараса Шевченка, 2019. – 281 с.

Посібник складений у відповідності з навчальною програмою дисципліни "Комп’ютерні технології", що викладається для студентів спеціальності 172 «Телекомунікації та радіотехніка» факультету радіофізики електроніки та комп’ютерних систем.

Метою дисципліни є ознайомлення студентів з базовими принципами обробки, накопичення, використання інформації, сучасними поглядами на реалізацію програмних та апаратних засобів для роботи з інформацією, в тому числі для автоматизації вимірювань, моделювання, потреб комп’ютерної та інженерної графіки; засвоєння елементів алгоритмізації засобів автоматичного керування, САПР.

Рецензенти:

завідувач кафедри фізичної електроніки Київського національного університету імені Тараса Шевченка, професор, доктор фізико-математичних наук Веклич А.М.,

доцент кафедри математики та теоретичної радіофізики Київського національного університету імені Тараса Шевченка, кандидат фізико-математичних наук Нетреба А.В.

Ухвалено Вченою радою
факультету радіофізики електроніки та комп’ютерних систем
Київського національного університету імені Тараса Шевченка
(протокол № 2 від 15.10.2019)

Зміст

Вступ

Спеціальність «Телекомунікації та радіотехніка» орієнтується на підготовку фахівців у галузі передачі інформації на певну відстань за допомогою електромагнітних сигналів.

Одразу зауважимо, що вже з самого початку доведеться використати деякі терміни (наприклад, вже використаний термін «сигнал»), детальне тлумачення яких буде описане у наступних главах. Поки що будемо спиратись на загальну ерудицію та знання, отримані раніше при вивченні інформатики та фізики.

Фактичним часом народження цієї галузі можна вважати появу телефонії та радіо, забов'язаних певним успіхам у дослідженні електрики, бо саме електромагнітна природа сигналів виявилась для потреб передачі на відстань значно ефективнішою за сигнали іншої фізичної природи. До речі, другою назвою телекомунікацій є саме «електрозв'язок». Доволі довгий час телекомунікаційні технології обмежувались аналоговими сигналами, але з часом потреба значного ускладнення структури аналогового кодування інформації, що передається, певною мірою стала суттєвою перешкодою для подальшого збільшення ефективності та функціональності електрозв'язку.

Впровадження цифрових технологій, пов'язане з появою комп'ютерних інформаційних мереж, дало новий потужний поштовх. Наразі існує стала тенденція витіснення аналогових технології в галузі телекомунікацій цифровими. Але цифрові технології вимагають зовсім інших методів роботи з інформацією і є невідривними від програмної обробки даних та програмованих пристроїв. Відповідно сучасний фахівець у галузі телекомунікацій та радіотехніки повинен мати досить глибокі знання у цифровій обробці сигналів, в тому числі в експлуатації та розробці певних складових програмованих пристроїв, таких як комп'ютери та мікроконтролери, які є основою переважної більшості сучасного радіоелектронного обладнання.

Телекомунікаційні технології від свого зародження активно використовують такі види представлення інформації, як звук та зображення (в тому числі динамічне, тобто змінюване у часі). Основним засобом роботи з цими видами інформації наразі став комп'ютер в усіх його формах, оскільки і смартфон, і сучасний телевізор, і цифровий побутовий радіоприймач або плеєр фактично є різновидами комп'ютерів. Саме тому вивчення комп'ютерних технологій для зазначеної спеціальності є фаховою потребою, а вивчення особливостей комп'ютерної обробки інформації, зокрема

зображення та звуку є необхідним для правильного розуміння тенденцій розвитку телекомунікаційної галузі.

Відповідно, курс «комп'ютерні технології» охоплює програмну обробку інформації, в тому числі особливості кодування, найпоширеніші методи обробки, тенденції розвитку апаратних засобів. Він спирається на загальний базис, отриманий у шкільному курсі «інформатика» і використовує знання, отримані у курсі «програмування».

Спеціальність «Телекомунікації та радіотехніка» є одним із різновидів інженерної підготовки, тобто забезпечує навчання фахівців в області інженерної справи або інженерії. Є багато доволі різних визначень цього терміну, але на наш погляд найбільш вдалим є визначення Американської Ради інженерів з професійного розвитку (American Engineers 'Council for Professional Development):

Інженерія – творче застосування наукових принципів для проектування або розробки структур, машин, апаратури, виробничих процесів, або робота по використанню їх окремо або в комбінації; конструювання або управління тим же самим з повним знанням їх дизайну; пророкування їх поведінки під певними експлуатаційними режимами.

Інженерна справа може бути формально поділена на

- конструювання;
- забезпечення виробництва;
- експлуатацію.

Конструювання передбачає саме розробку нового обладнання і подальше забезпечення його виготовлення та експлуатації. Зрозуміло, що конструювання спирається на створення певних об'ємів інформації (у вигляді документації – текстів та креслень) та підготовку блоку інформації для потреб виробника та користувача.

Забезпечення виробництва спирається на використання реалізованої при конструюванні документацію, яку необхідно в процесі розробки проекту зробити достатньою для розгортання усього циклу виробництва.

Експлуатація вимагає використання значно простішої документації, оскільки передбачає тільки кваліфіковане використання результатів проекту та (за потреби) ремонт реалізованого в ньому виробу, що явно є простішим за розробку та виготовлення.

Тисячі років основою інженерії був механічний рух, тобто технічна механіка та забезпечення достатньої міцності в умовах великих навантажень. Трохи пізніше до механіки додалась теплотехніка, в тому числі для створення нових джерел механічної енергії – теплових двигунів. Мабуть вже на цьому етапі почала

виникати потреба спеціалізації інженерії, але поки що не дуже мотивована.

Докорінний переворот в інженерії можна віднести до двадцятого сторіччя, оскільки саме в цей період до господарської діяльності людини потрапили надбання хімії, а також такої галузі фізики, як електрика та магнетизм. Найбільш могутнім поштовхом для розвитку інженерії став електричний струм як зручний засіб доставки енергії на велику відстань, а трохи пізніше і носій інформації. Поява пристроїв, які дозволили використати електричні сигнали для передачі та запису інформації, призвели до появи таких галузей, як електроніка та радіотехніка, які є нерозривно зв’язаними між собою. Радіотехніка за визначенням головним чином пов’язана із використанням електромагнітних коливань та хвиль (традиційно радіодіапазону, але в останній час фактично відбулась змичка з оптичним діапазоном), методів генерації, підсилення, випромінювання і прийому хвиль, обробки сигналів. Сучасна радіотехніка включає наступні основні розділи:

- радіопередавальні та приймальні пристрої;
- радіолокація та радіонавігація;
- зв'язкові системи та телебачення;
- системи радіоелектронної протидії.

Практично усі розділи з наведеного переліку пов’язані з передачею і обробкою інформації і за сучасними тенденціями, як вже було зазначено, активно використовують цифрові методи. Саме тому розуміння особливостей обробки і накопичення інформації, особливо цифровими методами, є нагальною потребою сучасного інженера-радіотехніка.

Електроніка як галузь, що відповідає за елементну базу радіотехніки, є також надзвичайно важливою для кваліфікованого фахівця з радіотехніки. Однак і в електроніці цифрові пристрої та методи обробки сигналу досить швидко витісняють класичні аналогові елементи, що також треба враховувати. Звичайно, не слід забувати і про те, що з іншого боку базою цифрових засобів є елементи на основі роботи з аналоговим електричним сигналом.

Інформатика та обчислювальна техніка є важливою для інженера-радіотехніка не тільки як частина змісту самого проекту, який значною мірою пов’язаний саме з обробкою інформації. Слід зауважити, що сучасні проекти будь-якої інженерної галузі у собі концентрують величезні обсяги інформації, які неможливо обробити без сучасних комп’ютерних засобів вже на етапі розробки. Так само, як і забезпечити керування складними технічними засобами на етапі їх експлуатації. Відповідно, комп’ютерні засоби використовуються ще

й як інструмент розробки, а також для керування виготовленням та при експлуатації.

Обробка інформації за сучасної ідеології виконується програмними засобами, але ці засоби для свого виконання повинні використовувати відповідне апаратне забезпечення. Таким чином, програмна та апаратна складові дуже тісно пов’язані між собою. Тому зміст даного посібника зосереджений як на програмній (логічній) частині процесу обробки інформації, так і на апаратному (фізичному) забезпеченні.

Для розуміння принципів роботи з інформацією більш докладно розглядаються окремі формати даних, найбільш важливі на сучасний момент принципи перекодування – стиснення даних та шифрування. Розглянуті також базові принципи створення розподілених систем, оскільки такі системи дуже щільно пов’язані з сучасним використанням телекомунікацій та радіотехніки. Окрему увагу приділено основам моделювання та використанню програмних інструментальних засобів проектування, які використовуються в інженерії, зокрема у галузі радіотехніки.

1. Основи обробки інформації

1.1. Місце інформації в комп'ютерних технологіях

1.1.1. Інформація та дані

Перш ніж перейти до розгляду практичних застосувань комп'ютерної техніки для обробки інформації спробуємо розібратись у базових визначеннях.

Класичним визначенням інформатики є таке:

Інформатика — теоретична та прикладна (технічна, технологічна) дисципліна, що вивчає структуру і загальні властивості інформації, а також методи і (технічні) засоби її створення, перетворення, зберігання, передачі та використання в різних галузях людської діяльності.

В даному визначенні використано інший базовий термін – інформація. З одного боку він настільки часто вживається, що здається зрозумілим. Але якщо спитати неспеціаліста з інформатики, наприклад, у чому полягає різниця між інформацією і даними, як правило виникає ніяковість. Спробуємо розібратись у цьому.

Якщо спробувати знайти чітке та уніфіковане визначення терміну «інформація», то нас очікує поразка. Це поняття одночасно настільки фундаментальне, а з іншого боку філософське, що однозначного визначення просто не існує. Але з'ясувати, що під цим терміном розуміють, є дуже важливим.

Людство протягом усієї історії має потребу відокремити опис властивостей певних об'єктів від самих об'єктів, максимально його абстрагувати. Така потреба виникає і при накопиченні певного досвіду та його передачі від одних людей до інших, без чого неможлива інтелектуалізація на рівні суспільства, і при створенні проекту того, що буде зроблено незабаром. Тобто є потреба описати об'єкт без самого об'єкта. Як визначення будемо розуміти наступне:

Інформація – відомості про стан системи (об'єкта), які можуть існувати незалежно від системи (об'єкта).

Відповідно, будь який опис об'єкта, який може існувати незалежно від цього об'єкта, є інформацією про цей об'єкт. Наприклад, розміри аркуша паперу, його колір, вага, вологість тощо. Ми можемо отримати все це через безпосереднє спостереження або дослідження даного об'єкта, але ми можемо записати цю інформацію,

передати її на велику відстань і вона буде сприйматись без потреби безпосередньої взаємодії з об’єктом, якому вона відповідає. Більш того, інформація може залишитись навіть тоді, коли сам об’єкт змінився або навіть припинив своє існування. Завдяки збереженню передачі інформації ми можемо уявити об’єкт, до якого не маємо безпосереднього доступу, навіть побачити або почути його, навіть відтворити його копію, звичайно, маючи певне обладнання та матеріали.

Але інформація в зазначеному розумінні не є матеріальною і для її збереження та передачі виникає потреба у деякому носії або середовищі передачі. Інформація не може бути передана, прийнята або збережена в чистому вигляді. Носієм її є повідомлення. Таким чином, інформація не існує сама по собі, а лише через використання носія.

Відповідно, треба згадати ще про одне з базових понять, дуже близьких для радіотехніки і електроніки.

Сигнал – матеріальний носій інформації, довільна фізична величина, що змінюється в часі та/або в просторі.

Сигнал має певну фізичну природу, наприклад є акустичним або електричним. Саме природа сигналу (тиск повітря або струм) зазначає фізичну величину. В інформатиці полюбляють абстрагуватись від фізичної природи сигналу, фактично розглядаючи його як довільний носій інформації і використовуючи постулат про інваріантність інформації щодо носія. Сигнали можуть бути:

- часовими (фізична величина змінюється в часі);
- просторовими (є залежність тільки від координат);
- просторово-часовими (зміна є як у просторі, так і у часі).

Прикладом часового сигналу є звуковий сигнал. Зрозуміло, що акустичні хвилі розповсюджуються у просторі, але з точки зору інформації суттєвою є саме часова залежність у довільній фіксованій точці, у якій, наприклад, знаходиться приймач сигналу. Зміна його положення не буде відповідати іншій інформації, яку приймач може отримати, хоча слід зауважити, що зміна положення приймача може призвести до зменшення повноти сприйняття сигналу, наприклад через затухання або завади від інших сигналів.

Прикладом просторового сигналу є статичне (незмінне у часі) зображення. В даному випадку точковий приймач у фіксованій точці не дає змоги отримати усю інформацію про зображення. Але з часом кожна точка зображення залишається незмінною.

Прикладом просторово-часового сигналу є динамічне зображення, тобто таке, що змінюється у часі. У цьому випадку у певний фіксований момент часу ми маємо просторовий розподіл, як і

у випадку статичного зображення, але кожна фіксована точка зображення змінюється у часі, як у раніше розглянутому випадку акустичного сигналу.

Для передачі інформації на певну відстань сигнали фізичної природи, прийнятні для безпосереднього використання, наприклад зручні для безпосереднього сприйняття їх людиною, можуть бути неприйнятними. В більшості випадків причиною є саме природа сигналу, для якої складно або взагалі неможливо реалізувати телекомунікаційне обладнання, або сигнал даної природи має сильне затухання. Відповідно електромагнітна хвиля та електричний сигнал (при використанні дротів) виявились найзручнішими і саме їх використання сформувало телекомунікаційну галузь.

Сигнал може своєю величиною повторювати поведінку фізичної величини, інформацію про яку він відтворює. Наприклад, деяка напруга може бути пропорційною зсуву об'єкта. Реалізувати такий зв'язок зазвичай є найпростішим, але одночасно з цим неефективним. Виникає потреба у більш складній функціональній залежності фізичного явища, яке відповідає об'єкту (тобто інформації про об'єкт), і сигналу-носія. З'являється потреба у кодуванні.

Кодування – встановлення співвідношення (таблиці співвідношення) двох систем формальних об'єктів (переклад, шифрування та інше).

Кодування може бути виконане фізичною (апаратною) реалізацією будь-якого перетворення сигналу. Якщо така функція є неперервною (тобто існує функціональний зв'язок для довільного значення аргументу функції), кажуть про аналогове кодування. Якщо ставиться у відповідність деяка множина значень (кодів), говорять про цифрове кодування. Відповідно, ***кодом зветься форма представлення повідомлень, в яких реалізовано деякі правила, що забезпечують відповідність між повідомленнями і кодовими символами.***

Прикладом кодування є реалізація довільної мови людини, повідомлення у якій складається з набору звуків (фонем), які утворюють певні наперед визначені структури мови. Для запису фонем використовується набір наперед узгоджених зображень – літер.

Вже на цьому прикладі, навіть без залучення прив'язки до сучасних комп'ютерних технологій, стає зрозумілим, що та ж сама інформація може бути кодованою різними способами, наприклад деяке повідомлення, не змінюючи свого змісту, може бути відтворене англійською, українською або китайською мовами, і такі представлення будуть зовсім різними. Ось тут і з'являється потреба

ввести поняття, яке буде відповідати деякій інформації, але відокремлене навіть від інформації.

Введемо термін «дані». Як і з терміном «інформація», виникає проблема однозначності визначення і цього терміна. В літературі можна знайти багато зовсім різних його трактовок. Наприклад, «даними називають факти, відомості, представлені у формалізованому вигляді, тобто закодовані, занесені на певні носії, придатні до деякої обробки за допомогою спеціальних засобів». Таке визначення є дуже близьким до «інформації» і стає незрозумілим, для чого вводити синонімічні терміни. Інші джерела визначають дані як «матеріальні об'єкти довільної форми, що є засобом представлення інформації».

Спробуємо дати більш відокремлений зміст цьому терміну, який виділяє його в інший рівень, а саме ***дані – це деяке представлення інформації, придатне для її накопичення, передачі, обробки.***

Саме як дані інформація вводиться до комп'ютера, зберігається у його пам'яті, передається через мережу, зберігається на деякому носії. Тому часто замість терміну «обробка інформації» вживають термін «обробка даних». Якщо ці терміни вживати як деяке узагальнення без конкретизації самого способу обробки, це не є помилкою, оскільки будь-яка обробка інформації відповідає обробці даних. Але при цьому слід пам'ятати і про те, що реальні дії з інформацією і даними, які відповідають цій інформації, можуть відрізнятись. Наприклад, може бути виконане стиснення (тобто зменшення формальної кількості) даних без зміни інформації, що відповідає цим даним. Це реалізується, наприклад, у загальновживаних програмах-архіваторах.

Виходячи зі сказаного, можна дані вважати результатом деякого кодування, тобто інформацією, яка однозначно характеризує іншу інформацію, однак такий рекурсивний ланцюжок використання понять краще не застосовувати через плутанину, до якої він може призвести. Одразу визначимо новий термін, який ми використали: ***рекурсія – повторення через самоподібність, тобто подібність фрагменту до повного об'єкта.***

Іноді використовують термін «інформація про інформацію», але для цього є спеціальний термін – «метадані». Вони характеризують допоміжні властивості, наприклад режими отримання інформації, власника, посилання на деякі зв'язані блоки інформації, пошукові параметри тощо. Зрозуміло, що можна визначити і «мета-метадані», і «мета-….метаметадані», але ми знову прийшли до надлишкового використання рекурсії.

1.1.2. Властивості інформації

Інформація як поняття дозволяє сформулювати певні задачі інформатики. Однак для більш повного розуміння таких задач слід розуміти і власні властивості самої інформації. Основними властивостями інформації є:

- об’єктивність;
- повнота;
- достовірність;
- доступність;
- актуальність;
- цінність (корисність).

Об’єктивність інформації означає її незалежність від того, яким чином вона була отримана, передана або збережена. Інформація може бути частково втраченою за рахунок неідеальності засобів отримання або передачі, але це не порушує об’єктивності тієї частини інформації, яка залишилась. В даному випадку зменшується не об’єктивність інформації, а степінь її повноти. Відповідно, під повнотою інформації можна розуміти те, наскільки докладно інформація описує об’єкт, чи є вона достатньою для розуміння його властивостей або стану.

Але навіть повна інформація при усій її об’єктивності може або бути близькою до реальних властивостей та стану об’єкта, або частково не відповідати дійсності через хибність способу отримання, або навіть через спеціальне її викривлення. Таким чином, ми приходимо до розуміння такої властивості, як достовірність.

Під доступністю інформації розуміють ресурсоємність отримання інформації, тобто сукупність часу, потребу у коштах, в апаратному забезпеченні тощо. Тобто чим доступніша інформація, тим простіше, дешевше, швидше її отримати.

Актуальність інформації визначає її відповідність моменту часу, коли вона ще адекватно описує об’єкт або його стан. Чим актуальність її вища, тим більш ефективно вона може бути використана. Як приклад можна навести задачі керування динамічною системою. У тому випадку, коли керуючий засіб для формування впливу використовує неактуальну, тобто отриману з певним запізненням, інформацію, хибність впливу може призвести навіть до краху системи.

Цінність визначається здатністю її забезпечити користувача необхідними умовами для досягнення поставленої мети.

1.1.3. Дії з інформацією

Довільна діяльність людей так чи інакше пов'язана з інформацією та вимагає виконання над нею деяких дій, сукупність яких створює ***інформаційний процес,*** який включає у себе:

- створення інформації,
- отримання,
- збирання,
- накопичення,
- зберігання,
- обробку,
- пошук,
- розповсюдження,
- використання.

Початком ланцюжка, який відповідає інформаційному процесу, є саме створення інформації на основі фізичної взаємодії з об'єктом, спостереженням або експериментом. В рамках даного інформаційного процесу початкова інформація або її частина може бути результатом іншого інформаційного процесу, що є отриманням інформації. Надходження інформації від декількох джерел або її накопичення у часі призводить до збирання інформації, яке зазвичай реалізується з її фіксацією на деякому фізичному носії, який має властивість зберігати свій стан протягом довгого часу – зберігання інформації. Створення нової інформації на основі наперед отриманої та накопиченої називається її обробкою. При накопиченні великих об'ємів інформації стає актуальною така дія, як пошук, тобто забезпечення вибірки підмножини інформації, яка вдовольняє певним умовам. Часто споживач інформації територіально є віддаленим від її джерела, що вимагає передачі або розповсюдження інформації. Кінцевим етапом інформаційного процесу є використання інформації.

1.1.4. Обробка інформації як послідовність команд

Цифрова обробка даних спирається на послідовність певних арифметичних розрахунків у вигляді окремих дій. Саме на послідовність і саме окремих дій. Відповідно як реалізація такої послідовності з'являється процес, який отримав назву ***програма***. При виконанні кроків програми довільно визначається не тільки сама ***операція*** (дія), а й деякий блок даних, що розглядається як деяка єдність, над яким вона виконується. Цей блок даних називається ***операндом***. Таким чином, для виконання програми треба задати деяку

множину дій і створити засоби виконання цих дій, точніше команд, які запускають їх реалізацію.

Оскільки обробка інформації базується на деяких сигналах. Для перетворення даних можна реалізувати пристрій, який буде на основі миттєвого значення одного чи декількох сигналів певної фізичної природи створювати новий сигнал, тобто нове його миттєве значення. Перевагою такої обробки є «миттєвість», а точніше мінімальний час отримання результату, який визначається виключно характерним часом інерційності пристрою-обробника. Наприклад, таким чином можна задати суматор двох механічних зсувів, які в даному випадку і будуть операндами, або двох напруг. Пристрій, який оперує з неперервними величинами сигналів, отримав назву аналогового комп'ютера. Такий пристрій має два суттєвих недоліки – потребу складного і повільного переналаштування при зміні характеру обробки та погану точність для випадку складної обробки.

Замість того, щоб окремим чином реалізувати виконавчий пристрій для кожної складної операції, зазвичай зручно ці операції звести до послідовності деяких простіших, які виконуються по черзі одна за одною. Тобто пристрій обробки інформації в цьому випадку виконує ***програму – набір інструкцій (команд), виконання яких забезпечує обробку інформації.*** Відповідно, ***все, що не відноситься до програми, тобто відтворює інформацію, а не команди, вважається даними***.

Зазвичай з врахуванням традиційного групування операцій на арифметичні та логічні (булеві), цей пристрій називають ***арифметико-логічним пристроєм***. Виконання операції призводить до формування нової єдності даних – результату (рис. 1.1.).

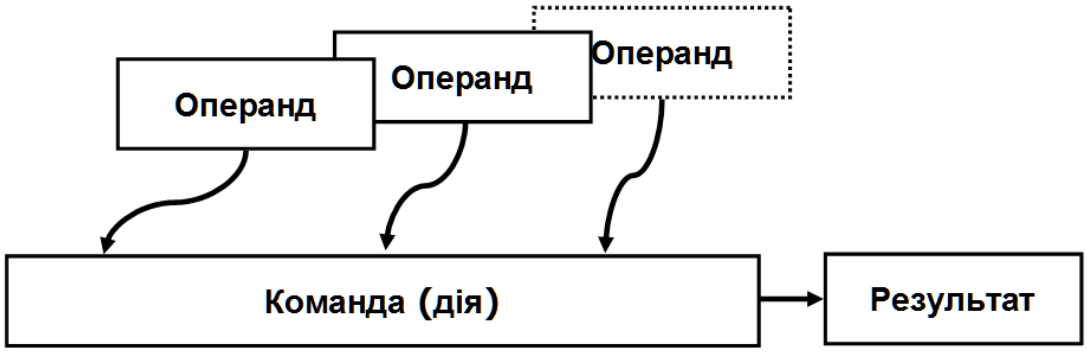


Рис. 1.1. Загальна схема виконання команди

Якщо дія формує результат при взаємодії з одним операндом, вона називається унарною. Прикладом унарної операції може бути

мінус, що змінює знак числа на протилежний, або булева операція заперечення.

Якщо для формування результату є необхідними та достатніми два операнди, операція називається бінарною, наприклад, звичні арифметичні дії. Аналогічним чином можна визначити тернарні (з трьома операндами) і т.д. операції. Іноді для сукупності усіх операцій, старших за бінарні, використовують загальний термін «багатооперандні».

Зазвичай використовують певну сукупність унарних та бінарних операцій, а багатооперандні є доволі рідкою екзотикою. Спробуємо розібратись, у чому причина такої організації системи обробки інформації.

Уявімо, що ми розробляємо певний засіб обробки інформації, який повинен виконувати деякий заданий набір обробок, кожна з яких є деякою функцією набору даних. Формально можна вважати, що кожний варіант обробки даних є багатооперандною операцією, для якої операндами є повний набір (вектор) вхідних даних. Для реалізації такої багатооперандної команди доведеться реалізовувати деякий апаратний пристрій, який виконує саме цю команду. Головним недоліком цього пристрою буде те, що він буде виконувати тільки цю операцію при відносній громіздкості цього вузькоспеціалізованого пристрою.

Як альтернатива, ця операція може бути представлена як сукупність послідовності більш простих операцій з деякого набору. Для виконання цієї послідовності можна реалізувати програму, більш детальну і подовжену, але з простішим набором дій. В результаті ми будемо мати універсальний пристрій, на якому замість однієї багатооперандної обробки може бути реалізована безліч варіантів обробки. Відповідно, для універсальних засобів обробки інформації, яких у світі переважна більшість, використовуються максимально прості за своєю структурою, тобто елементарні, операції, а складна обробка реалізується як їх сукупність (послідовність). При цьому виникає потреба виконання ланцюжка операцій, у якому результат поточної операції є операндом наступної (рис. 1.2.).

Це не означає, що, наприклад, тернарні операції не мають права на існування. В деяких випадках для спеціалізованої обробки інформації виділяють деякий обмежений набір операцій, що реалізують як багатооперандні. Зрозуміло, що обмежуються найбільш популярними діями з даними, які мають місце у даному варіанті спеціальної обробки. Чому саме окремими, а не усіма, стає зрозумілим з комбінаторики. Тернарна операція є комбінацією двох бінарних. Якщо взяти за основу такий традиційний набір бінарних операцій, як

чотири арифметичні дії, будемо мати повний набір таких операцій з 16 елементів, а набір кватернарних (чотири операнди) з 64. При цьому слід пам'ятати, що під кожну операцію повинен бути реалізований оптимізований апаратний пристрій, що виконує багатооперандну команду швидше, ніж більш простий пристрій обробить послідовність відповідних бінарних.

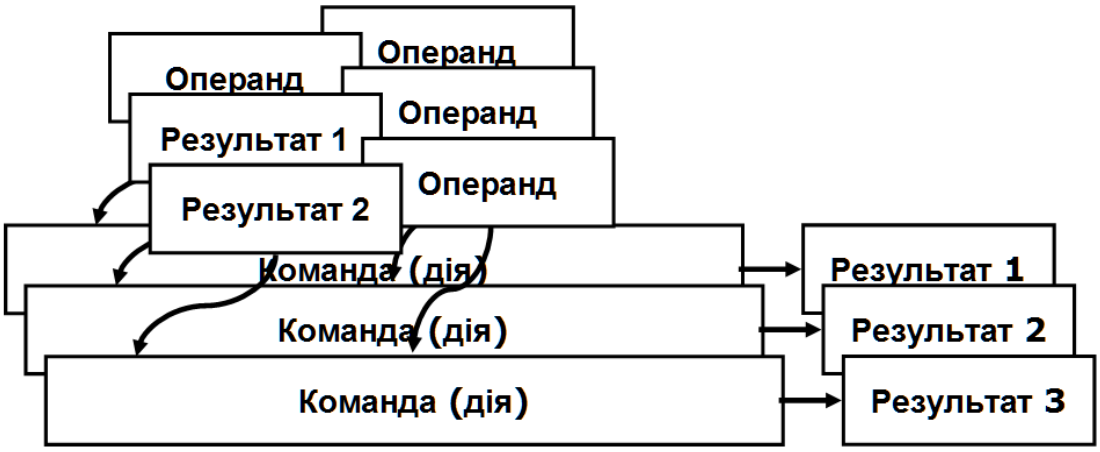


Рис. 1.2. Ланцюжок операцій

1.2. Пристрої для цифрової обробки інформації

1.2.1. Історія розвитку

Автоматизація процесу обробки інформації привернула увагу людства досить давно. Традиційно існують два способи передачі інформації між людьми – слова, точніше деяка мова на основі слів, та числа. Взагалі мова серед цих двох засобів є більш зручною для людини, але обробка мовної інформації є значно більш складною і ресурсоємною задачею. Саме тому перші (і майже усі сучасні) засоби автоматизації обробки інформації спирались на числову форму інформації, тобто обчислення.

Даний курс не є історичним трактатом і у ньому немає можливості детального викладення бурхливої і цікавої історії розвитку комп'ютерних технологій, але на деяких принципових моментах, які найсуттєвішим чином вплинули на сучасний вигляд комп'ютера і шляхи розвитку і галузі застосування інформаційних технологій, слід зупинитись, оскільки це потрібно для повноцінного розуміння багатьох з цих технологій.

Традиційно в літературі до перших обчислювальних засобів відносять Абак (з'явився приблизно 3000 років тому) – перший варіант звичайної рахівниці, яка звичний нам вигляд з прутками та

стрижнями отримала близько 300-500 років тому. Але ці пристрої є навіть не обчислювальним засобом, а деяким механічним пристроєм спрощення відтворення порозрядного підсумовування. Спробуємо розібратись, де у випадку рахівниці ховається «механізація» розрахунків. Для цього почнемо з традиційного способу усного рахування.

Якщо згадати шкільну програму молодших класів, реалізується двоетапний алгоритм (покроковий опис виконання дії – більш точно і детально цей термін буде розглянуто пізніше). Нижчий рівень цього алгоритму відповідає додаванню однорозрядних чисел, а верхній задає спосіб взаємодії операцій нижнього рівня. Цей рівень повинен враховувати додавання двох одно розрядних операндів та враховувати додаткове число, яке може виникати в попередній низькорівневій дії як результат переповнення попереднього розряду. Розглянемо приклад появи такого переповнення. 3+5 дорівнює 8 і не виходить за межі діапазону цифри. Переповнення не виникає. Але 6+7 дорівнює 13, тобто в даному розряді повинне бути значення 3, а 10 є саме «переповненням» і це значення враховується у у наступному розряді, тобто у наступній елементарній дії.

Схожим чином будується і алгоритм множення. І що цікаво – якщо, наприклад, однорозрядне множення зазвичай вивчають як деяку табличну операцію, тобто просто запам’ятовують наперед задану таблицю залежності результату від операндів, яка формується за визначенням операції множення, то при описі виконання операції додавання термін «таблиця додавання» зустрічається дуже рідко. Але таблиця додавання (та її запам’ятовування) неявним чином у навчанні школярів все одно існує. Рахівниця дає змогу замінити вивчення таблиці додавання використанням менш інтелектуальних механічних рухів, саме тому не зовсім коректно вважати цей пристрій «обчислювальним».

Ситуація суттєвим чином змінилась з винаходом справжнього механічного суматора (Блез Паскаль, 1642 р.) та подальшим розвитком механічних арифмометрів. Такі пристрої вже є певною реалізацією автомату, хоч і з ручним приводом (наприклад, додавання виконується повним обертом рукоятки). Він забезпечує виконання операції, як мінімум додавання та множення (через рахування кількості обертів рукоятки).

Наступним потужним кроком на шляху автоматизації обробки інформації стала різницева машина, запропонована у 1822 році англійським математиком Чарлзом Беббіджем. Цей пристрій мав за мету автоматизацію обчислень шляхом апроксимації функцій багаточленами і обчислення кінцевих різниць. Зауважимо, що

Беббіджу не судилось через технічні та фінансові складності реалізувати проект у повному обсязі, але це було першою спробою реалізації пристрою, який відтворює покрокову дію, тобто алгоритм.

При виконанні цього проекту у Беббіджа з'явилась також ідея створення «аналітичної машини», вже з арифметичним пристроєм (Беббідж назвав його «млином»), сукупністю регістрів («склад»), і пристроями введення/виведення (на основі використання перфокарт – картонок з дірочками у потрібних місцях). Більш того, було запропоновано використовувати окремі перфокарти для потоку команд і потоку даних. Тобто цю ідею можна вже розглядати як прототип комп'ютера, хоч і з суто механічною реалізацією.

Ще одним цікавим пристроєм, який можна віднести до прототипів комп'ютера, є табулятор, призначений для виконання простих операцій з великими об'ємами даних за допомогою перфокарт. Перший табулятор був побудований у США Германом Холлеритом для обробки результатів перепису населення 1890 року. Слід зазначити, що в наступні роки Холлерит отримав декілька премій, став професором Колумбійського університету. Крім цього, він заснував TMC (Tabulating Machine Company), яка у 1924 році отримала відому зараз назву IBM.

Тобто певні успіхи були досягнуті, але справжнім початком ери комп'ютерів можна вважати 1927 рік, коли в Массачусетському технологічному інституті (MIT) був розроблений механічний аналоговий комп'ютер (Венівар Буш). Наступним проривом стала заміна механіки як носія інформації, на більш швидкий у розповсюдженні та простий і надійній у перемиканні електричний струм. Звичайно, до сучасних мікросхем і, навіть, транзисторів було ще далеко, і основою елементої базі були реле. До таких машин відносяться, наприклад, «Марк I» (Automatic Sequence Controlled Calculator - автоматичний обчислювач, керований послідовностями), 1941 рік, Говард Ейкен). Надалі як елементна база почали використовуватись вакуумні лампи, тобто обчислювальна техніка почала використовувати засоби електроніки – ЕНІАК (ENIAC, від Electronic Numerical Integrator and Computer, 1946 р.).

Перш ніж розглянемо наступний крок, що перевернув комп'ютерний (і не тільки комп'ютерний) світ, ще раз звернемось до вже викладеної частини історії обчислювальних засобів. А саме зосередимо увагу на тому, які проблеми викристалізувались з цієї історії.

Основною з таких проблем є потреба збільшення кількості елементів для забезпечення універсальності і швидкодії. Розглянемо, наприклад, підсумовування багаторозрядних чисел. Для цього можна

використовувати апаратну реалізацію однорозрядного суматора, але для n-розрядних операндів його доведеться використати n разів, розряд за розрядом. Замість цього можна реалізувати n-розрядний суматор як декілька простих суматорів з додатковими зв'язками. Стає зрозумілим, що ми отримуємо значний виграш у швидкості виконання операції, але такою ж мірою програємо у простоті апаратної реалізації. Таким чином, окремі елементарні операції з більш складними даними можуть бути реалізовані більш складними апаратними пристроями (тобто з більшою кількістю базових елементів), замість послідовного виконання сукупності простіших дій. Зазначимо, що цей шлях є одним зі способів нарощування швидкодії. Він використовується для вдосконалення обчислювальної техніки протягом усієї її історії. Але це вимагає певної ціни, яку доводиться платити, а саме:

- збільшення загальних розмірів приладу;
- зменшення надійності;
- збільшення енергоспоживання;
- збільшення ціни.

Найбільш очевидним є саме збільшення розмірів, що ускладнює використання такого пристрою. З цього ж слідує і зростання ціни, що також значною мірою обмежує можливості широкого застосування. Збільшення енергоспоживання, крім впливу на ціну експлуатації, призводить до більш суттєвих технічних проблем. Уся спожита енергія перетворюється у тепло, яке повинне бути відведене від електронного пристрою для запобігання його перегріву та виходу з ладу.

Проблема надійності стає зрозумілою, якщо порахувати ймовірність виходу з ладу складного пристрою. Уявимо, що пристрій складається з N однакових елементів, для яких ймовірність виходу з ладу за певний час складає P. Відповідно, пристрій вийде з ладу за цей же час, якщо хоч один з елементів вийде з ладу, тобто загальна ймовірність такої події може бути розрахована відніманням ймовірності справності усіх і сягатиме $1-(1-P)^N$.

Виходячи з наведеного, для ефективного розвитку технологій обчислювальної техніки одночасно з нарощуванням кількості елементів конче необхідно зменшувати розмір базового елемента, зменшувати його енергоживлення та збільшувати надійність. Одночасно з цим є бажаним боротись і за збільшення швидкодії цього базового елемента. Розглянута вище заміна механіки спочатку на реле, а потім на радіолампу, відповідає такому шляху розвитку.

Наступним кроком розвитку електроніки став перехід від радіолампи до напівпровідникових елементів – транзисторів, а потім

до мікросхем, які відкрили шлях до створення надмініатюрних елементів, сформованих у величезній кількості на одному кристалі. Саме поява мікросхем забезпечила прорив по всіх фронтах і зробила комп'ютерні засоби обов'язковим атрибутом сучасного життя.

У 1971 році з'явились перші мікропроцесори (Texas Instruments TMS 1000 і Intel 4004). І вже у 1976 році Стів Джобс, великий комбінатор у сфері ІТ, запропонував талановитому інженеру і своєму приятелю, якого він однак не раз обманював, Стіву Возняку (він і був фактичним автором пристрою), тиражувати «Apple I». Цей поки що недосконалий обчислювальний пристрій став першим персональним комп'ютером «широкого вжитку», і цим фактично було розпочато надзвичайно широке розповсюдження комп'ютерних технологій.

Зараз комп'ютер настільки тісно увійшов у життя суспільства, що без нього практично неможливо уявити будь-яку галузь діяльності людини. Комп'ютери забезпечують усі види зв'язку, значну частину телебачення та радіомовлення, при цьому сучасні користувацькі пристрої – телевізори та більшість радіоприймачів є фактично спеціалізованими комп'ютерами. Термінальні пристрої для телефонії (телефонні апарати) також відноситься до цього класу, а смартфони вже, мабуть, можуть класифікуватись як універсальні комп'ютери. Обчислювальна техніка забезпечує наукові дослідження, інженерні розробки, засоби медичної діагностики, керування (від мініатюрного дистанційного пульту побутового пристрою до величезного виробництва). До музею відправились друкарські машинки, креслярські кульмани, майже не використовуються паперові карти місцевості та планшети з такими картами для орієнтування на місцевості. І навіть сам термін «планшет» отримав зовсім інше значення.

1.2.2. Архітектура обчислювального пристрою

Зрозуміло, що різноманітність пристроїв обробки інформації за їх призначенням, різні потреби у швидкості обробки інформації, можливість різних способів розв'язпння задачі призводять до певної варіації у структурі апаратного забезпечення, тобто пристроїв і систем, на яких обробляються сигнали, що несуть у собі потрібну інформацію. Але є певні загальні принципи побудови архітектури таких пристроїв.

Відповідно до ГОСТ 15971-90 ***архітектура комп'ютера визначається як концептуальна структура обчислювальної машини, що визначає обробку інформації та включає методи***

перетворення інформації в дані і принципи взаємодії технічних засобів і програмного забезпечення.

Основи архітектури комп'ютерів були закладені у 1943 році фон Нейманом у складі колективу розробників, які працювали над створенням ENIAC (рис. 1.3.).

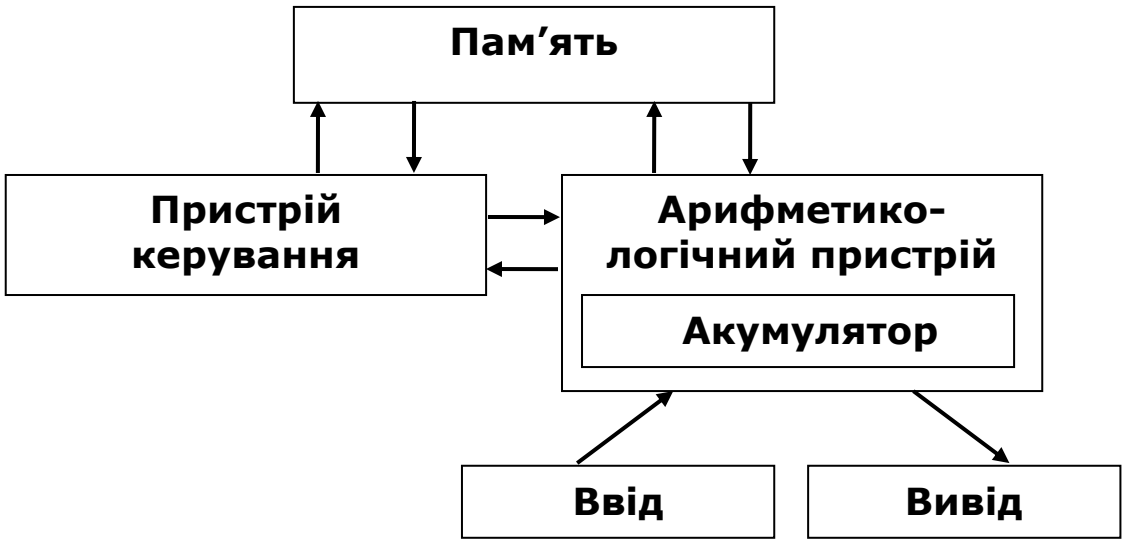


Рис. 1.3. Архітектура обчислювального пристрою, запропонована фон Нейманом

Основні ідеї цієї архітектури фон Неймана залишаються актуальними і сьогодні. Ця архітектура виходить з потреби виконання наперед зданого ланцюжка дій над заданим набором даних з метою виробити нові (оброблені) дані. Відповідно до цього є потреба у блоці, який безпосередньо виконує деяку дію з наперед заданого набору. Цю задачу виконує операційний або арифметико-логічний пристрій (АЛП). Частою є ситуація, коли результат виконання команди є операндом наступної. Для цього АЛП може бути реалізований на основі акумулятора, у який цей результат записується і надалі. Відповідно, значна частина команд використовує дані акумулятора як один з операндів. Зазначимо, що це не єдина можлива архітектура АЛП, але досить популярна. Детальніше це питання буде розглянуте у п. 5.1.1.

Необхідну послідовність виконання команд та узгодження цих команд з відповідними даними повинна бути забезпечена завантаженням їх кодів (також деяких чисел) в АЛП та розпізнаванням, що виконує пристрій керування. Цей пристрій повинен також обробляти позаштатні результати виконання операцій та виконувати інші керуючі та узгоджуючи дії. Зазвичай АЛП та

пристрій керування разом утворюють єдиний блок, який має назву процесор.

Результат виконання операції не завжди є операндом саме наступної дії, тобто є потреба у тимчасовому збереженні даних. Цю функцію виконує запам'ятовуючий пристрій (або скорочено – пам'ять). Цей пристрій може бути реалізований більш швидким за інші допоміжні пристрої, з яких можуть надходити команди та дані, і його також зручно використовувати і для збереження кодів команд при потребі їх багаторазового використання.

Оскільки є необхідність забезпечити зв'язок обчислювального пристрою з зовнішнім світом, виникає потреба у пристроях вводу та виводу. Через те, що зайшла мова про пристрої вводу/виводу, одразу введемо таке важливе в інформатиці поняття, як ***файл (file) – блок інформації на зовнішньому пристрої комп'ютера, що має певне логічне представлення (від послідовності бітів або байтів аж до інформаційних об'єктів) та спосіб однозначного виділення серед подібних, наприклад, за допомогою деякого спеціального атрибута (наприклад, імені).***

Одночасно з запропонованою архітектурою фон Нейманом було сформульовано деякі базові принципи, відповідно до яких повинен працювати пристрій наведеної вище архітектури:

- Принцип програмного керування визначає, що довільна послідовність дій обчислювального пристрою зводиться до послідовності керуючих слів – команд, кожна з яких виконує деяку елементарну дію з наперед заданого набору.
- Принцип двійкового кодування, тобто використання тільки двох можливих цифр (0 і 1), послідовністю яких кодується будь-яка інформація та команди. Відповідно, для команд кожна частина двійкової послідовності (поле) має своє призначення, наприклад, безпосередньо код команди, спосіб доступу до операндів тощо. Структура усіх полів команди визначає її формат.
- Принцип однорідності пам'яті. Команди і дані зберігаються спільно в єдиній пам'яті. Оскільки вони зведені до двійкового представлення, розпізнати їх можна тільки за способом використання. Це дозволяє проводити над командами ті ж операції, що і над даними.
- Принцип адресності. Виходячи з потреби сумісного збереження певної кількості команд і даних, треба забезпечити однозначне визначення фрагменту бінарного коду, який відповідає тім чи іншім командам або даним. Це забезпечує прив'язка кожної комірки пам'яті (фрагменту

пам’яті, який розглядається як неподільне ціле), адреси – певного номера, який однозначно ідентифікує цей фрагмент.

За сучасною ідеологією архітектура комп’ютера включає в себе:

- архітектуру процесора;
- особливості реалізації підсистеми пам’яті;
- реалізацію зовнішньої периферії;
- оточення процесора;
- структуру зв’язків.

Відповідно, архітектура процесора включає в себе:

- систему команд;
- орієнтацію на визначені типи даних;
- структуру блоку виконання операцій;
- структуру системи регістрів;
- структуру внутрішніх зв’язків.

Система команд повинна забезпечувати:

- однозначність інтерпретації;
- функціональну повноту;
- ефективність реалізації;
- обмеженість та нарощуваність.

Детальніше цю класифікацію розглянемо у п. 5.1.1.

1.3. Розробка програмних засобів

1.3.1. Мова програмування та послідовність команд

Оскільки сучасні обчислювальні засоби спираються на принцип програмного керування, стає зрозумілою особлива роль програмних засобів, без яких вони є непрацездатними. Складність програмних засобів вимагає створення деякого середовища розробки. Оскільки процесор як пристрій, що виконує команди, розуміє деяку наперед задану систему команд-кодів, то довільну програму можна розробити виключно на основі цієї сукупності кодів. Але створення програми у кодах і форматах, зрозумілих для процесора, є надто незручним для людини. Причиною цього є те, що бінарне кодування команд процесора вступає у протиріччя з рівнем абстракції нашого мислення, орієнтованого на слово, що відповідає фонемним структурам певної людської мови, яка є значно старшою за інформатику. Відповідно, для забезпечення можливостей створення програм виникає потреба

використання деякої мови програмування, яка є максимально наближеною до звичних людям мов спілкування.

При цьому в процесі розробки такої мови доводиться враховувати потребу у створенні спеціальної програми, яка на основі текстового запису в рамках даної мови може згенерувати сукупність коду у форматі системи команд деякого процесору, що накладає значні обмеження на таку мову, роблячи її значною мірою компромісом між зручністю для людини-програміста і ефективністю щодо пристрою виконання. ***Така програма називається транслятором. Відповідно, трансляція (в програмуванні) — перетворення програми, яка подана однією мовою, в еквівалентну програму іншою мовою.***

Найпростішим шляхом, який дозволяє примирити бінарний код зі звичними людям словами, є таблиця співвідношення коду і слова, що може скласти основу деякої мови з дуже простим транслятором. При цьому можна вибрати набір слів, що будуть відповідати діям, які вони кодують, з деякої загальнопоширеної мови, наприклад, як історично склалось, англійської. До цього можна додати ідентифікацію іменами, тобто замість числової адреси комірки пам'яті також використовувати деяке зрозуміле людям слово. Дійсно, в цьому випадку вираз (сукупність слів), які відповідають команді, стає семантично простим для сприйняття програмістом, а трансляція легко реалізується. Така компромісна (з врахуванням потреб не тільки програміста а й процесора) мова програмування отримала назву ***асемблера***. Її перевагою є можливість забезпечення найбільш ефективного використання системи команд процесора, оскільки кожна з цих команд усвідомлено задається програмістом. Ціною такої ефективності є складність як написання, так і читання тексту програми, оскільки навіть арифметичний вираз з декількома діями, який зазвичай ми записуємо у рядок, що легко сприймається, розписується на декілька рядків.

Слід зауважити, що прив'язка до певної системи команд означає, що асемблер не є універсальним. Кожен варіант асемблера обслуговує тільки одну систему команд, і переробка програми під принципово інший процесор вимагає не тільки перекомпіляції, а й повної переробки тексту програми, що також є досить незручним.

Незручність використання асемблера призводить до потреби в мовах високого рівня – універсальних мовах, вирази у яких є більш наближеними до мови людини, а відповідність до системи команд повністю перекладається на компілятор. Мови високого рівня можуть використовуватись для пристроїв з різними системами команд, для чого треба розробити тільки відповідні компілятори. Такі мови

використовуються значно частіше за асемблер, оскільки значною мірою спрощують і пришвидшують розробку програм. Ціною цього спрощення є деяке (іноді дуже суттєве) зменшення ефективності виконання таких програм. Під ефективністю можна розуміти кількість ресурсів обчислювального пристрою (наприклад, часу виконання), необхідних для розв'язання задачі.

У 1995 році Sun Microsystems реалізувала свій проект платформи, а відповідно і мову, Java. Це був початок активного використання ***проміжної мови***, що є додатковим рівнем, до якого відбувається перша (попередня) компіляція з мови високого рівня. При виконанні програма на проміжній мові докомпілюється до команд процесора, що і забезпечує її виконання на цьому пристрої. Використання проміжної мови забезпечує кросплатформовість, тобто довільна програма може бути виконана на різних процесорах при умові, що для кожного з них буде реалізоване, тобто скомпільоване, середовище виконання, тобто програма, яка за основну функцію має саме докомпіляцію проміжного коду прикладної програми. У випадку Java така програми має назву Java Virtual Machine (JVM).

Така ж ідеологія була використана Microsoft при реалізації MS.Net з тією різницею, що в даному випадку не було передбачено кросплатформовості. Але слід зауважити, що навіть для цієї технології пізніше незалежні від Microsoft програмісти на базі ідеології «вільного програмного забезпечення» (open source) реалізували для MS.Net таку властивість проектом «Mono».

Таким чином, на сьогодні сформувалась певна структура рівнів мов та команд:

- мови високого рівня;
- проміжний код;
- асемблер (мова низького рівня);
- система команд процесора.

Кожен з цих рівнів має свої переваги і недоліки. Відповідно, для складних задач часто використовують навіть певні їх комбінації, коли окремі компоненти реалізуються різним чином. Наприклад, сучасні мови програмування високого рівня можуть передбачувати включення асемблерних фрагментів.

Виходячи з особливостей мов високого рівня, історично склались три режими трансляції:

- компіляція (у команди процесора);
- часткова компіляція (у проміжну мову);
- інтерпретація (компіляція фрагментами під час виконання).

Процес компіляції використовує достатньо складний аналіз тексту і є досить повільним, особливо у складних проектах з дуже

великим об'ємом тексту програми. Але підготовлена таким чином програма зберігається та розповсюджується як файл саме на основі формату команд процесора (виконуваний файл). При завантаженні у пам'ять комп'ютера для подальшого виконання вимагається мінімум налаштувань, пов'язаний з потребою прив'язки до адреси, яка виділена під початок даної програми. Це забезпечує велику швидкість виконання програми при такому режимі трансляції, особливо якщо під час компіляції використовується оптимізація за швидкодією.

При використанні інтерпретації програма зберігається у первинному текстовому вигляді, а її компіляція виконується тільки при її завантаженні фрагментами. Якщо взяти до уваги, що практично у всіх програмах є значна частина фрагментів, що повторюються багато разів, стає зрозумілим, що інтерпретована програма виконується значно повільніше за компільовану через потреби багаторазової компіляції фрагменту тексту, що повторюється.

Використання проміжного коду пришвидшує компіляцію під час виконання у порівнянні з інтерпретацією, але все одно варіант «попередня трансляція + інтерпретація» є досить повільним. Проривом у забезпеченні ефективності технології двоетапної трансляції стала технологія Just-in-time compilation (JIT), тобто «компіляція на льоту». Інша її назва dynamic translation. Особливість цієї технології трансляції полягає у тому, що компіляція з проміжної мови відбувається як і при раніше описаній інтерпретації, але у даному випадку результат остаточної трансляції буферується, і наступного разу вже може використовуватись готовий до виконання код без потреби повторної трансляції. Як результат, програми на основі JIT є трохи більш ресурсоємними (в основному повільнішими) за компільовані, але значно швидшими за інтерпретовані.

1.3.2. Класифікація програмних засобів

Ефективність програмних засобів значною мірою залежить від якості програмного забезпечення, правильного його використання та коректності взаємодії окремих його компонентів. Загалом програмне забезпечення включає в себе:

- прикладні засоби;
- системні засоби;
    - завантажувачі, монітори тощо;
    - операційні системи та їх компоненти;
    - засоби керування файлами;
    - засоби керування базами даних;
    - утіліти;

- інструментальні засоби;
    - засоби редагування;
    - засобі трансляції;
    - засоби відладки;
    - засоби генерації документації;
    - SDK та бібліотеки;
    - засоби інтеграції пакетів розробника.

Прикладна програма (Application) – це користувацька програма, що дає змогу вирішувати безпосередні задачі користувача. Зазвичай вона включає у себе такі компоненти:

- виконуюча частина;
- інтерфейс користувача;
- система допомоги;
- інсталятор.

За сучасною ідеологією розробки значна частина цих компонент реалізується використанням бібліотек, які є частиною засобу розробника та операційної системи. До основних особливостей сучасних прикладних програм можна віднести:

- орієнтацію на певні формати (внутрішню будову) даних;
- реалізацію деякої самодостатньої сукупності дій;
- гнучкість порядку виконання дій.

Якщо прикладні програми є безпосередньо тим, що забезпечує утилітарні потреби користувача, то системні засоби за своїм призначенням призначені виключно для ефективного використання прикладних програм. У сучасних умовах системні засоби згруповані навколо операційної системи (ОС). До основних функцій ОС відносяться такі задачі:

- завантаження програм в оперативну пам'ять, їх виконання (***виконання програми називається процесом***), звільнення пам’яті по завершенню виконання;
- керування пам'яттю, її розподіл між процесами та забезпечення її ізольованості від інших процесів;
- взаємодія між процесами, їх синхронізація, розмежування доступу різних процесів до ресурсів.
- підтримка файлової системи;
- уніфікація доступу до зовнішніх (периферійних) пристроїв;
- забезпечення інтерфейсу користувача;
- взаємодія з мережею.

Одразу зауважимо, що можуть існувати обчислювальні пристрої навіть без ОС. Виходячи з перерахованих функції ОС, вона є необхідною, коли програмний пристрій:

- потребує частої зміни програмного забезпечення;
- використовує багатозадачність;
- активно використовує файлові операції;
- потребує гнучкості настроювання;
- активно використовує ресурси та взаємодію з периферією.

Пристрій може працювати без ОС при умові:

- нечастої потреби зміни програмного забезпечення (як правило, воно не міняється протягом життя пристрою);
- однозадачності;
- простої структури файлових операцій;
- обмеженості гнучкості настроювання;
- інтерфейс взаємодії з оточенням передбачає розвинену апаратну взаємодію.

ОС надає програмному забезпеченню так зване ***операційне середовище – набір функцій і сервісів та правила звернення до них*** (завдяки цьому програми виконують звернення до ОС). При цьому ОС в загальному випадку може містити декілька операційних середовищ. Таким чином, ОС надає, а прикладна програма використовує ***системні ресурси.*** Ресурси можуть бути:

- неділимими (використовуються одним процесом);
- такими, що розділюються (використовуються декількома процесами разом).

Однією з важливих складових системних засобів є драйвер (driver) – програма, за допомогою якої операційна система отримує доступ до керування апаратним забезпеченням.

Є особлива категорія ОС, які зазвичай використовуються для керування складними приладовими або виробничими системами – ОС реального часу. Такі системи мають здатність забезпечити рівень сервісу, який вимагається за визначений проміжок часу, тобто така ОС повинна реагувати за передбачуваний час на непередбачувану появу зовнішніх подій.

Інструментальні засоби призначені безпосередньо для розробки програмного забезпечення і включають у себе:

- засоби редагування;
    - текстові;
    - графічні;
    - спеціальні;
- засобі трансляції;
    - компілятор;
    - інтерпретатор ;
- засоби відладки;

- засоби генерації документації;
- SDK та бібліотеки;
- засоби інтеграції пакетів розробника.

В разі, коли програмне забезпечення для одного засобу (наприклад, контролеру, який забезпечує функціонування деякого пристрою) розробляється на іншому (зазвичай – універсальному комп'ютері), застосовуються кросплатформові засоби:

- кростраслятор (трансляція на одному програмованому пристрої у команди іншого);
- емулятор (програмна інтерпретація та обробка команд одного обчислювального пристрою на іншому).

В процесі розробки активно використовуються засоби відладки, які забезпечують тимчасові зупинку у обраних місцях коду, покрокове виконання, перегляд/модифікацію значень змінних із метою виправлення помилок часу виконання.

Для спрощення створення супроводжуючої документації можуть використовуватись засоби її автоматичної генерації. Це дозволяє отримувати документацію, призначену для програмістів і користувачів на основі особливим чином коментованого тексту програми, а в деяких випадках і по виконуваних модулях (отриманих на виході компілятора).

При розробці програм часто використовуються такий інструментарій, як SDK (Software Developer Kit) – набір із засобів розробки, утиліт і документації, який дозволяє створювати прикладні програми за визначеною технологією або для певної платформи. Зазвичай використовуються і певні заготовки, які дозволяють використати раніше зроблені програмні блоки, що суттєво спрощує і пришвидшує розробку. Такі заготовки збираються у бібліотеки.

1.3.3. Процес розробки програмних засобів

Розробка сучасного програмного забезпечення є дуже складним процесом, який враховує багато факторів, пов'язаних як з архітектурою комп'ютера, так і з вибором технологій створення програми. Відповідно, можна розбити життєвий цикл довільної програми на декілька взаємозв'язаних етапів (одразу зазначимо, що у літературі таке розбиття часто робиться трохи інакше, наприклад докладніше):

- формування вимог, концепцій та технічного завдання;
- розробка проекту;
- тестування та доопрацювання;

- впровадження та розповсюдження;
- супроводження.

На першому етапі визначаються потреби, які повинен вдовольняти даний програмний засіб. Такі потреби можуть бути пов'язані з появою принципово нових задач обробки інформації, появою нового обладнання, для роботи якого потрібна програма, заміною старого програмного забезпечення на більш досконале та більш функціональне тощо. На основі програмних технологій, які існують на поточний момент, вибирається платформа, набори готових компонентів (бібліотеки), засоби розробки. Разом з цим створюється концепція розв'язання задачі та загальна архітектура програмного продукту.

Далі виконується безпосередня розробка текстів програми та додаткових ресурсів, які можуть створюватись навіть не у текстовій формі. В процесі розробки як до окремих складових, так і до усього програмного засобу застосовується відладка, яка дозволяє виявити ти виправити основні помилки проекту.

У стані, коли програмний засіб вже є придатним до виконання своїх задач, використовується детальна перевірка працездатності, часто багатоетапна з залученням не тільки колективу розробників, а й потенційних користувачів, з метою виявити більш глибоко приховані помилки. Цей процес називається ***тестуванням***.

Відтестований програмний продукт готується до розповсюдження. За потреби готується необхідна документація, проводиться навчання користувачів тощо.

В процесі експлуатації програмного продукту можуть виявитись раніше не знайдені помилки, з'явитись необхідність розширення функціональності, інших модифікацій. Крім цього, в багатьох випадках є необхідність консультування користувачів, надання допомоги у налаштуваннях. Все це називається ***супроводженням***.

Життєвий цикл програмного продукту завершується втратою потреби у ньому або заміною на нову версію.

Під час розробки програмного проекту доводиться брати до уваги необхідність розв'язання певних проблем, пов'язаних безпосередньо з запуском програми, її роботою на деякому обладнанні. Основними проблемами є:

- ефективність виконання;
- ефективність розробки;
- розміри коду та його складність;
- стійкість, надійність;
- сумісність.

При своєму виконанні програмний засіб може потребувати меншого або більшого об’єму використовуваної пам’яті, кількості звернень до зовнішніх пристроїв, часу виконання при певній кількості задіяних процесорів або їх частин, об’єму даних, що передається через мережу. Усі ці компоненти у комплексі отримали назву ресурсів. Відповідно, під ефективністю програмного засобу зазвичай розуміють мінімальну ресурсоємність при виконанні. Досягти високої ефективності програми можна використанням мови низького рівня, ефективних (особливо оптимізуючих) трансляторів, режиму компіляції, іноді оптимізацією структури програми ще при її написанні мовою високого рівня.

Ефективності розробки також відповідає ресурсоємність, але ресурсоємність розробки та доводки усього програмного проекту. Відповідно, її збільшення може бути досягнуто використанням раніше розробленого коду, наприклад, у вигляді бібліотек та шаблонів, засобів графічного програмування та спеціальних редакторів для підготовки окремих компонентів програми, сучасних засобів розробки, у яких передбачені елементи для спрощення роботи програмістів, засобів автоматизації адміністрування проектів. Все це зменшує потребу у кількості розробників та іншого персоналу, зменшує загальний час розробки.

Слід зауважити, що в останні десятиріччя складність структури програмних засобів суттєво збільшилась, що впливає і на розміри коду. Крім того, прагнення до збільшення ефективності розробки може саме по собі призводити до зростання коду і навіть до деякого зменшення ефективності виконання через значні нашарування старого коду.

Однією з важливих проблем при розробці програмних засобів є забезпечення надійності, тобто блокування помилкових ситуацій, які призводять до помилкового результату, некоректного завершення роботи, або, більш того, до втрати даних.

І ще раз нагадаємо вже раніше розглянуту проблему – можливість виконання програмного засобу на різних платформах, що може бути досягнуто або перекомпіляцією, яка іноді вимагає часткового переписування тексту програми, або використанням технологій на основі двоетапної компіляції (проміжної мови).

Відповідно, засоби розробки повинні забезпечити:

- підготовку та редагування тексту програми;
- трансляцію та збірку в рамках проекту;
- попередню перевірку синтаксису;
- виконання;
- трасування та відладку;

- розробку ресурсів (допоміжних елементів, необхідних для реалізації програми);
- комфортність роботи розробника, в тому числі зручність та повноту системи допомоги.

Оскільки першим етапом роботи з новою програмою є введення до комп'ютера її тексту, для роботи потрібен деякий текстовий редактор. У сучасних засобах розробки використовуються спеціалізовані текстові редактори, які мають деякі функції, пристосовані саме для програмування До цих особливостей відносяться автоматичне виділення кольором деяких синтаксичних структур та слів, оптимізація системи пошуку, в тому числі для роботи з декількома файлами. Останнє пов'язане з тим, що великий обсяг тексту сучасної програми робить незручним його створення в одному файлі. Відповідно, текст розбивається на декілька частин, кожна з яких навіть компілюватись може окремо з подальшою збіркою. Введемо таке поняття, як ***проект – (в комп'ютерних технологіях) сукупність файлів, які обробляються разом для реалізації результату.***

Після написання тексту зазвичай є потреба перевірки на відповідність синтаксичним вимогам використаної мови програмування. Зазвичай це робиться разом із трансляцією.

При відсутності синтаксичних помилок відтрансльована програма може бути запущена на виконання, але поки що гарантії правильності її виконання немає. Пошук помилок часу виконання забезпечується трасуванням та відладкою, які використанням спеціальних засобів. які дозволяють зупинити виконання у потрібних для докладного дослідження роботи програми місцях, виконувати покрокове проходження, перегляд значень даних та їх зміну при виконанні дій, що контролюються, і навіть змінювати ці дані вручну.

1.4. Представлення інформації у комп'ютері

1.4.1. Представлення інформації у цифровій формі

Зазвичай у природі людина має справу з інформацією, яку передають неперервні сигнали, величина яких може приймати будь-яке значення у межах певного діапазону (обмеженого або необмеженого), і це значення існує у довільний момент часу (для часових сигналів) або при довільних координатах (для просторових сигналів). Радіотехніка та електроніка саме займається обробкою та передачею таких сигналів головним чином заради забезпечення

інформаційних процесів, тобто довільна дія традиційно виконується аналоговими пристроями, які реалізують вихідний сигнал в залежності від вхідного (вхідних). Основним недоліком аналогової обробки є мала точність, що спотворює інформацію, а, відповідно, зменшує її достовірність. До цього додається також погіршення співвідношення сигнал/шум, що з точки зору інформатики означає часткову втрату інформації. Використання цифрових технологій дозволяє якщо не позбутись, то значно зменшити подібні втрати.

Для цифрової обробки або передачі інформації потрібно від кодування інформації неперервним значенням деякої фізичної величини перейти до такого коду, який буде захищений від зазначених недоліків. Але, як уже зазначалось, все одно носієм буде аналогова фізична величина. Відповідно, єдиним шляхом захисту від впливу шумів, або, наприклад, нелінійного викривлення залишається тільки внесення деяких суттєвих обмежень на спосіб інтерпретації аналогового сигналу. Саме спосіб інтерпретації і відповідає цифровому кодуванню інформації на основі неперервної фізичної величини. Одним із найпоширеніших способів подібного кодування є використання обмеженої кількості допустимих значень, кожне з яких розглядається як окремий елемент із заданої множини. Таке представлення називається квантуванням. При наявності будь-якого відхилення від визначених (квантованих) станів сигнал інтерпретується за найближчим з лінійки квантованих значень.

Таким чином, якщо замінити довільні значення на квантовані, ця операція вносить певну похибку, так звану похибку квантування, яка дорівнює половині кроку квантування. Однак при подальшій обробці, якщо додатковий шум або викривлення не перевищують похибки квантування, інформація може бути отримана в точності такою, що залишилась у сигналі при виконанні квантування. За рахунок цього, наприклад, можна забезпечити декілька послідовних ретрансляцій для передачі на надвелику відстань або багатократне послідовне копіювання з копії взагалі без втрати інформації.

Аналогічно до квантування для забезпечення можливості комп'ютерної обробки, яка орієнтується на скінченну кількість окремих даних, слід замінити нескінченну кількість відліків за часом або координатами на скінченну. Таке перетворення називається дискретизацією. Введемо визначення:

Дискретизація сигналу – це утворення відповідності неперервного за часом сигналу до дискретного за часом, тобто такого, що змінюється тільки в певні моменти часу (рис. 1.4.)

Квантування сигналу – операція утворення відповідності досліджуваному сигналу до квантованого, значення якого належать деякій наперед заданій множині (рис. 1.5.)

Рис. 1.4. Дискретизація аналогового часового сигналу

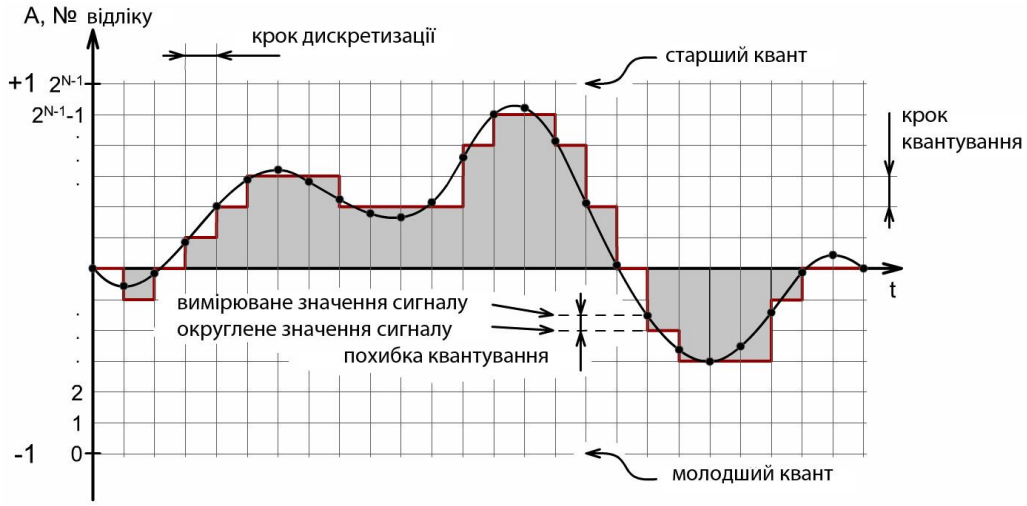


Рис. 1.5. Квантування аналогового часового сигналу

Сумісне застосування операцій дискретизації та квантування зазвичай називають ***оцифровуванням***. Оскільки будь-який природний сигнал має деяку шумову компоненту (або похибку), можна сказати, що при квантуванні з кроком не більшим за середньозважений рівень шуму, квантування не вносить додаткової похибки. Відповідно відношення значення сигналу до шуму визначає динамічний діапазон сигналу, тобто кількість інформації у відліку.

При дискретизації частотні складові, які перевищують половину частоти дискретизації, як слідує з теореми Котельникова (в англомовній літературі теореми Найквиста-Шенона), втрачаються. ***Якщо сигнал має спектр, обмежений частотою F_{max}, то він може біти однозначно представлений за своїми дискретними відліками з частотою $2F_{max}$***. Це також обмежує кількість інформації. Зауважимо, що і для аналогових пристроїв, саме гранична частота пристроїв, з яких складається система, є таким самим обмежуючим фактором. Аналогічним чином можна сказати, що квантування вносить похибку, яка дорівнює половині кроку квантування.

Призначенням дискретизації та квантування є обмеження кількості інформації від формально необмеженої кількості до скінченної множини. Для неперервного сигналу між двома довільними відліками завжди існує деякий проміжний, а неквантована величина сигналу у загальному випадку формально задається дійсним числом з необмеженою кількістю розрядів дробової частини. Але для реальних сигналів можна побачити, що дуже швидкі зміни не відбуваються через інерційність системи, а точність, з якою реалізується або вимірюється сигнал, обмежується додаванням паразитних сигналів (шумів), і аналоговий опис фактично є надлишковим, тобто застосувати дискретизацію і квантування є сенс навіть у випадках, коли є потреба мінімізувати можливу похибку обробки.

Відповідно до зазначеного можна стверджувати, що лінія передачі сигналу є виключно аналоговим пристроєм, але може використовуватись для передачі у цифровому режимі і при цьому створюється цифровий канал передачі. За означеннями:

Лінія – фізичний носій (середовище) для передачі сигналу, як правило електромагнітного (дріт, вита пара, коаксіальна лінія, оптичне волокно, ефір).

Канал – логічний носій, тобто спосіб інтерпретації сигналу для виділення даного потоку інформації (базується на повному або частковому використанні деякої лінії або середовища передачі).

Зведення оцифровуванням формально необмеженої кількості інформації (але як було показано – обмеженої рівнем шумів та частотним діапазоном) до скінченної кількості обмежених відліків дозволяє надалі застосувати програмну обробку інформації, яка виконується операція за операцією для кожного відліку окремо. Таким чином обробка інформації розгортається у часі як дискретна послідовність скінченної кількості дій. Одразу зауважимо, що замість багаторівневого відтворення кожного з відліків після квантування може бути застосовано його кодування у сигнал з меншою кількістю

рівнів квантування. Зрозуміло, що зменшення кількості рівнів повинне для збереження кількості інформації бути скомпенсоване використанням декількох відліків у модифікованому сигналі замість одного у попередньому. Історично склалось (через технологічну зручність та максимальну захищеність від завад) використання перекодування сигналу в бінарний, при якому задаються способи завдання (стани пристроїв) тільки двох чисел (0 та 1), а складніші числа задаються певною послідовністю таких значень. Одне двійкове значення отримало назву ***біт*** та стало одиницею кількості інформації в комп'ютерних технологіях.

Із врахуванням двійкового кодування інформації (відповідно до принципів фон Неймана) зазвичай її кількість визначають кількістю бітів, необхідних для запису або передачі. Із врахуванням традиційних для інших галузей науки і техніки префіксів, які означають множення на 1000 (кіло-), 1000000 (мега-) тощо, в інформатиці вводять аналогічні множники, найближчі за значенням до цих, але такі, що є цілим степенем двійки, тобто Кбіт (1024 біт), Мбіт (1048576 біт) тощо. Оскільки крім надто дрібного біту часто вводять більш велику одиницю кількості інформації – байт, який відповідає 8 бітам, аналогічним чином задають:

$$1\ Кілобайт\ (Кбайт) = 1024\ байт = 2^{10}\ байт,$$

$$1\ Мегабайт\ (Мбайт) = 1024\ Кбайт = 2^{20}\ байт,$$

$$1\ Гігабайт\ (Гбайт) = 1024\ Мбайт = 2^{30}\ байт,$$

$$1\ Терабайт\ (Тбайт) = 1024\ Гбайт = 2^{40}\ байт.$$

1.4.2. Представлення числа у комп'ютері

Технічно двійкове кодування пов'язане з тим, що пристрій обробки може бути апаратно реалізований на основі ключових (бістабільних) елементів, що полегшує розв'язання багатьох технічних проблем. Але про особливості апаратної реалізації будемо розмовляти пізніше, у главі 5, а зараз просто обмежимось використанням принципу бінарного кодування.

Для розуміння того, як закодовані бінарними відліками числа представляють більш складну інформацію, згадаємо таке поняття, ***як системи числення.*** За свою історію людство придумало досить велику кількість способів кодування чисел у вигляді символьного представлення, які і є системами числення. Основу таких систем складають спеціальні зображення-цифри, кожна з яких є формальним замінником деякого додатного цілого числа. Зрозуміло, що діапазон

можливих значень, з якими зазвичай оперує людина, значно перевищує кількість окремих символів, які може розпізнати та запам'ятати мозок. Відповідно, людство прийшло до потреби кодування довільного числа комбінацією цифр.

Таким чином, якщо задати деяку множину цифр N_0, N_1, … N_m, число у найпростішому варіанті кодування визначається як сума цифр (точніше чисел, які відповідають цифрам) помножена на натуральні вагові множники, які можуть кодуватись кількістю k_i однакових цифр

$$K = \sum_i N_i k_i .$$

Прикладом подібної системи є римське кодування. Особливістю такого представлення є те, що число, яке відповідає цифрі, не залежить від її положення, хоча у римській системі є певні обмеження на порядок слідування цифр – префіксне, тобто попереднє, положення використовується як від'ємне значення, наприклад IX означає 9, тобто 10-1. Системи подібного типу називаються непозиційними. Їх основним недоліком є незручність алгоритмізації дій над числами, які записуються декількома цифрами. Саме можливість масштабування алгоритму операції над парою цифр на числа з довільною кількістю цифр забезпечили розповсюдження ***позиційних систем числення***, у яких одна і та ж цифра у записі числа набуває різних значень залежно від своєї позиції (розряду). Історично склалось (через відповідну кількість пальців на руках), що широкого вжитку отримала десяткова система, тобто така, яка за основу бере значення 10. Відповідно, основа позиційної системи числення означає кількість цифр у системі. Довільне додатне ціле число записується як сума множників-розрядів (саме вони кодуються цифрами), помноженими на степінь основи *N*, а цій степені відповідає позиція розряду:

$$X = x_m * N^m + x_{m-1} * N^{m-1} + ... + x_2 * N^2 + x_1 * N^1 + x_0 * N^0 ,$$

що зазвичай записується як $x_m x_{m-1} ... x_2 x_1 x_0$.

Для представлення від'ємних чисел використовується унарний префіксний мінус. Дуже важливим є й те, що дану суму можна продовжити на від'ємні степені, що й дозволяє записувати довільну дробову частину, тобто крім цілих чисел задавати дійсні. Наприклад для десяткової системи

$$24.173 = 2*10^1 + 4*10^0 + 1*10^{-1} + 7*10^{-2} + 3*10^{-3} .$$

Замість основи 10 можна використати будь-яку іншу. Для комп'ютера найзручнішою виявилась бінарність, відтак основою є двійка. Наприклад,

$$10011 = 1*2^4 + 0*2^3 + 0*2^2 + 1*2^1 + 1*2^0 = 1*16 + 0*8 + 0*4 + 1*2 + 1*1 = 19.$$

Для дробної частини можна використати, як і для десяткової системи, від'ємні степені

$$0.1101 = 1*2^{-1} + 1*2^{-2} + 0*2^{-3} + 1*2^{-4} + ...$$

А довільне дійсне число може бути представлене як сума цілої та дробової частин

$$10011.1101 = 10011.0 + 0.1101.$$

Основним недоліком бінарної системи числення є те, що запис у такій системі призводить до великої кількості цифр у числі. Як деякий компроміс можна за основу взяти більше число, яке є цілим степенем двійки. В цьому випадку один розряд модифікованої системи буде відповідати декільком розрядам двійкової. Через наближеність за ціною розряду до десяткової певне розповсюдження отримала вісімкова система (три бінарних розряди як нова основа), але з часом її витіснила шістнадцяткова (чотири бінарні розряди як основа). Слід зауважити, що подібні системи є виключно більш зручним (для людини) способом запису у символьній формі і принципових відмінностей в порівнянні з двійковою системою та її апаратною реалізацію у пристрої обробки вони не мають.

Відповідно, вісім бінарних розрядів (дві тетради, які відповідають двом шістнадцятковим розрядам) створили над бітом, як мінімальною кількістю інформації (0 або 1), наступний рівень – байт. Зрозуміло, що при цьому байт має діапазон можливих значень 0-255. Ще раз повторимо, що для комп'ютера при умові кратності довжини даних до байту різниці між шістнадцятковою та двійковою системами немає взагалі. Для людини для використання шістнадцяткового запису визначаються додаткові цифри поза межами звичних десяткових (табл. 1.1.), а довільне значення байту записується як пара таких шістнадцяткових цифр.

Таблиця 1.1. Цифри шістнадцяткової системи числення

Шістн.цифра	0	1	2	…	8	9	A	B	C	D	E	F
Значення	0	1	2		8	9	10	11	12	13	14	15

Алгоритм виконання арифметичних дій над двійковими багаторозрядними операндами є аналогічним до знайомого зі школи алгоритму для десяткової системи. Але для зручності реалізації суматора на апаратному рівні для запису знакових чисел у комп'ютері використали деяку відмінність від того, як це робиться використанням

звичного унарного мінуса. Приклад такого кодування наведений у табл. 1.2.

Таблиця 1.2. Завдання знакових цілих доповняльним кодом

№	Двійкове кодування	Десяткове кодування
1	00000111	7
2	00000110	6
4	00000101	5
5	00000100	4
6	00000011	3
7	00000010	2
8	00000001	1
9	00000000	0
10	11111111	-1
11	11111110	-2
12	11111101	-3
13	11111100	-4
14	11111011	-5
15	11111010	-6
16	11111001	-7

Принцип викладеного у таблиці так званого доповняльного кодування визначається тим, що якщо до числа з довільного рядка таблиці додати 1, отримаємо код, що стоїть у рядку над даним. При цьому єдиною особливістю буде те, що для рядка з умовним номером 10 (досягнення нуля) буде відкинутий дев’ятий бінарний розряд. Це означає, що суматор, спроектований для додатних цілих чисел, при такому кодуванні буде виконувати дії зі знаковими числами. Відповідно, байт дозволяє записати знакові числа від -128 (10000000) до 127 (01111111).

Дещо складнішою є ситуація з записом у комп’ютері дійсних чисел. Класичний спосіб запису таких чисел з позиціюванням роздільника дробової частини (комою, у англомовних країнах – крапкою) є незручним для апаратної реалізації та обмеження динамічного діапазону, тобто співвідношення найбільшого та найменшого чисел, які можна записати певною обмеженою кількістю цифр. Але ще задовго до комп’ютерної ери було запропоновано цікавий спосіб запису дійсних чисел, який знайшов своє використання і в даному випадку. Цей спосіб запису спирається на представлення довільного дійсного числа у вигляді мантиси з фіксованим

положенням роздільника дрібної частини, наприклад перед першою значущою цифрою, та степенем основи, яка записується як окреме поле деяким цілим числом.

Наприклад,

$$-345600 = -0.3456 * 10^6 .$$

Зауважимо, що якщо додатково до домовленості про положення коми/крапки у мантисі діє домовленість і про кількість розрядів мантиси, зайві символи типу коми або позначки основи у записі можна опустити. Таким чином, наведене число може бути скорочене до запису -34566. У цьому прикладі ми обмежили кількість розрядів мантиси значенням 4. Такий скорочений запис є незручним для людини, але для бінарного запису у комп'ютері є прийнятним і ефективним.

Відповідно, для кодування дійсного числа загальній бітовий блок розділяється на три частини (рис. 1.6.), розподіл за довжиною яких залежить від загальної кількості байтів, які виділяються на таку комірку (табл. 1.3.). У даному прикладі наведено число типу double, яке для дійсного числа вважається зараз основним форматом.

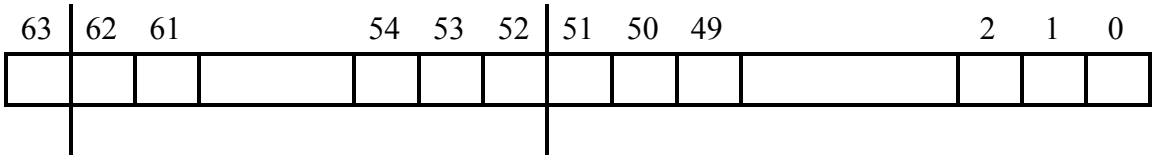


Рис. 1.6. Розподіл бітів за змістом для збереження дійсного числа

Таблиця 1.3. Структура формату запису дійсного числа для типів різної довжини

Тип		Номера бітів		
	Загалом бітів	Знак мантиси	Зсунутий порядок	Мантиса
single	32	31	30-23	22-0
double	64	63	62-52	51-0
long double	80	79	78-64	63-0

Для знаку мантиси в даному випадку виділяється окремий біт, а запис знакового цілого, яке відповідає степені двійки, тобто порядку, для потреби врахування знаку записується зсунутим кодом.

Динамічний діапазон дійсного числа визначається кількістю бітів, що припадає на запис порядку, а точність – бітовою довжиною мантиси. Відповідно, основним призначенням більш довгого формату дійсного числа є саме збільшення точності.

1.4.3. Типізація даних

Інформація не залежить від того, якими даними і як вона закодована, але форма представлення інформації може бути зроблена більш зручною для обробки. Саме з цією метою визначається деякий набір типів даних. Зазвичай набір цих типів відповідає традиційним методам роботи. Для рахування або нумерації використовуються цілі числа, а дійсні числа забезпечують роботу з природними неперервними величинами. Мовні потреби обслуговують літерні типи, методи формальної логіки – булевий тип (рис. 1.7).

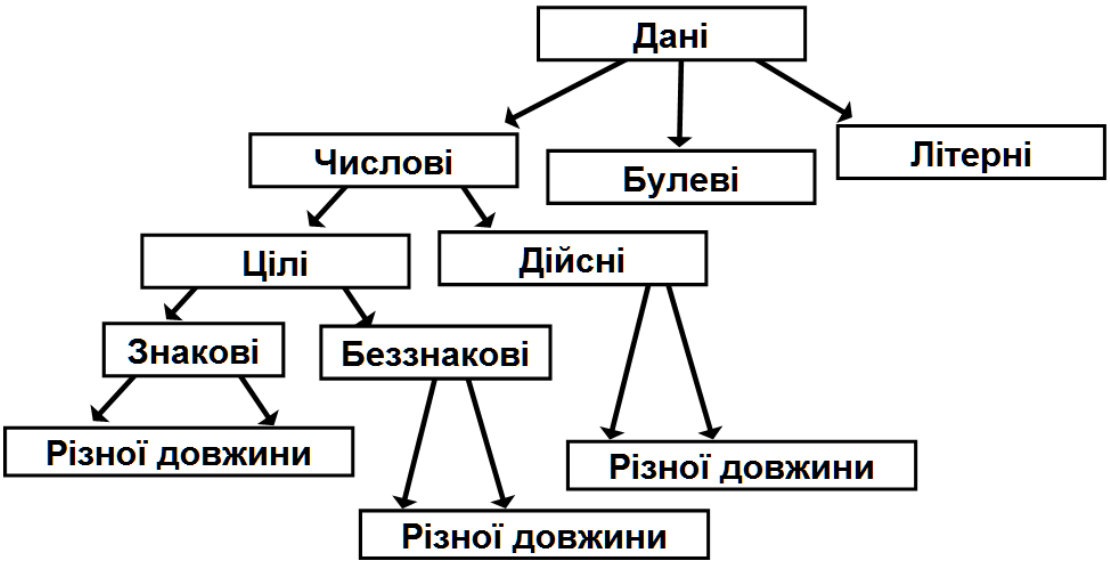


Рис. 1.7. Основні типи даних

Як вже зазначалось, числові типи можуть бути використані різної довжини. Для цілих типів довжина поля визначає динамічний діапазон числа, яке може бути збережено у цьому полі. Для дійсного, як було зазначено в п. 1.4.2, довжина поля визначає точність. Оскільки при виконанні дій з дійсними даними фактично відкидаються «зайві» розряди (які не вмішуються у формат запису числа), більше обмеження за довжиною мантиси, вноситься деяку похибку при виконанні окремих операцій. Через те, що результат операції у подальшому стає операндом, такий ланцюжок дій призводить до ***накопичення похибки***. Саме з цієї причини виникає потреба

подовжити мантису. З іншого боку для запису зайвих розрядів числа доводиться збільшувати розмір комірки.

В залежності від реалізації АЛП збільшення довжини числа може призводити до сповільнення виконання операції за рахунок потреби її розбиття на декілька послідовних операцій з більш короткими числами. Одразу зауважимо, що зазвичай сучасні універсальні процесори мають апаратну реалізацію для достатньо великої довжини даних, як цілих (64 біти), так і дійсних (80 бітів), тобто така проблема існує виключно на більш обмежених обчислювальних пристроях. Але саме для оптимізації довільної програми є бажаним підтримувати комірки, тобто і типи, різної довжини, що зазвичай реалізується в обчислювальних засобах.

Для кодування літерних даних за основу беруться комірки, що зберігають одну літеру. Традиційно для цього виділяється байт, але за сучасних технологій можуть використовуватись два байти.

Булевий (логічний) тип має два значення (істина/неправда), відповідно, для цього є достатнім одного біту. Однак комірка робиться більшого розміру і фактично є цілою коміркою з обмеженими властивостями. Але у багатьох випадках ціла комірка (певна кількість байтів) розглядається як набір однобітових булевих значень, що, наприклад, є корисним при програмуванні мікроконтролерів.

Можуть використовуватись і більш складні типи, які є об’єднанням деякого набору зазначених простих типів. Але уся різноманітність типів має деякі спільні властивості. Для запису елементу даних довільного типу у загальній пам’яті комп’ютера треба створити деяку комірку, яку можна виділити серед інших. Таким чином, створення комірки для збереження даних фактично означає завдання:

- адреси початку блока даних, тобто комірки;
- довжини комірки, наприклад у байтах;
- способу інтерпретації даних.

Відповідно, зазвичай використовують ***статичну типізацію***, тобто збереження типу протягом усього часу існування комірки даних. В принципі можна забезпечити і деякий спосіб створення нетипізованої комірки, тобто з можливістю зміни типу в процесі існування. Цей варіант є дещо складнішим і ресурсоємнішим для обробки, хоча у певних випадках використовується. Відповідно, певні пізніше розроблені мови високого рівня використовують ***динамічну типізацію***. У цьому випадку тип комірки змінюється відповідно до значення, яке у поточний момент туди записується.

В мовах програмування за аналогією з математикою ***комірка для збереження даних називається змінною***, враховуючи, що в процесі роботи з даними значення у цій комірці може бути змінене. Для ідентифікації такої комірки використання числової адреси є незручним для людини. Через це до цієї адреси прив'язують деяке умовне ім'я – ідентифікатор, яке протягом виконання усієї програми, або, як мінімум, певного фрагменту програми (коли це не порушує умову однозначності ідентифікації), є унікальним.

1.5. Алгоритм

1.5.1. Визначення алгоритму та його властивості

Відповідно до потреби реалізації програмного засобу для обробки інформації виникає необхідність введення поняття алгоритму. Він визначає послідовність дії цієї обробки і реалізується як програма. Але для програмного засобу представляє інтерес не будь-яка послідовність дій, а така, яка має сукупність певних властивостей. Є багато варіантів визначення поняття алгоритму, які так чи інакше визначають таку сукупність властивостей.

В загальному розумінні під алгоритмом розуміють систему правил виконання деякого процесу, що призводить до результату за обмежений час. З прив'язкою до обробки інформації можна уточнити таке визначення як точний опис виконання в зазначеному порядку операцій з деякої системи, що забезпечує розв'язання задач певного класу за лічену кількість кроків. Із врахуванням сучасних тенденцій розвитку поглядів на обробку інформації будемо вважати визначенням таке твердження:

Алгоритм – система правил виконання обчислювального процесу, що обов'язково приводить до розв'язання певного класу задач після скінченного числа операцій.

Відповідно до визначення, до основних властивостей алгоритму можна віднести:

- детермінованість (у будь-який момент часу однозначно визначається стан);
- результативність (обов'язкове завершення результатом);
- завершуваність (обмеженість кількості кроків);
- масовість (застосовуваність до довільного набору даних в рамках певної задачі або класу задач);
- зрозумілість (використовуються тільки команди, зрозумілі виконувачу);

- елементарність (кожен крок алгоритму має бути простим, елементарним, можливість виконання якого не викликає сумнівів);
- дискретність (розчленованість процесу виконання алгоритму на окремі кроки).

Вважається, що алгоритм не має помилок, якщо він забезпечує результат при будь-яких допустимих наборах даних. Якщо ж при деякому наборі даних алгоритм призводить до неправильного результату або не дає результату взагалі, він вважається таким, що має помилки. Зрозуміло, що для алгоритму часто є потреба визначити обмеження на дані, для яких він може бути застосований.

Для довільної задачі може існувати багато різних алгоритмів, що забезпечують досягнення результату. При цьому результат, звичайно, не повинен залежати від того, який саме з можливих алгоритмів було застосовано. Але різні алгоритми, або навіть той самий алгоритм при певних наборах даних потребують різної кількості кроків (часу виконання) або інших ресурсів.

В загальному випадку ефективність алгоритму можна визначити за кількістю усіх потрібних для отримання результату ресурсів для даного класу задач із врахуванням усереднення (зваженого усереднення) по можливим наборам даних.

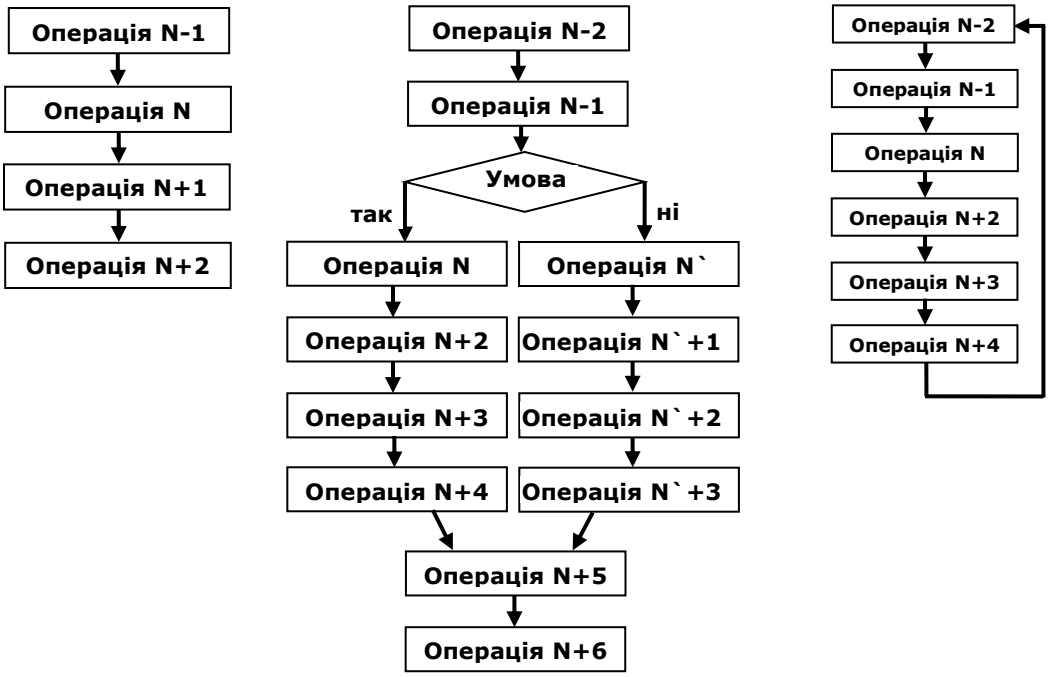
Зазвичай прагнуть досягти максимальної ефективності обробки інформації і приділяють увагу оптимізації існуючих та пошуку більш досконалих алгоритмів.

1.5.2. Класифікація алгоритмів

Алгоритми ділять на декілька базових класів. Найчастіше визначають такі:

- лінійний алгоритм;
- розгалужений алгоритм;
- циклічний алгоритм;
- рекурсивний алгоритм;
- паралельний алгоритм;
- стохастичний алгоритм.

Лінійним називається алгоритм, у якому операції виконуються послідовно, одна за одною. Таким чином, він відтворює найбільшу частину коду програмних засобів, оскільки визначає послідовний принцип виконання окремих елементарних дій. Формально виконання кожної наступної операції починається тільки після завершення попередньої (рис. 1.8.а).



а б в

Рис. 1.8. Основні різновиди алгоритмів

При виконанні обробки інформації часто виникає потреба реалізувати розгалуження виконання, тобто в залежності від стану даних на деякому кроці виникає потреба перейти до одного або іншого шляху продовження (рис. 1.8.б), що є розгалуженим алгоритмом. Також може виникати потреба виконання фрагменту алгоритму певну кількість разів (рис. 1.8.в). Такий алгоритм називається циклічним. Ці типи алгоритмів можна вважати класичними, але нещодавно крім них стали активно розвивати деякі інші. Так, рекурсивним називається алгоритм, при якому виконується звернення до себе. Наприклад, факторіал довільного числа можна реалізувати через факторіал числа, на одиницю меншого за дане.

$$n!=\begin{cases} n\times(n-1)! & при\ n>0 \\ 1 & при\ n=0 \end{cases}$$

Також активно розвиваються алгоритми, у яких не завжди виконується правильна традиційна послідовність виконання за рахунок одночасності виконання декількох команд (паралельні алгоритми) або стохастичного характеру вибору шляху продовження при розгалуженні (стохастичні алгоритми).

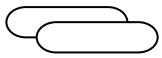
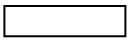
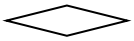
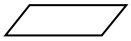
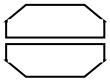
1.5.3. Опис алгоритмів

Для формального представлення алгоритмів, наприклад для публікації або формулювання завдання програмісту-кодувальнику, можуть використовуватись різні засоби. Алгоритм можна описати:

- звичайною мовою;
- спеціальною мовою описів;
- однією з мов програмування;
- блок-схемою.

Опис звичайною мовою є незручним і погано формалізованим, відповідно, його зазвичай не використовують. Спеціальна мова опису алгоритмів використовується у публікаціях. Така мова за основу бере синтаксис однієї з мов програмування. При цьому використовуються максимальні спрощення (не використовуються складні синтаксичні конструкції, типи різної довжини тощо). Такий опис є досить зручним для реалізації мовою, яка є прототипом обраної мови опису, але не є найкращим для швидкого сприйняття або при використанні інших мов програмування. Найбільш зручним способом опису алгоритмів є блок-схема, тобто сукупність графічних символів, уточнених текстовими вставками. Стандартизовані позначення на блок-схемах алгоритмів наведені у табл. 1.4.

Таблиця 1.4. Геометричні позначення основних типів елементів блок-схеми

	Термінатор (початок, завершення)
	Загальна операція або блок
	Умова галуження
	Обробка вводу/виводу
	Цикл (початок і кінець)

Видно, що окремі графічні символи використовуються для найважливіших елементів блок-схеми. Найбільш універсальними блоками є прямокутники, які відповідають більшості кроків алгоритму. Наприклад, у таких блоках вказуються розрахункові (арифметичні) вирази. Для виділення початкового та завершальних

кроків (завершень в алгоритмі може бути декілька!) виділяються термінатори. Через особливу важливість галужень для розуміння зв'язків окремих частин алгоритму для даного елемента обрано окреме позначення. Цикли в алгоритмі можна позначити на основі галуження та декількох допоміжних загальних операцій. Часто так і роблять, але для зручності є спеціальна пара блоків, які позначають початок та кінець циклу.

Зауважимо, що блок-схема може мати різний рівень деталізації. На початку історії розвитку інформаційних технологій було прийнято до безпосередньої реалізації програми на мові програмування робити детальну блок-схему, один блок якої відповідав одному оператору. Зараз такі детальні блок-схеми не дуже популярні і використовуються зазвичай для публікації алгоритмів. Але блок-схема не втратила свого значення, залишившись, наприклад на рівні макроблоків, з метою викладення загальної архітектури програмного засобу. Відповідно, класифікувати рівень деталізації блок-схем можна таким чином:

- на рівні модулів;
- на рівні виразів;
- на рівні окремих операцій;
- на рівні команд процесора.

На рис. 1.9. наведено блок-схему програми на прикладі обчислення факторіала.

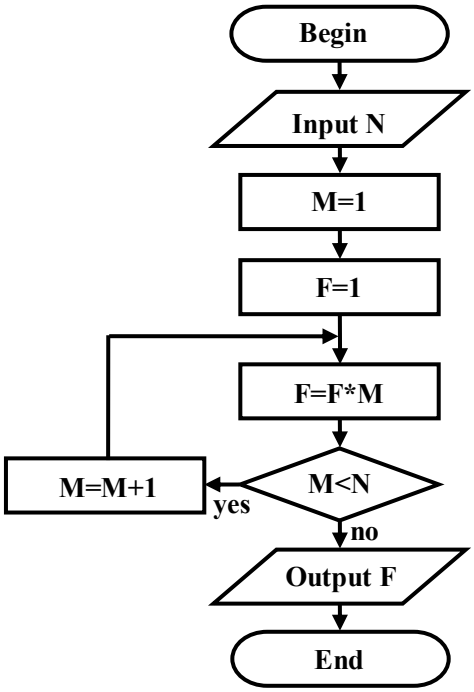


Рис. 1.9. Блок-схема програми на прикладі розрахунку факторіала

1.6. Питання для самоконтролю

1. Чому в телекомунікаційних технологіях для передачі інформації зазвичай використовують сигнали електромагнітної природи?
2. Чи є кодуванням інформації використання деякої людської мови?
3. Чи може бути достовірною неповна або неактуальна інформація?
4. Чи можна асемблерну програму, написану для процесору однієї архітектури перекомпілювати для використання на процесорі іншої архітектури?
5. Чи можна вважати графічний редактор системним програмним засобом?
6. Чи можна для написання тексту програми використовувати текстовий редактор загального вжитку (наприклад Notepad)?
7. В чому відмінність відладки програми від її тестування?
8. Чому оцифровування сигналу забезпечує безвтратність багаторазового копіювання (копія з попередньої копії)?
9. Чому шістнадцяткова система отримала широке розповсюдження в комп'ютерних технологіях?
10. Чи можна вважати алгоритмом деякий «спосіб множення 35 на 120» ?
11. Чи можна вважати алгоритмом опис дій для реалізації текстового редактора?

2. Комп'ютерна графіка та САПР

2.1. Загальні принципи реалізації комп'ютерної графіки

2.1.1. Використання зображень

У попередній главі було розглянуто основи комп'ютерних технологій для обробки деякої абстрактної (тобто без прив'язки до конкретного її виду та призначення) інформації. У даній главі розглянемо приклад використання комп'ютерних технологій для роботи з деяким конкретним видом інформації, яка займає одне з чільних місць для інженера.

Загалом діяльність людей (господарча, культурна, інженерна або наукова) значною мірою пов'язана з передачею інформації від однієї людини до іншої. Ця передача повинна бути максимально ефективною. Зрозуміло, що для її забезпечення слід використовувати ті канали надходження інформації, які в організмі людини є найбільш швидкодіючими. Порівняємо (оціночно) швидкість надходження даних за такими каналами, як слух та зір. З врахуванням частотного обмеження слуху у 20 кГц та теореми Котельникова, а також динамічного діапазону, який вимагає розрядності відліку 12 бітів, і наявності двох каналів (стерео) за секунду у вигляді звукового сигналу надходить 120 кБ даних. Реально кількість інформації, що надходить у мозок через цей канал у багато разів менша. З оцінкою зони чіткого зору 1000x1000 точок (реально – більше) 2 байтів на точку (чому саме 2, розглянемо трохи пізніше) і 10 зображень за секунду, що гарантує дискретність сприйняття) отримаємо 20 МБ, тобто маємо перевагу більш, ніж на два порядки! Крім того запис зорової інформації, необхідний для її довгочасного збереження, відпрацьований практично значно раніше і краще за запис звуку. Навіть запис звичайної мови (звук) людство реалізувало як сукупність нескладних зображень (літер).

Але літерне представлення інформації є лише деяким кодуванням мови, до того ж – надзвичайно надлишковим з точки зору інформатики і, відповідно, неефективним. На допомогу літерному опису ще в прадавні часи використали те, що з'явилось навіть раніше літерної спроби записати мову – малюнки. Вони з часом отримали риси сучасних креслень, тобто специфічний вид зображень – інженерну графіку. За визначенням ***інженерна графіка (креслення) – створення стандартизованих технічних рисунків, що***

виконуються фахівцями (інженерами, архітекторами тощо). Основні задачі інженерної графіки

- створення технічної документації (засіб наочного опису):
- інструмент проектування (саме інструмент інтелектуальної діяльності).

Відповідно її основною відміною є умова гарантії однозначності сприйняття інформації на зображенні людиною. Ця вимога призводить до потреби уніфікації (а надалі і ***стандартизації – юридичного закріплення уніфікації***) позначень, вказування розмірів та інших додаткових позначок, тощо. Крім того в інженерній графіці, яка пов'язана з передачею форми, з'явилось використання декількох видів, тобто зображення об'єкту з різних напрямків, що забезпечує відтворення для кожного елемента об'єкту усіх трьох просторових координат.

Як і уся інша графіка до початку використання комп'ютерних технологій, інженерна графіка спиралась на нанесення графічних елементів фарбою (або її замінником – графітовим або вугільним стрижнем), на деякому носії (дошці, пергаменті, папері). Комп'ютерні технології значною мірою успадкували попередній варіант інженерної графіки через реалізацію аналогів введення та візуалізації графічних елементів. Але не слід відноситись до комп'ютера, точніше відповідної комп'ютерної програми, тільки як до «цифрового кульмана» та «цифрового олівця», як засобів малювання. Комп'ютер значно розширює інструментарій проектування, за рахунок чого суттєво змінюється, стає швидшим та ефективнішим, сам процес проектування у будь-якій інженерній галузі. Крім того більш зручними в порівнянні з паперовим носієм стають засоби збереження та передачі даних.

Розглядаючи загалом ***комп'ютерну графіку, тобто використання комп'ютерних технологій для обробки та синтезу візуальної інформації***, крім інженерної графіки не можна оминути і художню графіку, сучасною основою якої є цифрова фотографія. В даному випадку комп'ютерна графіка забезпечує покращення характеристик зображення, отриманого традиційними оптичними методами, та синтез зображення з певними специфічними вимогами (від максимально подібного до фотографії до сюрреалістичного). Комп'ютерні технології роботи з інформацією в даному випадку також значною мірою наслідують добре розвинутий «докомп'ютерний» аналог. Не є в цьому плані виключенням і динамічна графіка (відео), яка використовує технології, які були напрацьовані у кінематографі та телебаченні, а саме розбиття відео потоку на кадри та представлення кожного кадру деяким растром.

Докладніше усе це буде розглянуто у наступних параграфах цієї глави.

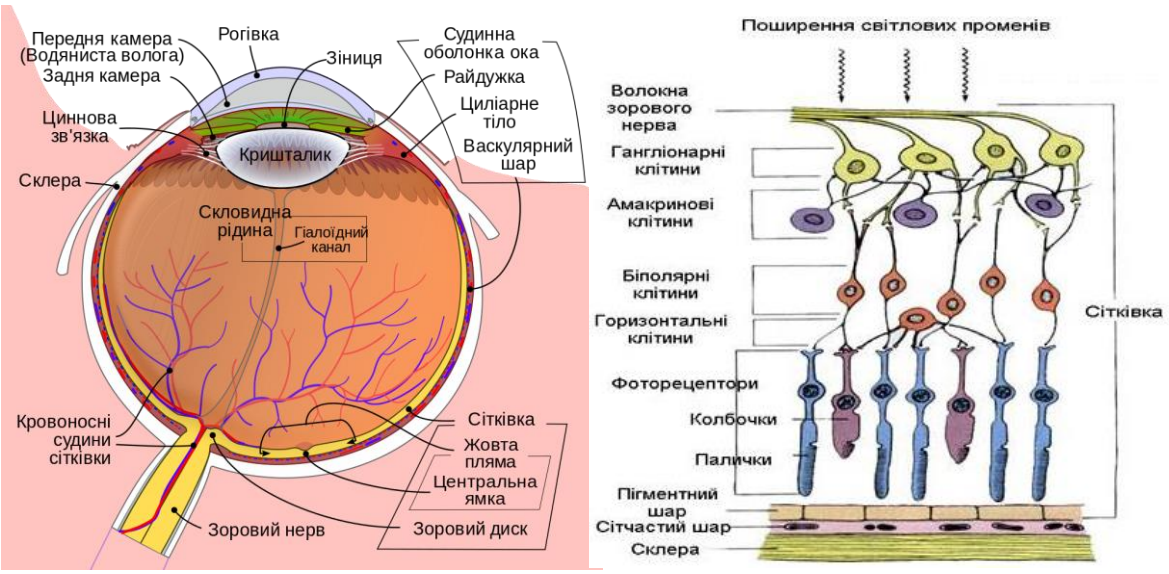
Нагадаємо, що являє собою оптичне зображення і специфічні риси такого просторового сигналу, оскільки у загальному випадку зображення, підготовлене для запису або передачі на відстань (не тільки при використанні комп'ютерних технологій), тісно пов'язане з поняттям зображення, яке дається саме в оптиці. Будемо виходити з положення, що оптичне зображення – результат проходження променів світла через оптичну систему. Відповідно кожній точці з простору об'єктів відповідає умовна точка перетину променів у просторі зображень (наближення геометричної оптики). Слід зауважити, що якщо простір об'єктів є тривимірним, то і простір зображення є також тривимірним. Однак зазвичай для запису, або введення для телекомунікаційних потреб у деякий радіоелектронний пристрій (в тому числу у комп'ютер) доводиться обирати у просторі зображень деяку обрану площину (площину кадру). На цю площину певним чином проеціюються усі точки тривимірного простору зображень. Таким чином формується плоске зображення, кожна точка (x,y) якого характеризується яскравістю світла, що на неї падає після проходження через оптичну систему. Відповідно отримуємо двовимірний просторовий сигнал і його запис, як результат дискретизації по двох координатах і квантування за яскравістю, може бути використаний, як комп'ютерне зображення. Але в даному випадку з розгляду випадає важлива характеристика – колір кожної точки зображення, тобто описаний підхід є придатним для некольорових зображень.

Колір точки зображення характеризує розподіл світла за довжинами хвиль, який детально показує, такий прилад, як спектрометр. Спектр світла (його розподіл по довжинах хвиль) є одновимірним сигналом. Формально для повного запису інформації, що відповідає точці зображення треба було б оцифрувати спектр і записати певну кількість значень замість одного (яскравості). Однак реально замість цього запис кольору точки зображення робиться значно меншим обсягом даних. Щоб зрозуміти ідею такого зменшення кількості даних розглянемо будову ока та особливості його функціонування, основну увагу звернувши саме на сприйняття кольору.

2.1.2. Сприйняття зображення людиною

Нагадаємо основні відомості про будову ока людини (рис. 2.1). Воно є досить складним оптичним пристроєм, який включає у себе

багатолінзову оптичну систему. Іноді як лінзу визначають тільки кришталик, але фактично до кришталика можна додати передню камеру та склисту рідину, яка заповнює об'єм між кришталиком та сітківкою. З точки зору формування оптичного зображення все є дуже схожим на інші оптичні прилади і особливої специфіки, яку можна використати для спрощення запису даних у комп'ютері для подальшого сприйняття оком, немає. А ось врахування наступного етапу перетворення сигналу є більш суттєвим.



а б

Рис. 2.1. Будова ока (а) та сітківки (б)

Датчиком світла в оці є специфічні клітини (нейрони) сітківки – ковбочки та палички. Палички відповідають за сприйняття слабкого сигналу, але не забезпечують розпізнавання довжини хвилі. Ковбочки є менш чутливими, але саме вони відповідають за сприйняття кольору, що забезпечується поділом ковбочок на три класи, кожен з яких має максимум чутливості на своїй довжині хвилі (рис. 2.2).

Якщо врахувати цю особливість сітківки, стає зрозумілим, що сприйняття кольору (саме не довжина хвилі, а умовне сприйняття) виконується по співвідношенню сигналів від трьох типів ковбочок. Якщо хвиля є монохроматичною, таке співвідношення однозначно відповідає певній довжині хвилі, але однозначність втрачається для сукупності вже декількох окремих довжин і, звичайно, для випадку широкосмугового спектру. Це обмеження дозволяє суттєво спростити і зменшити за кількістю даних (до трьох значень) запис кольорового сигналу на довільний носій або при введенні до електронного

пристрою або комп'ютера. Завдяки цьому запис кольору точки зображення у комп'ютері може бути обмежений трьома числами, кожне число умовно відповідає трьом окремим довжинам хвиль.

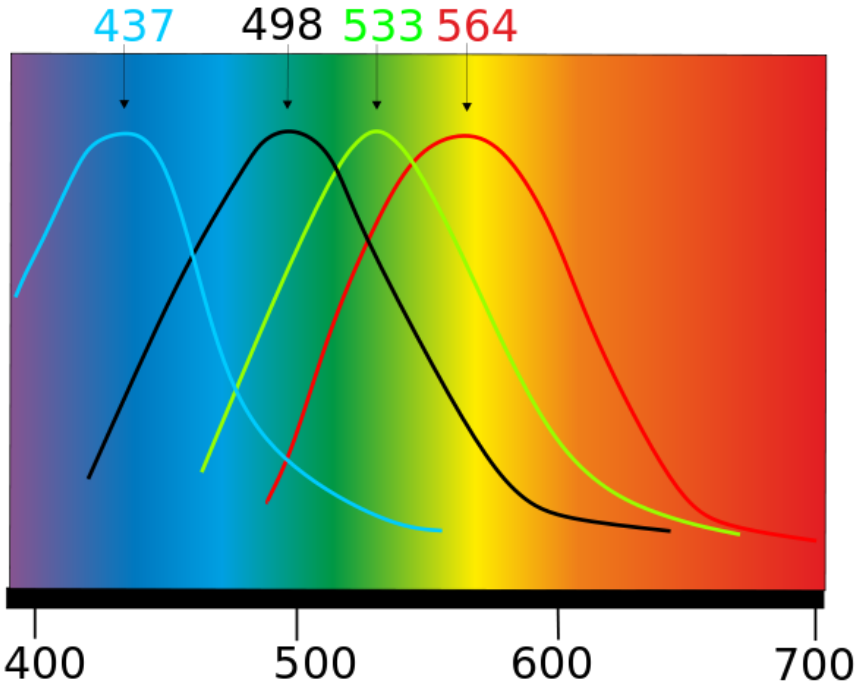


Рис. 2.2. Спектральна чутливість трьох типів ковбочок у порівнянні з спектральною чутливістю паличок (числа відповідають довжині хвилі у нм)

При виборі параметрів кодування зображення у комп'ютері слід врахувати динамічний діапазон ока людини. З врахуванням адаптивності зору до рівня освітленості за рахунок розширення зіниці (швидка адаптація) та зміною концентрації родопсину (повільна адаптація) динамічний діапазон ока сягає 12 порядків. Однак при відтворенні даних зображення немає потреби використання настільки великих чисел. Це пов'язано з тим, що «миттєвий» динамічний діапазон ока, тобто в межам одочасного сприйняття на одному кадрі є значно меншим і трохи більший за 100. Таким чином для відтворення кожного з трьох субкольорів із певним запасом достатньо одного байту, Таким чином колір кожного з відліків зображення задається трьома байтами.

Іноді для скорочення запису кольору використовують навіть меншу кількість даних. Для цього дані зображення зводяться до ***індексованого*** виду (або зображення з палітрою). ***Палітрою називають таблицю кольорів, яку формують для кожного***

окремого зображення занесенням у її рядки значень найбільш популярних на даному зображенні тріад. Для опису кольору точок у масиві даних самого зображення при цьому замість трьох байтів самої тріади використовується номер відповідного рядка палітри. Зрозуміло, що частину правильних тріад доведеться замінити на деякі наближені. Зазвичай розмір палітри вибирається рівним 256 або 64 К, що відповідно дозволяє записати значення одного відліку зображення одним або двома байтами замість трьох.

Обмеження кількості даних для динамічного зображення (як вже було зазначено, певної послідовності зображень-кадрів) також можна зробити виходячи із швидкодії ока. Це обмеження було досліджено ще при розробці технологій кінематографу (близько 25 кадрів за секунду), однак зараз для забезпечення високоякісного відео цю кількість підвищують до 50 - 60.

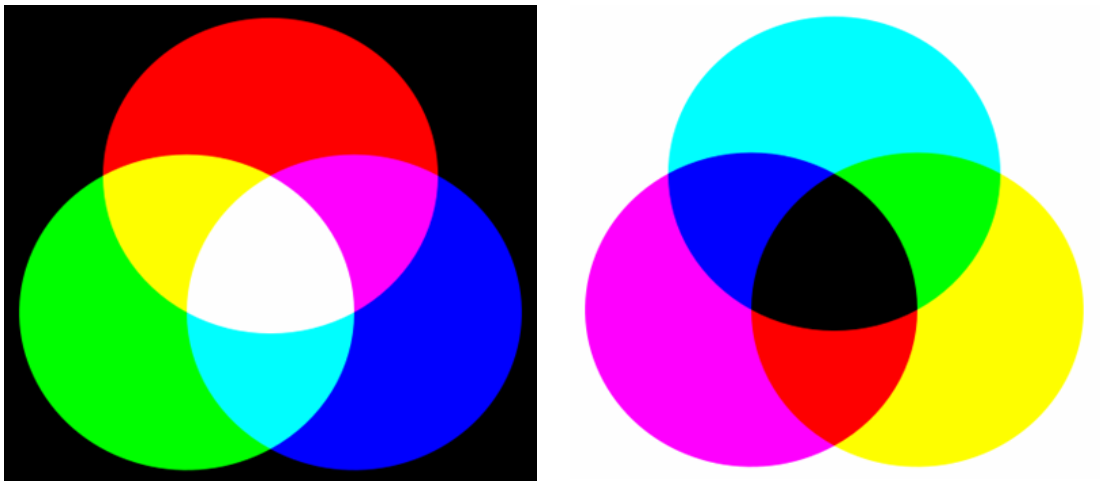
2.1.3. Види комп'ютерної графіки

Виходячи з особливостей сприйняття людиною статичного зображення усі «докомп'ютерні» технології запису та передачі зводились до двовимірної неперервної функції трьох умовних кольорів від координат кадру. Відповідно оцифровування таких функцій зводить дані до деякої матриці відліків (дискретної функції двох координат), кожен з яких фактично описує маленьку прямокутну (зазвичай квадратну) ділянку зображення, градієнтами в межах якої знехтували при виконанні оцифровування. Такий ***базовий елемент зображення отримав назву «піксель» (pixel, від pictures element).*** Загалом ***представлення зображення у вигляді матриці пікселів отримало назву растрової графіки.***

Відповідно до особливостей растрової графіки для відтворення зображень використовуються спеціальні пристрої, які представляють числові дані сукупності пікселів у формі двовимірної скалярної (не кольорові зображення) або векторної (кольорові) функції.

Одразу зауважимо, що набір чисел, які задають значення кольорового пікселя не є однозначним. Формально – довільна незалежна лінійна комбінація значень раніше обраної тріади базових кольорів формує новий базис, який є рівноправним по відношенню до попереднього. Це нагадує зміну базису (координатних осей) у звичному тривимірному просторі. Тому часто кажуть про кольоровий простір, у якому кожна з координат – значення одного з субкольорів. Для кожної конкретної задачі вибирається той кольоровий простір, який для цієї задачі є більш зручним.

Найчастіше використовується простір RGB (red – green – blue). Це пов'язано з тим, що найпопулярнішим пристроєм відтворення зображень є монітор, який додає кольорові складові до умовного чорного. Відтак ця тріада називається адитивною (тобто на основі додавання), рис. 2.3.а.



а б

Рис. 2.3. Представлення кольорів (а – RGB, б – CMY)

Однак, наприклад, для друку використовують інший базис – CMY (cyan, magenta, yellow) як спосіб віднімання від білого (паперу або іншої підкладки) світлофільтрами, якими фактично є фарба відповідних кольорів. Така тріада називається субтрактивною, рис. 2.3.б.

Альтернативою растровій графіці є ***векторна графіка – спосіб представлення зображення у вигляді сукупності простих геометричних об'єктів – примітивів.*** До таких примітивів можуть відноситись фрагменти ліній (відрізки, дуги, сплайни, багатокутники, тощо), замкнені лінії із заповненням внутрішньої області, прості зображення (літери). При цьому треба для кожного з об'єктів задати усі дані, необхідні для його побудови, тобто кожен об'єкт (екземпляр якогось з наперед визначених класів-примітивів) визначається повним набором атрибутів, до яких відносяться:

- координати, що задають положення та базовий розмір об'єкта (наприклад для еліпса – координати лівого верхнього кута та розміри прямокутника, описаного навколо еліпса);
- спосіб нанесення лінії (колір, стиль лінії, її товщина)

- спосіб (в найпростішому випадку – колір) заповнення внутрішньої частини лінії, що визначає границю об’єкту;
- для літерного примітива – характеристики шрифту та зміст напису.

Для візуалізації такий опис зображення є незручним і вимагає виконання операції растеризації, тобто розрахунку растрового зображення, яке є певним еквівалентом векторного опису.

Незважаючи на потребу растерізації для візуалізації векторна графіка має цілу низку переваг:

- компактність даних (за виключенням складних фотореалістичних зображень);
- якість масштабування та побудови ліній
- зручність редагування завдяки параметричному запису окремих об’єктів (фактично записується сукупність даних для послідовної побудови цих об’єктів на площині кадру).

До основних її недоліків слід віднести такі:

- не всі об’єкти можуть бути реалізовані;
- ресурсоємність растеризації;
- ресурсоємність переведення складного зображення у векторну форму (трасування).

Таким чином растрова і векторна графіка існують паралельно, доповнюючи одна одну. При цьому для запису даних конкретної задачі обирається більш ефективний для цієї задачі спосіб представлення даних. Якщо, наприклад, розглянути потреби інженерної графіки (креслення), саме об’єктність векторного запису та первинність числових значень та атрибутів визначають перевагу саме такого способу реалізації.

2.2. 3D графіка

2.2.1. Загальні принципи 3D графіки

Крім вже розглянутої 2D графіки, сучасні комп’ютерні технології важко уявити без реалістичного відтворення на екрані тривимірних об’єктів. ***3D графіка – сукупність прийомів та інструментів (як програмних, так і апаратних) для побудови двовимірної проекції тривимірної сцени (сукупності об’єктів)***. Відповідно до цього визначення для такої побудови (рис. 2.4) потрібен розгляд взаємного положення усіх елементів, що впливають на формування зображення у кадрі, тобто:

- площини кадру;
- точки спостереження (її зазвичай називають положенням камери) , через яку буде забезпечуватись проеціювання;
- сцени, а саме взаємного положення у просторі усіх об’єктів, що будуть проеціюватись у кадр;
- сукупності освітлювачів (пам’ятаємо, що зображення завжди є результатом проходження світла, навіть при моделюванні).

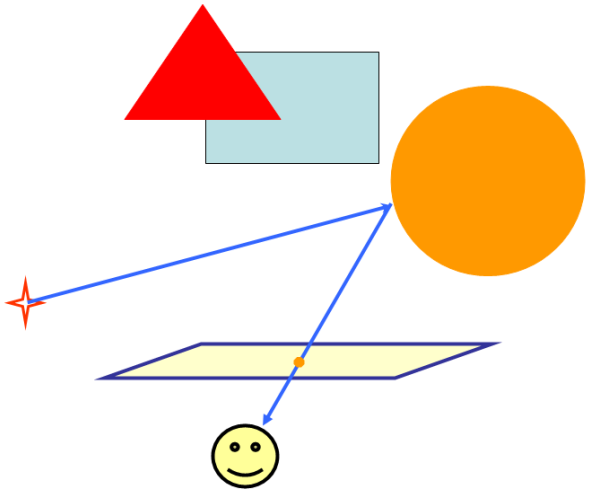


Рис. 2.4. Принцип побудови у 3D комп’ютерній графиці

Зазвичай задача розрахунку зображення розбивається на два етапи:

- розрахунок усіх складових сцени;
- ***рендерінг*** (відмальовування на основі проеціювання сцени у кадр з врахуванням особливостей поверхні та положення освітлювачів).

Зрозуміло, що в даному випадку використовується центральне проеціювання з центром, що співпадає з камерою на відміну від паралельного (для побудов в інженерної графіці).

При побудові кадру виникає ціла низка задач. Першою з цих задач є саме проеціювання окремих графічних елементів кожного з об’єктів сцени на площину кадру, розв’язання якої тісно пов’язане з тим, яким чином, тобто якою сукупністю даних, описуються самі об’єкти тривимірної сцени. Є два основних підходи такого опису:

- завдання усіх точок об’єкту у просторі;
- завдання поверхні об’єкту.

На перший погляд може здатись, що саме опис повного об’єму є кращим. Але в більшості випадків це не так. Якщо згадати дискретний

(піксельний) опис графічних об’єктів на площині і перенести ідею растру пікселів на об’єм, отримаємо кубічні об’ємні елементи зображення. Ці «кубіки» отримали назву вокселів (volum pixel, тобто об’ємний піксель). Воксельний опис іноді використовується, але досить рідко і в тих випадках, коли кількість пікселів у кадрі, який потрібно розрахувати, є не дуже великою. Щоб це зрозуміти, розглянемо умовний кадр з лінійними розмірами 1000x2000 пікселів (близько до найбільш поширеної кількості точок комп’ютерного монітора). Для якісного розрахунку такого кадру є потреба мати розмір вокселя менший за розмір пікселя, що вимагає $10^{11}-10^{12}$ вокселів для опису сцени. Отримуємо катастрофічно велику ресурсоємність обрахунків. Більш того просте проеціювання вокселів не дозволяє врахувати навіть нескладну модель освітлення, тобто врахування положення та характеристик освітлювачів, світло від яких відбивається від поверхні об’єкту та заломлюється, якщо він прозорий. До цього можна додати, що у випадку непрозорого об’єкта (а таких об’єктів значно більше за прозорі) більшість вокселів будуть під поверхнею і не повинні впливати на зображення у кадрі.

Виходячи з зазначених зауважень стає зрозумілим, що більш ефективним є саме завдання поверхні. Задача завдання поверхні також не є однозначною, оскільки вона може мати досить складну форму і не описуватись аналітичними формулами, тобто її доводиться задавати наближено. Найпростішим способом наближеного опису поверхні є тріангуляція (рис. 2.5).

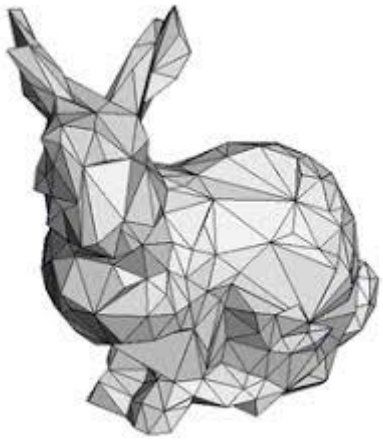


Рис. 2.5. Триангуляція поверхні

Вона спирається на те, що кривий фрагмент поверхні замінюється на плоску грань, для чого треба:

- задати певну кількість точок (***вершин***), які точно належать поверхні.

- з’єднати трійки вершин відрізками (***ребрами***);
- визначити ***грань***, що обмежується трьома ребрами (звідси і термін «тріангуляція»), оскільки три точки однозначно визначають положення площини у просторі.

Зрозуміло, що крім опорних вершин точки грані не співпадають с тими точками поверхні, наближенням яких вони є. І похибка зростає із зменшенням кількості вершин і, відповідно, кількості цих трикутних граней. Але ця похибка є контрольованою і з врахуванням потреб точності можна збільшити кількість граней (трикутників) поплатившись за це збільшенням ресурсоємності розрахунків. Ці грані часто називають полігонами (тобто багатокутниками), але ця назва все одно не виходить за використання саме трикутників. Альтернативою використання плоских граней є фрагменти аналітичних поверхонь більш високого порядку, що при збереженні точності дозволяє зменшити кількість вершин. Але такий опис поверхонь використовується відносно рідко.

Таким чином ми описали поверхню кожного з об’єктів сцени за допомогою тріангуляції, тобто як список трикутних граней. Наступною фазою загального розрахунку кадру є проеціювання цих граней на площину кадру (рис. 2.6).

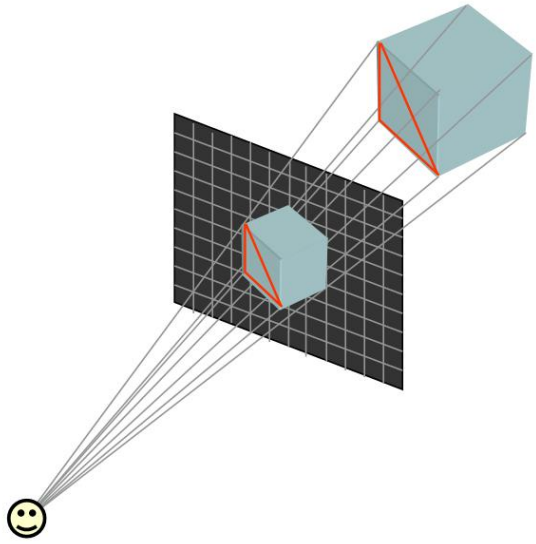


Рис. 2.6. Проціювання полігонів із простору сцени на площину кадру

З точки зору аналітичної геометрії ця задача є досить простою, однак у кадрі ми можемо побачити суцільний хаос відрізків. Це пов’язано з тим, що будуть спроеційовані усі ребра, навіть ті, які закриваються іншими гранями, тобто є невидимими із заданої точки

спостереження. Додається задача видалення невидимих поверхонь, які не повинні наноситись на площину кадру. Частково цю задачу можна вирішити відносно малими ресурсами за допомогою аналізу атрибутів граней із загального їх списку. Таке попереднє зменшення кількості даних є сенс робити, оскільки в більшості випадків обрахунки 3D графіки виконує зв'язка із двох пристроїв – центрального процесора (він виконує опис сцени та попереднє видалення невидимих елементів) і спеціалізованого графічного процесору, який виконує усі подальші розрахунки (рендерінг), забезпечуючи реалістичність відтворення сцени у кадрі. Для цього графічному процесору передаються усі дані, підготовлені центральним процесором. Відповідно зменшення навантаження на графічний процесор дозволить збільшити загальну швидкодію обрахунків.

Остаточне видалення невидимих елементів об'єктів, що проеціюються у кадр, найпростіше виконати вже при безпосередньому розрахунку пікселів кадру. Будемо виходити з того, що кожен з списку полігонів сцени проеціюється у кадр. При цьому точки полігону повинні перекрити інші точки, вже внесені у буфер кадру раніше, якщо вони в межах простору сцени знаходяться ближче до камери, ніж раніше оброблені. Для врахування цього використовується так званий z-буфер, який зберігає значення відстані від камери для кожного вже занесеного у буфер кадру пікселя. Відповідно, якщо нове значення відповідає елементу сцени, який є ближчим за те, що вже оброблено для цих координат пікселя у кадрі, нове значення вноситься як у кадр, так і в z-буфер. В іншому випадку точка відкидається.

В процесі рендерінгу відбувається проеціювання на площину кадру не тільки границь полігонів (ребер, а точок самих полігонів). При цьому в залежності від положення нормалі у точці поверхні, що обраховується, і напрямків на освітлювач та камеру з цієї точки повинен залежати результат відбиття світла. Тобто кожен елемент поверхні повинен бути описаний відповідно до деякої моделі освітлення, наприклад у найпростішому випадку коефіцієнтами дзеркального та дифузного відбиття. Якщо ці значення задати для трьох субкольорів окремо, отримаємо кольорову поверхню.

Для усіх точок полігону (грані) при врахуванні відбиття необхідно враховувати положення нормалі до поверхні в цій точці. З врахуванням того, що грань є плоскою, для усіх її точок нормаль, а відповідно і результат обрахунку відбиття, будуть однаковими. Крім того яскравість зображення поверхні буде мати розрив при переході з однієї грані до іншої. Для вирішення цієї проблеми використовується інтерполяція.

Є два найбільш популярних способи врахування моделі освітлення:

- затемнення по Гуро:
- затемнення по Фонгу.

При використанні метода Гуро модель освітлення розраховується один раз для кожної грані (в положенні умовного центру грані), а надалі використовується інтерполяція за значеннями виходячи з відстаней точки, що обраховується до найближчих центрів граней.

Метод Фонга забезпечує кращій результат, але є більш ресурсоємним.В цьому випадку інтерполюється напрямок нормалі, і для вже «правильного» положення нормалі у кожній точці враховується модель освітлення.

Зазначені методи затемнення дозволяють у кадрі задати яскравість (та колір) пікселів з врахуванням форми поверхонь об'єктів, що складають сцену, та положення освітлювачів у просторі, по кожному з яких проводиться окремий підрахунок з подальшим підсумовуванням. Зрозуміло, що найпростіше врахувати невелику кількість точкових освітлювачів. За потреби врахувати подовжений у просторі освітлювач, він розбивається на певну сукупність точкових.

2.2.2. Забезпечення фотореалістичності

Грубої реалістичності, яку забезпечує розглянутий, простий, спосіб реалізації 3D графіки цілком достатньо, наприклад, для відтворення результатів моделювання та відображення поточного стану 3D проекту в системах проектування. Але для багатьох задач комп'ютерної графіки є потреба значно якіснішого відтворення сцени. Загалом до складових фотореалістичності можна віднести:

- детальність опису поверхні;
- точність моделі освітлення з точки зору врахування законів оптики;
- перспективні спотворення, тобто чим далі предмет, тим меншим від здається (автоматично виникають при центральному проеціюванні);
- відтворення властивостей поверхні (колір, шорсткість, матеріал, тощо);
- текстури, як додатковий спосіб завдання властивостей поверхні;
- тіні;
- об'ємні ефекти (заломлення, та віддзеркалення);

- ефекти постобробки.

Перші чотири пункти були розглянуті у попередньому параграфі. Реалізація врахування інших складових є значно складнішою та ресурсоємнішою.

Відтворення непрозорої поверхні з точки зору фізики практично повністю характеризується розподілом по довжинах хвиль (в спрощеному варіанті по значенням для трьох субкольорів) коефіцієнтів дифузного та дзеркального відбиття. Правда для дифузного відбиття це справедливо тільки для ідеально рівної поверхні, розсіювання від яких описуються законом Ламберта. Реальні поверхні мають складнішу мікроструктуру. Формально її можна описати тріангуляцією на рівні окремих мікрограней, що відповідають нерівностям, але це призведе до сповільненню розрахунків на декілька порядків, що є неприйнятним. Тому зазвичай для певних матеріалів наперед розраховують індикатрису розсіювання, що враховує статистику розподілу мікроскопічних нерівностей. Врахування наперед заготовлених характеристик спрощує остаточні розрахунки. Такий опис називають уточненою моделлю освітлення.

Ще одним способом спрощення відтворення особливостей поверхні використанням «домашньої заготовки» є нанесення на етапі рендерінгу текстури – ***даних про залежність характеристик поверхні від координат***. Найбільш традиційним способом завдання текстури є просте зображення – спеціально відредагований прямокутник з періодичним по координатах зображенням, що дозволяє трансляцією на період заповнити текстурою усю площину. Більш якісним варіантом, але більш критичним до об'єму пам'яті та швидкості передачі даних в комп'ютері, є використання фотографічних зображень достатньо великого розміру, щоб відмовитись від трансляції. Для збільшення швидкості рендерігнгу можуть бути попередньо підготовлені текстури різного масштабу, що дозволяє обрати найближчу за потребами для конкретного випадку (MIP mapping).

Ще одним способом реалізації текстури є рельєфне текстурування (bump mapping), тобто врахуванні під час рендерінгу даних про нерівності поверхні, менші за розміри полігонів.

До цього моменту були розглянуті тільки непрозорі об'єкти і їх оптичні властивості зводились виключно до відбивання світла. Прозорість об'єктів вимагає використання зовсім іншої моделі, яка враховує явище заломлення, а, відповідно, і проходження променів крізь товщу об'єкту. Звідси назва таких ефектів – «об'ємні». Зазвичай для створення такої моделі достатньо обмежитись наближенням

геометричної оптики, коли проходження світла не враховує хвильових властивостей світла і описується розглядом окремих променів. Одразу зауважимо, що крім заломлення подібним чином можуть бути і віддзеркалення одних об’єктів у поверхні інших. Для врахування подібних явищ використовується алгоритм трасування променів. Трасування променів зводиться до того, що світло від точкового джерела описується обмеженою (вимогами допустимої похибки) кількістю променів. При потраплянні на довільну поверхню кожен промінь розпадається на віддзеркалений, заломлений та певну кількість дифузно розсіяних. В результаті отримаємо величезне дерево променів, з якого тільки маленька частина проходить через точку спостереження (камеру).

Для зменшення ресурсоємності розрахунків використовують зворотне трасування, при якому розглядаються промені з камери, і проходять через усі пікселі кадру. З точки зору геометричної оптики переміна місцями джерела світла та точки спостереження не міняють хід променів. Надалі формується дерево зворотного їх проходження до джерел освітлення (рис. 2.7). Для економії часу трасуванням обробляється не уся сцена, а та її частина яка вимагає застосування цього алгоритму, тобто тільки окремі поверхні об’єктів сцени.

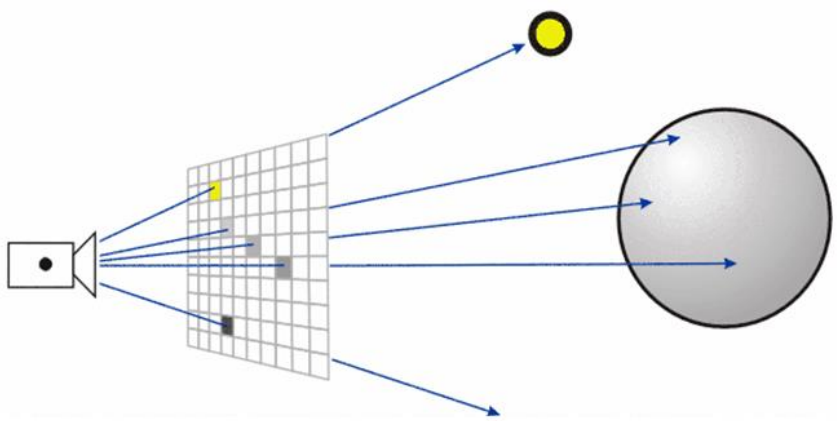


Рис. 2.7. Зворотне трасування променів

Для врахування тіней можуть використовуватись різні методи, такі як створення списку неосвітлених граней (не дає змогу розрахунку плавних напівтіней при наближенні освітлювачів точковими), накладання імітаційних карт освітлення, трасування променів.

2.3. Графіка в інженерії та автоматизація проектування

2.3.1. Перехід від традиційного креслення до САПР

Перш ніж розібратись із використанням комп'ютерних технологій, в тому числі графічних, для потреб інженерної графіки, розглянемо загальні особливості представлення даних в інженерній графіці. Як основний приклад розглянемо машинобудівну графіку, тобто проектування об’єктів складної форми та забезпечення правильної взаємодії об’єктів-деталей. При цьому задача проектування є найскладнішою в плані роботи з графікою, оскільки вимагає значної кількості геометричних побудов та багатоетапного редагування, а подальше представлення для виготовлення та експлуатації вже випливають з функціональності графіки, яка закладається для проектування.

Загалом проектування в інженерії зазвичай використовує ***декомпозицію – розбиття складної проектної задачі або складного об’єкту на сукупність більш простих***. Відповідно в інженерній графіці використовують розподіл виробів (будь-який предмет або набір предметів виробництва, що підлягають виготовленню) на такі:

- ***деталь*** - виріб, виготовлений з однорідного по найменуванню і марці матеріалу, без застосування складальних операцій;
- ***складальна одиниця*** - виріб, складові частини яких з'єднують між собою за допомогою складальних операцій (згвинчення, клепка, тощо);
- ***комплекс*** - два чи більше виробів, не з'єднаних складальними операціями, але призначених для виконання взаємозалежних експлуатаційних функцій;
- ***комплект*** - набір виробів, які мають загальне експлуатаційне призначення допоміжного характеру.

Розробка проекту зазвичай починається із окремих деталей, з який компонується більш високий рівень загального проекту. При цьому розробка наступної деталі часто вимагає урахування раніше розробленої з метою узгодження їх форми, в тому числі можлива потреба корекція раніше розробленої частини загального проекту (однієї і навіть декількох деталей). Таким чином простою ієрархією розробки справа не обмежується, викликаючи потребу багаторазової переробки раніше зроблених креслень.

Традиційне креслення деталі вимагає однозначного відтворення форми, прийнятне для реалізації певної технології виготовлення, тобто однозначне визначення будь-якого елемента поверхні деталі. На площині визначаються тільки дві координати з трьох просторових. Відповідно у загальному випадку простого перенесення (проеціювання) деталі на одну площину є недостатнім. Для розв'язання такої проблеми використовується проеціювання на дві чи більше площин (прийнято використовувати взаємно ортогональні площини). Для спрощення роботи використовується паралельне проеціювання, тобто проеціювання сукупністю паралельних прямих. ***Кожна з таких проекцій називається видом, а сукупність декількох видів, розташованих за наперед визначеною домовленістю називається епюром Монжа*** (рис. 2.8).

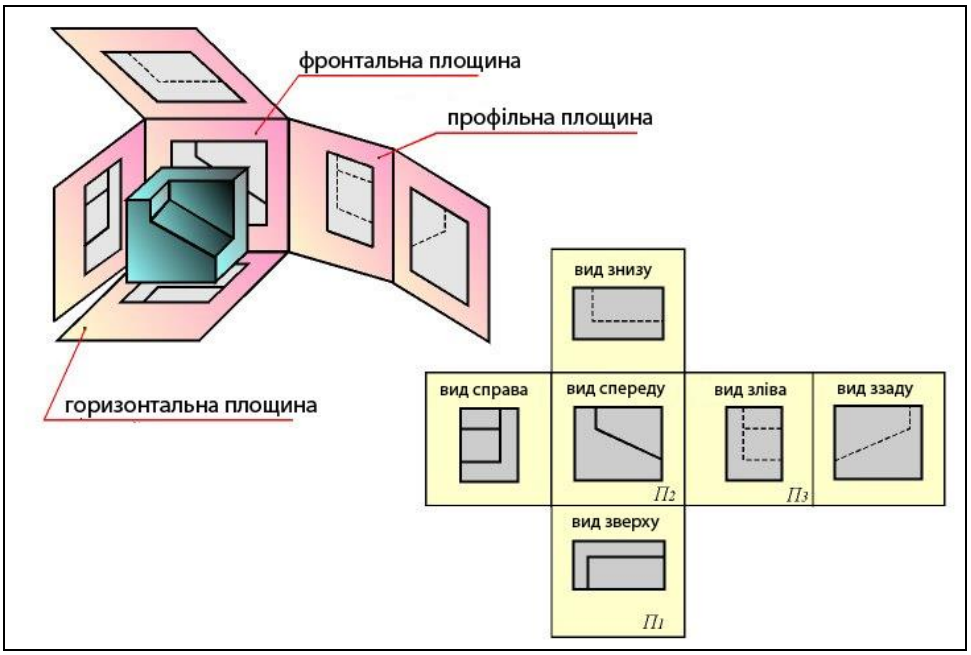


Рис. 2.8. Епюр Монжа

Границі окремих складових поверхні об'єкту, що відтворюється на кресленні, визначаються основними (товстими) лініями. Оскільки для виготовлення потрібно притримуватись певних заданих у проекті розмірів, до цих ліній додаються виносні та розмірні (тонкі) лінії та позначення розмірів числом. Історія розвитку креслярських інструментів зводилась до забезпечення зручного та якісного нанесення ліній потрібної товщини на поверхні паперу. Особливу увагу приділяли засобам реалізації відрізку (лінійка), кола (циркуль), кривих змінної кривизни (лекало), засобам нанесення паралельних ліній (рейсшина, косинці). З часом було реалізовано зручний універсальний інструмент – кульман, який дозволяв проводити

паралельні відрізки під довільним заданим кутом. Основними інструментами малювання став олівець (дозволяє наносити тимчасові лінії та легко редагувати проект) та рейсфедер (для чистового креслення, для якого виконується спеціальна фарба з сажі – туш).

Комп'ютерна програма, що реалізує формування та редагування графічної інформації (графічний редактор) дозволяє реалізувати значно складніші інструменти та шаблони для нанесення простих графічних елементів – примітивів (п. 2.1.3). Такі редактори орієнтуються зазвичай або на растрову, або на векторну графіку. Відповідно довільну лінію в комп'ютерній реалізації інженерної графіки зручно реалізувати як один з об'єктів в межах векторного представлення (п. 2.1.3). При цьому легко реалізується редагування проекту та збереження усіх розмірів як атрибутів лінії, не залежно від того – виводяться вони на зображення у вигляді додаткових шрифтових написів, чи ні.

Хоча реалізувати креслення з усіма вимогами стандартів, розроблених для «паперової» інженерної графіки, можна у довільному векторному графічному редакторі, зручність самого процесу інженерного проектування така програма, орієнтована саме на графічну складову, не забезпечує. Відповідно для потреб розробки інженерних проектів загалом і проектів, специфічних для окремих підгалузей інженерії (машинобудування, архітектури, електротехніки, радіоелектроніки) реалізуються спеціальні програмні засоби. Вони мають специфічні інструменти редагування та значно розширені атрибути елементів проекту та їх зв'язки між собою. Такі програмні засоби об'єднуються у такий клас, як ***САПР (системи автоматизованого проектування), англійською мовою CAD (Computer-aided design).*** Термін «автоматизований» підкреслює те, що даний засіб не замінює розробника, а відіграє роль сукупності інструментів для виконання рутинних дій в проектуванні.

Найпростішим використанням САПР для потреб інженерного проектування є режим «цифрового кульмана», коли використовуються саме графічні можливості.

Як приклад збільшення зручності роботи інженера розглянемо відносно нескладну задачу – побудову проекції кола, що знаходиться на площині, непаралельній до площини проекції. Правильною проекцією є еліпс, але ефективних креслярських інструментів для побудови правильного еліпсу немає. Відповідно були вигадані варіанти приблизної побудови за допомогою декількох дуг, що будуються циркулем. Така фігура у креслярів отримала назву «овал». Комп'ютер дозволяє задати і вивести на екран саме еліпс. Зазвичай у САПР для цього задаються чотири числа, які зводяться до координат

двох точок (двох діагональних кутів прямокутника, у який вписується потрібний еліпс).

Ще одним прикладом автоматизації рутинних дій є інструмент позначення розмірів, який за заданими людиною точками сам формує виносні та розмірні лінії, а значення розміру для створення напису автоматично береться із відстані між обраними точками. Є й багато інших рутинних дій традиційного креслення, які полегшує виконувати комп'ютерна програма, але їх краще розглядати в процесі практичного навчання. Відповідно в межах даного посібника, який є не замінником, а доповненням до лекцій та практичних завдань, їх наводити не будемо. Але слід зупинитись докладніше ще на одній принциповій різниці реалізації та використання САПР в порівнянні з «паперовими» технологіями.

При машинобудівному або архітектурному проектуванні виникає потреба паралельного проеціювання складних елементів на деяку площину. І навіть відносно проста задача перетину двох циліндричних поверхонь може викликати певні проблеми побудови. Необхідність вирішувати такі проблеми призвела свого часу до появи ***нарисної геометрії – набору алгоритмів, орієнтованих на використання людиною, для побудови складних випадків проеціювання***. Комп'ютерні технології, якщо САПР використовувати тільки як кульман, не позбавляє від потреби використання таких побудов, які можна охарактеризувати як «довічний головний біль кресляра». Слід згадати, що комп'ютер – в першу чергу є засобом розрахунків і координатне представлення усіх точок деталі в просторі забезпечує усі дані, потрібні для довільних розрахунків, в тому числі для довільного проеціювання. Платформою таких розрахунків є ***аналітична геометрія – алгебраїчний підхід до аналізу геометричних об'єктів, що задаються у координатному вигляді***. При цьому реалізація засобів використання аналітичної геометрії повністю покладається на розробників САПР і не стосується інженера, який використовує вже готовий програмний засіб. Але для того, щоб повністю задіяти переваги цього прихованого розрахункового підходу проект повинен виконуватись не на площинах видів, а в просторі. Саме тому використання САПР у режимі 3D (замість 2D, який і відповідає «цифровому кульману») є значно ефективнішим.

2.3.2. Реалізація графічної підсистеми у САПР

Програмний засіб класу САПР вимагає якісної реалізації графічної підсистеми. При цьому слід брати до уваги, що підгалузь

кожного САПР (п. 2.3.1) має специфіку роботи з даними. В даному випадку основний наголос будемо робити на проектування формоутворення, тому за основу беремо машинобудівні САПР.

Оскільки деталь проектується як просторовий об'єкт, є бажаним запам'ятовувати усі три координати кожного з її елементів, тобто використовувати, як вже зазначалось у п. 2.3.1, 3D режим розробки проекту. Окремі САПР при цьому дозволяють будувати 2D види традиційного креслення автоматично з такого проекту і навіть зберігають при цьому параметричний зв'язок, що дозволяє на онові вже 2D видів автоматично будувати розрізи та перерізи.

Загалом графічні інструменти САПР можна розділити на дві великі категорії:

- 2D (графіка на обраній площині);
- 3D (робота з повним набором координат у просторі).

2D інструменти повинні забезпечувати як мінімум:

- формування даних для нового графічного об'єкту (примітива), в тому числі прямим завданням даних у цифровому вигляді з клавіатури;
- перегляд та редагування усіх атрибутів примітива;
- коректне (в реальному масштабі часу) виконання растеризації усього списку доданих у проект примітивів для візуалізації поточного стану проекту.

3D інструменти у САПР зазвичай мають деяку специфіку, пов'язану із завданням більшості поверхонь з точними розмірами (на відміну від дизайнерських редакторів, які повинні задавати складні плавні форми, для яких художній образ не завжди узгоджується з технологічною доцільністю). Відповідно усі можливі тривимірні операції реалізуються як додаткові дії (вичавлювання, обертання тощо) по відношенню до двовимірного графічного об'єкту, який до цього будується на деякій наперед обраній площині (раніше реалізованій грані деталі або спеціально проведеній допоміжній площині). Ланцюжок з таких комбінацій (двовимірної побудови у площині та додаткової тривимірної дії) і реалізує повне формоутворення деталі. Зауважимо, що САПР у 3D режимі зберігає повну просторову інформацію, що дозволяє відповідний файл даних використовувати для виготовлення деталі пристроями з комп'ютерним керуванням.

САПР у 3D режимі повинен мати достатньо якісну візуалізацію об'єктів для інтерактивного перегляду деталі з різних боків, оскільки це суттєво допомагає конструктору. Така візуалізація повинна відбуватись у реальному масштабі часу, тобто час розрахунку наступного кадру (при динамічній зміні положення точки спостереження) не повинен бути більшим за час відтворення одного

кадру, тобто 1/25 – 1/30 секунди. Таким чином 3D режим значно ускладнює програмний засіб і робить його більш вибагливим до характеристик апаратного забезпечення. Реалістичність при цьому не повинна обов'язково відповідати іграм та стимуляторам, але певний мінімально достатній рівень повинен бути забезпечений (п. 2.2.1).

Зауважимо, що деякі САПР мають додатковий блок фотореалістичного відтворення з досить складним і ресурсоємним алгоритмом розрахунку, але він використовується як додаткова обробка даних для обраного положення спостерігача відносно площини кадру при завершенні проекту і особливого впливу на процес розробки не має.

2.3.3. САПР радіоелектронного призначення

Проектування радіоелектронних пристроїв стоїть трохи осторонь від машинобудівних або архітектурних проектів, для який основа проекту – формоутворення. Основними задачами радіоелектронного САПР є:

- розробка схеми з'єднань окремих радіоелектронних елементів
- формування топології розміщення елементів на платі та доріжок-з'єднань (розведення плати).

До цього може додаватись ще досить складна задача перевірки коректності роботи розробленої схеми (симуляція її роботи).

Такі задачі накладають інші вимоги до графічного модуля САПР. 3D режим стає необов'язковим і може використовуватись хіба-що для фінальної візуалізації зовнішнього вигляду зібраної плати, що не є надто критичним для проектування. Таким чином відносно проста 2D графічна система є цілком достатньою. Для такого САПР найбільш складними для реалізації є підтримка бази даних готових (серійних) елементів, з яких складається схема та автоматизація розведення плати.

База даних готових елементів може використовуватись і в машинобудівних та архітектурних САПР для використання у проекті готової номенклатури допоміжних деталей (щоб не розробляти їх ще раз). Але переважна частина проекту зазвичай є унікальною і має повний цикл розробки. Для радіоелектронного САПР наявність і повнота бази елементів є більш критичною, оскільки обрані елементи надалі будуть використані для складання пристрою і іншого джерела крім готових радіодеталей немає. У базі елементів повинні бути збережені усі дані для їх позначення на схемі, габаритні та установочні розміри для подальшого розміщення на платі.

Ще складнішою є задача автоматизації розведення плати. Її розв'язання може бути реалізовано як в автоматичному режимі (без втручання людини), так і в напівавтоматичному (розведення кожного окремого з'єднання за вказівкою конструктора та можливістю редагування).

Найбільш складною частиною радіоелектронного САПР є модуль симуляції роботи схеми (якщо його взагалі включено до програмного засобу). Цей модуль вимагає для кожного з елементів реалізації коректної математичної моделі, яка може бути використана як для статичного аналізу, так і в динаміці. Загалом для реалізації моделювання розробляються спеціальні комп'ютерні засоби, які зазвичай включають розвинену математичну обробку даних (часто в використанням деякої мови програмування), потужні засоби візуалізації результатів.

2.4. Моделювання у САПР

2.4.1. Загальні принципи моделювання

Однією з задач інженерії, в тому числі в галузі радіотехніки, є розробка нових пристроїв або систем, необхідних для діяльності людей. Кожна така розробка повинна пройти шлях від початкової ідеї до практичної реалізації, повністю працездатної, достатньо зручної та надійної, придатної до виробництва. Перевірити ці властивості можна на основі самої реалізації. Але виготовлення навіть одного дослідного зразка (а часто є потреба навіть у дослідній серії) вимагає значних витрат – використання обладнання, часу, фінансування, людських ресурсів тощо.

До цього можна додати такі проблеми, як випробування на час життя елемента чи пристрою (у звичайних умовах може вимагати років), надійність (потрібне накопичення певної статистики виходів з ладу, в тому числі з варіюванням умов експлуатації), випробування в агресивних умовах тощо. Тобто і сам процес випробування зазвичай вимагає значних витрат ресурсів і значно подовжує час до впровадження розробки. Будь-яка проблема, яка виявляється на етапі випробовування дослідного зразка, вимагає виправлення проекту та виготовлення нового дослідного зразка. Крім цього слід враховувати, що в сучасних умовах інженерія будь-якої господарської галузі пов'язана з дуже складними за структурою розробками і проектами, у яких часто задіяні новітні наукові ідеї, які ще не пройшли детальної апробації в інших проектах. Усе це вступає у протиріччя до вимоги

максимального скорочення часу від ідеї до впровадження у виробництво. З метою забезпечення конкурентноздатності розробка довільного проекту повинна проводитись у максимально стислий час. Для цього повинні застосовуватись методив швидкої перевірки характеристик елементів та систем, що розробляються в рамках загального проекту. Одним із потужних інструментів такої перевірки є ***моделювання – використання в процесі пізнання замість справжнього об'єкта спрощеного, але подібного до справжнього, який і називається моделлю.*** Загалом моделювання має досить довгу історію застосування в інженерії.

Зазначимо, що моделювання може бути конче необхідним не тільки під час розробки, а й при експлуатації виробів, які серійно випускаються та активно експлуатуються. Під час експлуатації зазвичай однією з проблем є виявлення причин виникнення аварійних ситуації та підготовка персоналу до дій у подібних ситуаціях. В разі аварії саме моделювання може допомогти з'ясувати можливі причини негараздів та відпрацювати зміни в проекті для запобігання їх повторення. Так само і тренування персоналу дешевше і безпечніше проводити з використанням імітаційного моделювання, ніж в екстремальних умовах з реальними системами.

Окремо зазначимо, що телекомунікаційні та радіоелектронні пристрої у сучасних умовах часто є важливими складовими дуже відповідальних систем. Відповідно, інженеру-радіотехніку необхідно орієнтуватись в методах та засобах, які використовуються для моделювання.

Загалом моделювання охоплює досить широкий спектр методів, які дозволяють забезпечити достатню точність аналізу поведінки на основі дослідження спрощеної моделі реального об'єкта. Єдиної класифікації видів моделювання не існує, але серед інших видів як найважливіші можна зазначити такі:

- фізичне;
- математичне.

Фізичне моделювання має глибокі традиції. Воно зазвичай спирається на створенні об'єкта, який за тими властивостями, що досліджуються, є аналогом справжнього об'єкта. Наприклад, відтворюються форми при аеродинамічних дослідженнях або використовуються масштабні (тобто меншого розміру) моделі елементів при дослідженні на міцність. У радіотехніці роль подібного моделювання часто відіграє макетування, тобто виготовлення пристрою з аналогів майбутніх елементів, але за технологією, яка не передбачає підготовки та запуску в роботу технологічних операцій, потрібних для серійного виробництва. При цьому макет може

збиратись за допомогою технологій навіть ресурсоємніших за серійні, але які не вимагають довгої підготовки обладнання, яке до того ж довелося б виводити з поточного виробництва. Але подібне моделювання часто є неможливим. Наприклад, якщо спробуємо виконати макетування сучасної мікросхеми з мільйонами елементів, то навряд чи такий макет взагалі може бути працездатним, якщо взяти до уваги ймовірність проблемності його складових. Крім того ситуація ускладнюється значно більшими часовими затримками при розповсюдженні сигналів між елементами макету через принципово іншу фізичну топологію, що призводить до порушення синхронізації.

За таких обставин є бажаним замінити (як мінімум для попереднього дослідження) фізичну модель на деякий математичний опис. ***Математичною моделлю називається сукупність математичних співвідношень, рівнянь, нерівностей тощо, які описують основні закономірності, притаманні досліджуваному процесу, об'єкту або системі*** [10]**.** Використання математичної моделі замість фізичної зазвичай стає переважним, а часто навіть єдиним можливим способом коректно змоделювати об’єкт. Саме тому математичному моделюванню та розвитку його технологій в останні десятиріччя приділяють особливу увагу. Математичні моделі можна поділити на

- структурні;
- функціональні.

Їх відмінність витікає з назви цих класів. Структурні моделі відтворюють будову об’єкта, тобто розкладають об’єкт на сукупність деяких базових елементів (які розглядаються як неподільні) та зв’язки між ними. Зрозуміло, що від вибору рівня елементів, які вважаються базовими, буде залежати точність моделювання. І хоча зазвичай і при структурному моделюванні задачею все одно є функціонування об’єкта дослідження, ця функціональність реалізується саме на основі зв’язків обраної системи елементів.

Функціональні моделі не використовують докладного представлення структури і відображають тільки деякі наперед відомі ознаки поведінки (функціонування) об'єкта. Досить часто такі ознаки отримують емпірично, іноді навіть за недостатньою базою спостереження або не зовсім у відповідних умовах. Це, звичайно, також впливає на точність та достовірність моделювання, але у багатьох випадках подібне наближення є єдиним можливим.

Ідеологія дослідження поведінки взагалі без прив’язки до внутрішньої структури часто називається моделлю «чорного ящика». За аналогією комбінована модель (частково структурна, але з

використанням функціонального моделювання непідкріпленого структурою) іноді називається моделлю «сірого ящика».

Особливістю функціональних моделей може бути відносно висока універсальність, тобто одна математична (функціональна!) модель може використовуватись для опису принципово різних явищ, часто навіть різної природи. Наприклад, гармонійний осцилятор відповідає механічним (маятник, математичний та пружинний маятник) або електричним (LC контур) коливанням. Але рівняння осцилятору може описувати, наприклад, і розвиток деякої біологічної популяції або коливальний процес в економіці.

Часто при використанні математичного моделювання кажуть про пряму та зворотну задачі.

Для ***прямої задачі*** структура моделі та параметри її складових вважаються відомими, а безпосередньою задачею дослідження є поведінка об'єкта при таких параметрах. Наприклад, необхідно визначити форму вихідного сигналу деякого радіотехнічного пристрою при відомій його структурі та формі вхідного сигналу, або визначити розподіл світлового пучка при проходженні через складну систему розсіювачів чи в неоднорідному середовищі.

Зворотною задачею є визначення структури або параметрів моделі для отримання заданого результату. Найпростішим є випадок, коли відома структура моделі, і моделювання повинне саме визначити тільки її параметри.

Крім використання моделювання для потреб випробовування властивостей розробленого пристрою, воно знаходить своє місце і як безпосередній елемент проектування. Наприклад, в архітектурі або у машинобудуванні значну частину проектування складають формоутворення та дослідження механічної міцності. Традиційні прийоми ведення інженерних розробок саме через потреби цих двох задачі призвели до надзвичайного значення в інженерії «інженерної графіки» та «супротиву матеріалів» (більш відомий російській термін «сопромат»).

Проблема формоутворення полягає в тому, що інженер традиційно виконував проект тривимірної форми у вигляді деякого набору плоских видів, а тривимірну реалізацію можна було отримати тільки в разі виконання як мінімум фізичної моделі. Відповідно, аналіз міцності міг бути виконаний вже на моделі або суттєво спрощеними методами, значною мірою на основі емпіричних оцінок. Такий метод має значні недоліки. Наприклад, низька точність таких оцінок не дозволяє оптимізувати форму за умовою мінімальної ваги при заданій міцності, не може відтворити поведінку об'єкта при всіх конфігураціях навантаження.

Саме математичні методи зв'язування форми у просторі з довільними плоскими проекціями та математичні моделі, які спираються на рівняння фізики суцільного середовища, дозволяють зробити таке проектування значно швидшим, більш надійним та дешевим.

При розробці радіотехнічних пристроїв зазвичай проблеми тривимірного формоутворення та міцності не виникають, але є необхідним виконати моделювання обробки сигналів (як аналогової, так і логічної), а крім цього розрахувати топологію з'єднань на платі (зазвичай з великою кількістю шарів). Обидві ці задачі можуть бути виконані математичним моделювання з використанням комп'ютера.

Загалом математичне моделювання передбачає:

- постановку задачі (створення для об'єкта моделювання опису – фізичного, технічного, хімічного, економічного тощо);
- формалізацію (формування системи рівнянь, які відповідають зробленому опису об'єкта);
- вибір та реалізацію способу розв'язання сформованої системи рівнянь;
- перевірку точності розв'язку та інтерпретацію отриманих результатів.

Розглянемо детальніше перераховані етапи. Для створення опису за основу береться докладна конфігурація об'єкта моделювання з врахуванням законів тієї наукової або технічної галузі, якій відповідає об'єкт дослідження. Зазвичай ці закони мають певне математичне відтворення, що дозволяє виконати формалізацію. Вже на цьому етапі можна відкинути чистину опису об'єкта дослідження як несуттєву для поставленої задачі моделювання.

Формалізовану задачу в принципі можна спробувати розв'язати аналітично, тобто на основі використання тотожних математичних перетворень, що призводять до формульного виразу, який дозволяє отримати результат прямою підстановкою вхідних даних задачі. На перший погляд саме такий, аналітичний, розв'язок є найбільш бажаним, як той, що не вносить додаткову похибку процесу розв'язання. Але за дуже рідким виключенням результат формалізації виходить дуже складним і для застосування аналітичних методів доводиться або з самого початку суттєво спрощувати модель, що вносить додаткову похибку, або значним чином звужувати границі застосування розв'язку.

Для більш загального розв'язання задачі треба шукати інший метод, і саме чисельне моделювання, яке зазвичай називають комп'ютерним, може суттєво покращити ситуацію.

2.4.2. Програмні засоби для моделювання

Для моделювання довільних прикладних задач може використовуватись безпосередня розробка деякої програми звичайними методами програмування. При цьому необхідно розробити досить складну математична частину загального проекту. До цього потрібно додати значний обсяг засобів попередньої підготовки даних до моделювання та їх виведення у зручній для науковців та інженерів формі. Останнє зазвичай робиться на основі використання комп'ютерної графіки. Звичайно, все це може також бути реалізовано засобами деякої загальної мови програмування. Але саме для виконання проектів з моделювання часто використовують дещо інший підхід. Оскільки у задачах моделювання багато спільних складових (наприклад, реалізації математичних методів, засобів виведення та візуалізації результатів), є сенс розробити окремий засіб (програмний пакет) для моделювання. Такий засіб може мати певну універсальність, або бути пристосованим для відносно вузького класу задач або об'єктів.

На сучасний момент є багато досить потужних розробок програмних продуктів, призначених саме для математичного моделювання наукових та інженерних задач. Активно розробляються та використовуються також засоби комп'ютерної алгебри. До систем комп'ютерної алгебри відносяться програми, які дозволяють виконувати найрізноманітніші математичні операції та перетворення алгебраїчних виразів, заданих у чисельній та символьній формах, тобто з використанням змінних, функцій, поліномів, матриць тощо. До найбільш популярних пакетів для моделювання можна віднести, наприклад, Maple, Mathematica, Mathcad, MATLAB.

Maple (розробка Waterloo Maple Inc., починаючи з 1980 року) в актуальній версії містить понад 5000 функцій для більшості розділів моделювання та візуалізації, підтримує свою власну мову програмування, дозволяє комбінувати алгоритми, результати обчислення, математичні формули, текст, графіку, анімацію, звук. Сучасна версія з врахуванням особливої ресурсоємності задач моделювання орієнтується на засоби прискорення обчислень, до яких можна віднести:

- автоматичне розпаралелювання;
- багатопотокове програмування;
- обчислення в грід мережах;
- підтримка CUDA.

Крім того, забезпечуються розвинені методи роботи з базами даних та засоби обміну даними з іншими пакетами, наприклад, Matlab.

Іншим подібним пакетом є **Mathematica** (рис. 2.9.), розроблена та розвивається компанією Wolfram Research, починаючи з 1988 р. Цей пакет крім засобів чисельного моделювання містить багато функцій для аналітичних перетворень:

- розв'язання систем поліноміальних і тригонометричних рівнянь і нерівностей, а також трансцендентних рівнянь, що зводяться до них;
- розв'язання рекурентних рівнянь;
- спрощення виразів;
- знаходження границь;
- інтегрування і диференціювання функцій;
- знаходження скінченних і нескінченних сум і добутків;
- розв'язання диференціальних рівнянь і рівнянь в частинних похідних;
- перетворення Фур'є і Лапласа, а також Z-перетворення;
- перетворення функції у ряд Тейлора, операції з рядами Тейлора.

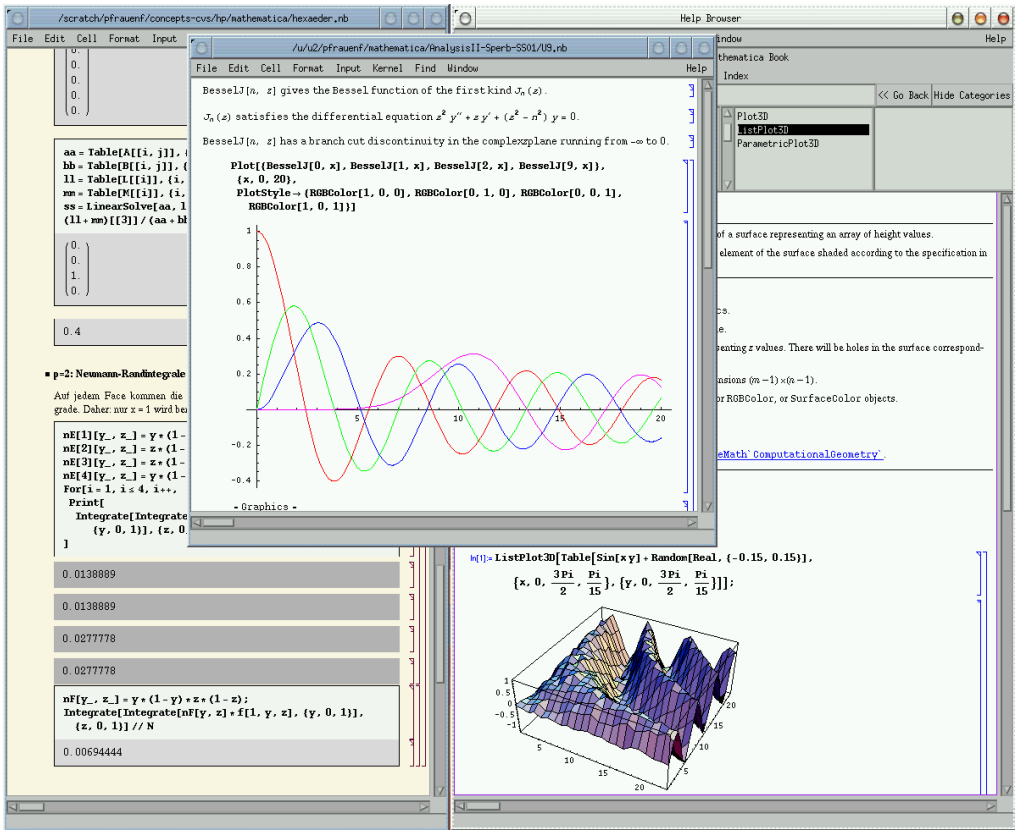


Рис. 2.9. Робоче вікно пакету Mathematica

Mathcad являє собою найпершу широковживану систему комп'ютерної алгебри (початкову версію реалізовано ще у 1986 р.) з

орієнтацією на підготовку інтерактивних документів з обчисленнями і візуальним супроводженням (рис. 2.10.). Mathcad отримав широке розповсюдження завдяки простому й інтуїтивно зрозумілому інтерфейсу користувача. Наприклад, введення формул і даних можна виконувати як за допомогою клавіатури, так і через спеціальні панелі інструментів. Робота користувача відбувається в межах робочого аркуша, з графічним відображенням рівнянь та виразів (на противагу текстовому запису в загальних мовах програмування), завдяки чому він є зручним для навчання, інженерних розрахунків. Сучасні версії підтримують технологій .NET та XML, що забезпечує інтегрування з іншими застосуваннями.

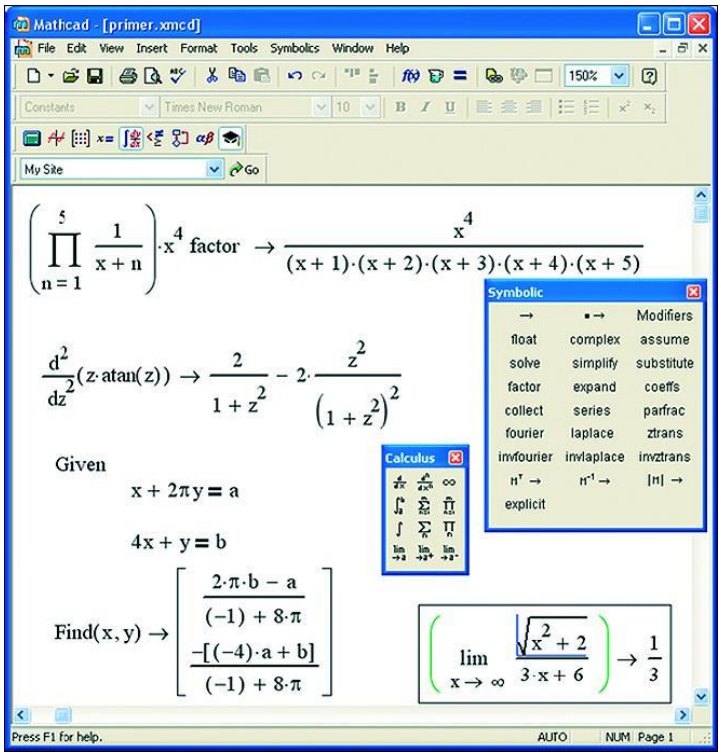


Рис. 2.10. Робоче вікно пакету Mathcad

Серед іншого можливості Mathcad охоплюють:

- розв'язання диференціальних рівнянь;
- символьні обчислення;
- операції з векторами і матрицями;
- символьне розв'язання систем рівнянь;
- згладжування кривих;
- знаходження коренів функцій і поліномів;
- статистичні функції і розподіли ймовірностей;
- пошук власних значень і власних векторів.

Іншим популярним пакетом, придатним для реалізації моделювання, є **MATLAB** (був створений компанією The MathWorks) (рис. 2.11). Він має власну вбудовану мову програмування, є зручним засобом для роботи з матрицями, математичною статистикою, інтерполяцією. Хоча цей продукт спеціалізується на чисельних методах, спеціальні інструментальні засоби роблять його сумісним із засобами Maple, що робить його придатним для роботи з алгеброю.

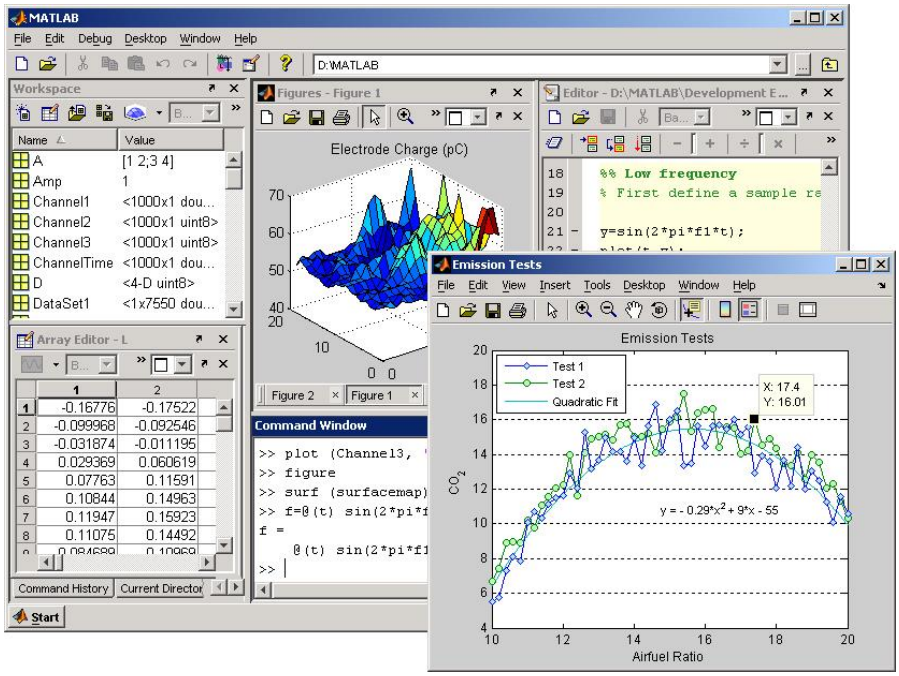


Рис. 2.11. Робоче вікно пакету MATLAB

2.4.3. Моделювання у радіотехніці

Як і інші галузі інженерії, радіотехніка активно використовує моделювання як складову процесу проектування радіотехнічної апаратури. Специфіка даної галузі добре узгоджується з розглянутими вище принципами математичного моделювання та використання для нього комп'ютерів. Але є і деякі особливості. Загалом, зазначене проектування апаратури може використовувати такі методи:

- декомпозиція (алгоритмічна та об'єктна);
- абстракція (спрощення відкиданням неважливих структур або характеристик);
- класифікація (встановлення ієрархії).

Алгоритмічна декомпозиція передбачає розбиття загального процесу на модулі, відповідає відтворенню проектування «згори донизу». Об'єктна декомпозиція передбачає відтворення взаємодії

об’єктів, що і визначає поведінку системи. Для розробки радіотехнічних пристроїв активно використовується певна номенклатура елементів та рішень на різних ієрархічних рівнях схемотехніки, що добре узгоджується з обома ситуаціями.

Основною задачею, що стоїть при проектуванні, є створення деякого рішення та визначення параметрів його елементів для реалізації наперед заданої обробки аналогових або цифрових сигналів. Відповідно, довільний спосіб розробки повинен бути завершений визначенням відповідності проекту до технічного завдання, що, як вже було зазначено, може бути виконано:

- макетуванням (довго і дорого);
- фізичним моделюванням (в радіотехніці особливого сенсу не має через неможливість фізичної реалізації подібності);
- математичним моделюванням з використанням багатоваріантного проектування та оптимізації.

Проектування радіоелектронних систем включає у себе такі методи:

- розрахунок (визначення параметрів підсистеми або модуля на основі його характеристик, які задовольняють загальному завданню, наприклад, розрахунок окремого каскаду підсилення);
- аналіз (визначення залежності характеристик радіоелектронної структури відповідно до варіювання її параметрів);
- оптимізація (визначення найкращих значень характеристик відповідним підбором параметрів);
- синтез (генерування варіанту підсистеми, в тому числі його структури – структурний синтез, наприклад, синтез фільтра).

Все це може передбачати використання програмних засобів обробки інформації. Найбільш розвиненими і універсальними при цьому у використанні математичного моделювання радіоелектронних систем на поточний момент є засоби схемотехнічного синтезу та аналізу їх характеристик. Як досить популярний приклад такого засобу можна розглянути програмний пакет Multisim.

Загалом Multisim є інтегрованою програмною системою моделювання радіоелектронних пристроїв на основі використання деякого набору елементів, які використовуються як складові загального схемотехнічного рішення. При цьому Multisim надає інструментальні можливості формування цього схемотехнічного рішення та подальше дослідження та вимірювання характеристик створеного (синтезованого) рішення на основі імітації робочого місця

дослідника, обладнаного необхідними традиційними засобами створення тестових сигналів та вимірювальних пристроїв. Все це дозволяє створювати як аналогові, так і цифрові системи різного рівня складності.

Відповідно, проектування з використанням даного програмного засобу включає в себе наступні етапи:

- створення схеми;
- вибір і підключення вимірювальних приладів;
- активація схеми.

Перший етап передбачає вибір необхідних елементів з деякої бази елементів та їх розміщення на робочому полі.

Для зручності база елементів зроблена ієрархічною, розділи цієї бази обираються відповідними іконками. Завдання місця розташування елемента відповідає загальним принципам візуального програмування та принципам організації GUI графічних редакторів, що дозоляє швидко пристосуватись до передбачених для цього інструментів. Бази елементів є трьох рівнів:

- головна база даних (Master Database), яка містить усі елементи, з цієї бази можна тільки зчитувати дані;
- користувацька база даних (User Database), яка відповідає поточному користувачу, призначена для зберігання компонентів, які є небажаним надавати в загальний доступ;
- корпоративна база даних (Corporate Database), призначена для тих компонентів, які повинні бути доступні іншим користувачам.

На цьому ж етапі виконуються з’єднання вказуванням мишею потрібних точок (дві точки на одне з’єднання), та, за потреби, точок галуження (рис. 2.12.).

Підключення до схеми віртуальних приладів проводиться аналогічним чином. Всі прилади розташовані на панелі інструментів, що забезпечує зручний їх вибір.

Прилади загалом традиційні. Наприклад, мультиметр забезпечує вимірювання змінного та постійного струму та напруги, а також опору. Можна вимірювати також загасання сигналу між двома вузлами схеми. Діапазон вимірювань підбирається автоматично. Внутрішній опір приладу є близьким до ідеального, але за потреби може бути змінений.

Генератор сигналів (рис. 2.13.) є джерелом напруги, може генерувати періодичні сигнали різної форми (синусоїдальної, пилкоподібної) та прямокутні імпульси. Параметри цих сигналів можуть бути змінені аналогічно до звичайних генераторів. Діапазон частот дуже широкий, від одиниць герц до радіочастотного.

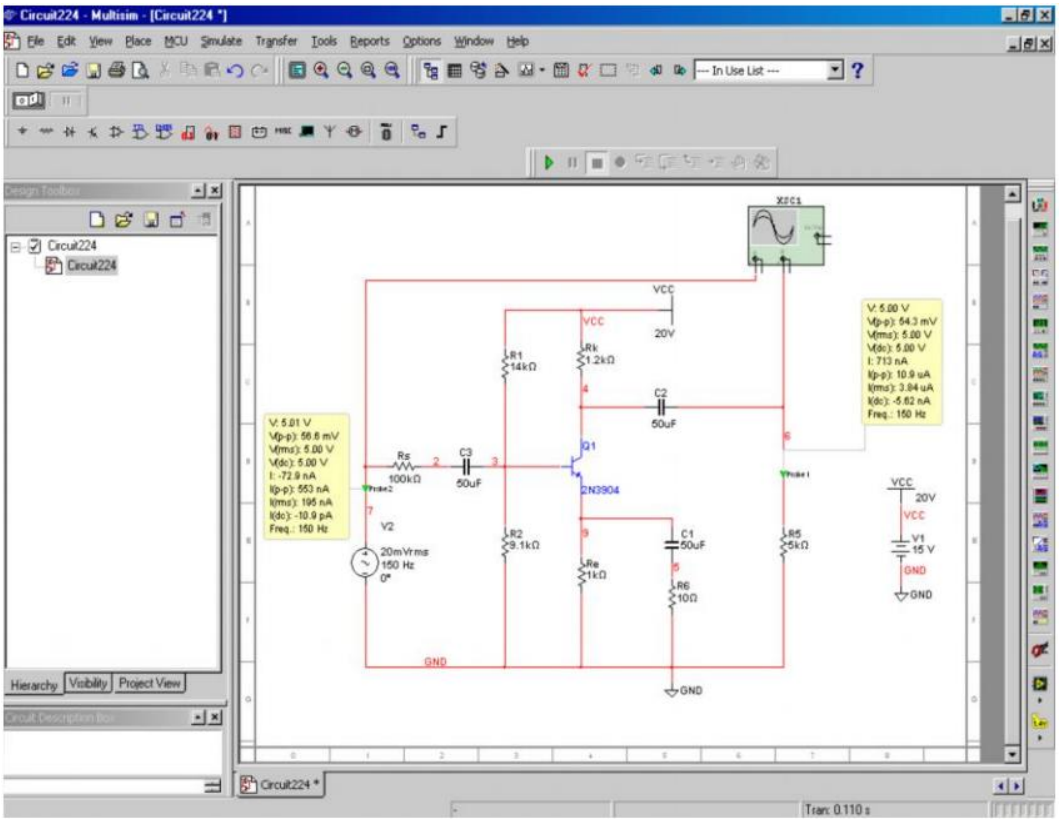


Рис. 2.12. Приклад моделювання схеми за допомогою Multisim

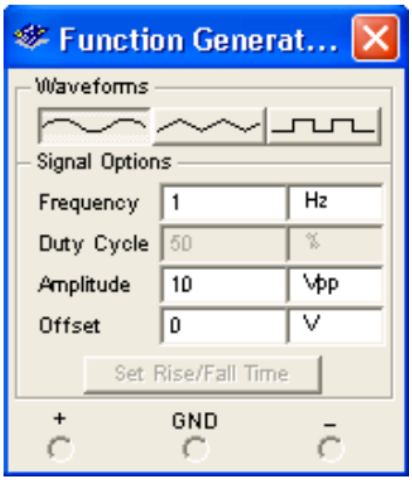


Рис. 2.13. Віртуальний генератор сигналів

Є й інші віртуальні прилади, такі як осцилографи (декілька варіантів, дво- та чотириканальні), аналізатор спектра, вимірювач АЧХ та ФЧХ (рис. 2.14.).

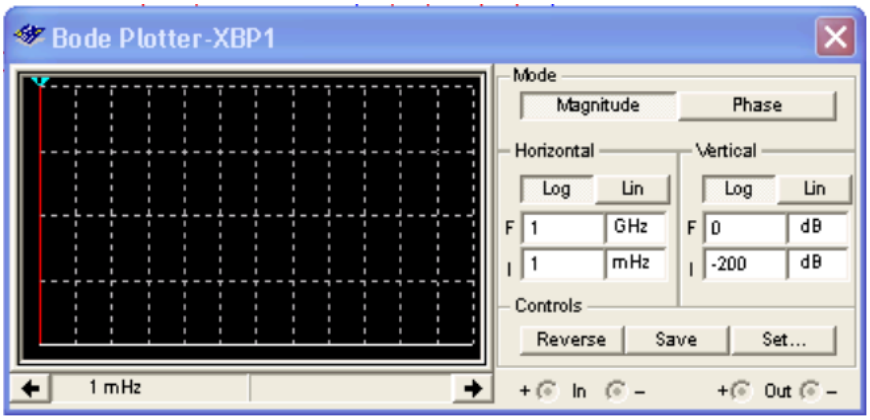


Рис. 2.14. Віртуальний вимірювач АЧХ та ФЧХ

Аналіз роботи розроблюваного пристрою є таких типів:

- DC – аналіз ланцюга на постійному струмі. Здійснюється для резистивних схем;
- AC – аналіз ланцюга на змінному струмі. Полягає в побудові частотних характеристик;
- Transient – аналіз перехідних процесів. Дозволяє визначити форму вихідного сигналу.

При моделюванні схем необхідно дотримуватись певних загальних правил:

- схема повинна обов'язково мати хоча б одне заземлення;
- будь-які два кінця провідника або контакт пристрою, які зустрічаються в точці, завжди вважаються з'єднаними. При з'єднанні трьох кінців (Т-з'єднання) необхідно використовувати елемент «з'єднання». Ті ж правила застосовуються при з'єднанні чотирьох і більше контактів;
- у схемах повинні бути присутніми джерела сигналу (струму або напруги), що забезпечують вхідний сигнал, і не менше однієї контрольної точки (за винятком аналізу схем постійного струму).

Топологія схем має такі обмеження:

- у схемі не повинні бути присутніми контури з котушок індуктивності та джерел напруги;
- джерела струму не повинні з'єднуватися послідовно;
- не повинно бути короткозамкнених котушок;
- джерело напруги повинне бути з'єднане з котушкою індуктивності і трансформатором через послідовно включений резистор. До конденсатора, підключеного до джерела струму, обов'язково повинен бути паралельно приєднаний резистор.

2.5. Питання для самоконтролю

1. Для яких потреб людство використовує зображення, як це враховується в розвитку комп'ютерних технологій?
2. Чому при недостатньому рівні освітлення погіршується сприйняття очима кольору?
3. Чи можна відтворити деякий умовний колір різними комбінаціями монохроматичних променів, або розв'язок є однозначним?
4. Яка мінімальна кількість видів (проекцій) забезпечує відтворення усіх координат елементів креслення?
5. Які переваги надає 2D проект у САПР в порівнянні з традиційним кресленням?
6. Які переваги надає 3D проект у САПР в порівнянні з 2D?
7. Чи використовуються інструменти САПР, призначені для 2D побудов, у 3D проекті?
8. Для чого при реалізації САПР, як програмного продукту, використовується растрова графіка?
9. Для яких потреб використовується комп'ютерна графіка при моделюванні?
10. Чи може існувати фізична модель транзистора?
11. В чому полягає моделювання деякого підприємства?
12. Що є спільного між пікселем та вокселем?
13. Чому полігон у 3D графіці завжди має три вершини?
14. Які розміри повинен мати z-буфер, якщо розмір кадру 600x800?
15. Чи виникає потреба використовувати метод трасування променів для непрозорих об'єктів?

3. Обробка зображень

3.1. Засоби введення зображень

3.1.1. Загальні принципи побудови датчика зображення

При використанні комп'ютера для обробки графічної інформації треба розв'язати три тісно пов'язаних між собою задачі:

- введення зображення до комп'ютера;
- безпосередньо математична обробка даних;
- виведення (візуалізація для сприйняття людиною).

Задача введення зображення традиційно використовує оптичні засоби, основні принципи формування яких сформувались задовго до появи цифрових технологій обробки сигналів. Призначенням таких засобів є формування на основі зміни ходу світлових променів оптичного зображення, яке максимально точно відтворює простір об'єктів. В просторі зображення зазвичай розміщувалась плоска фотоплівка (пізніше фотоплівка), яка дозволяла записати проекцію простору зображень на цю площину. З появою цифрових технологій залишилось замінити платівку або плівку на двовимірний сенсор, електричний сигнал з якого можна оцифрувати за допомогою спеціального пристрою – аналого-цифрового перетворювача.

Оскільки зображення є двокоординатним просторовим сигналом, його оцифровування передбачає просторову дискретизацію, тобто представлення його у вигляді двовимірної матриці неділимих ділянок – пікселів. Нагадаємо, що зазвичай піксель представляється сумою трьох субпікселів, які представляють так звані основні кольори пікселя. Фктично субпіксель є не частиною пікселя, а саме одною з компонент його векторного значення. Високоякісне зображення вимагає декількох мільйонів відліків.

Особливою проблемою в плані ресурсоємності введення даних є динамічне зображення (відеопотік), для якого до потреби введення мільйонів відліків зображення додається необхідність обробити 25–30 кадрів (зображень) за секунду.

Технології введення зображень охоплюють декілька задач:

- мінімізація кількості даних при збереженні інформації, яку може сприйняти око;
- створення піксельного сенсора (ефективного перетворювача в електричний сигнал світла, що потрапляє

протягом заданого проміжку часу на невелику ділянку поверхні);

- реалізація матриці піксельних сенсорів для ефективного отримання даних від усіх елементів матриці.

Розглянемо, яким чином розв'язуються зазначені задачі. Мінімізація кількості даних значною мірою вирішується на основі врахування особливостей сприйняття зором людини навколишнього світу (п. 2.1.2). Загалом оптичне зображення є сукупністю світлових променів, які відбивають, випромінюють, або заломлюють предмети навколишнього світу, а хід цих променів певним чином змінила оптична система. Зміна характеру проходження променів оптичною системою визначається фокусуванням, тобто збиранням у деяку точку в просторі зображення променів, що надходять від точкового об'єкту. Це робить, наприклад, об'єктив фотоапарата, який складається з декількох лінз.

Колір світла визначається довжиною хвилі світла, тобто в загальному випадку – усією сукупністю довжин хвиль, що проходять через піксель. Таким чином формально треба для кожного пікселя записувати увесь спектр, тобто виконати дискретизацію по дожині хвиль та виконати квантування амплітуди кожної складової хвилі. Ситуацію значно спрощує той факт, що око не є повноцінним аналізатором спектру, воно сприймає умовний колір на основі порівняння відгуку загального (складного) сигналу у точці від трьох датчиків – світлочутливих клітин, які називаються ковбочками. Нагадаємо, є три типи ковбочок, які мають максимум чутливості для трьох різних діапазонах довжин хвиль (умовно – червоночутливі, зеленосутливі та синьо чутливі, п. 2.1.2. Таким чином замість оцифровування для кожного пікселя усього спектру є достатнім обмежитись трьома сигналами, які відповідають зазначеним кольорам. Виділити ці сигнали з загального світлового потоку, що потрапляє на піксельний датчик можна смуговою фільтрацією, яка в оптиці реалізується навіть простіше, ніж у радіотехніці – підбором відповідного барвника, на основі якого створюється кольоровий світлофільтр.

У реальних системах, які зазвичай орієнтуються на подальшу комп'ютерну обробку, кількість даних при введенні зображення робиться ще меншою. Формально для введення трьох кольорових каналів треба було б розділити піксельний датчик на три субпіксельних, кожен з яких має свій світлофільтр. Відповідно кількість датчиків буде втричі більша за кількість елементів (пікселів), які визначають роздільну здатність матриці піксельних датчиків. Більш ефективним рішенням є використання по одному датчику на

піксель та використання сітки світлофільтрів по одному на піксель – так званого фільтру Баєра (рис. 3.1).

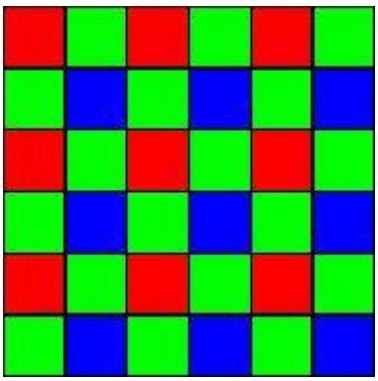


Рис. 3.1. Матриця Баєра для введення кольорового зображення до комп'ютера (R – red, червоний G – green, зелений, B – blue, синій)

Зрозуміло, що при введенні даних від такої матриці піксельних сенсорів виникає потреби визначити відсутні кольори для кожного з пікселів. Це робиться спеціальним алгоритмом інтерполяції, тобто додатковою цифровою обробкою даних. При цьому можуть виникати втрати роздільної здатності, яка залежить від конкретного змісту зображення. Частіше за все для реальних зображень, з врахуванням того як око сприймає оброблене зображення, погіршення якості сягає 10–20%, тобто є виграш у порівнянні з використанням повної тріади субпіксельних датчиків. Цьому виграшу сприяє і той факт, що для ока при сприйнятті дрібних деталей більш важливими є перепади за яскравістю, ніж за кольором. Крім того найбільшу чутливість око має у зеленій частині спектру, саме тому фільтр Баєра має вдвічі більше зелених ділянок, ніж інших.

3.1.2. Піксельний сенсор

Створення піксельного сенсора, як вже зрозуміло з розглянутого, передбачає світлофільтр та деякий електронний прилад, який може генерувати електрони пропорційно отриманій сенсором кількості світла, та прийнятний для інтегрального виконання (потрібна матриця, а не окремий сенсор). Зараз активно використовуються для класи подібних приладів, кожен з яких має свої переваги, але явного фаворита визначити важко:

- прилад з зарядовим зв'язком, ***ПЗЗ*** (англійською мовою – Charge-coupled device, ***CCD***);

- комплементарна структура метал-оксид-напівпровідник, ***КМОН*** (англійською –, complementary metal-oxide-semiconductor ***CMOS***).

Рядок CCD піксельних датчик створюється за рахунок того, що у кремнієвій підкладці p-типу формується канал n-типу, над якими формується лінійка електродів з полікристалічного кремнію вкритих ізолюючим шаром оксиду кремнію. Електричний потенціал, що подається на піксельний електрод, створює потенціальну яму в збідненій зоні під зазначеним каналом. Фотони, що проникають в кремній, генерують пару електрон-дірка, електрони накопичується у потенціальній ямі. Більша яскравість світла або довша експозиція (час освітлення датчика), визначає більшу кількість фотонів, а відповідно і більший заряд, що накопичився у ямі. Для зчитування з лінійки датчиків використовується спеціальний набір імпульсів, які подаються на сукупність електродів таким чином, що з кожним тактом заряди з ями стікають у сусідню в один бік, а наприкінці лінійки отримується дискретний часовий електричний сигнал, що відповідає просторовому сигналу розподілу заряду у рядку (регістр зсуву), рис. 3.2.

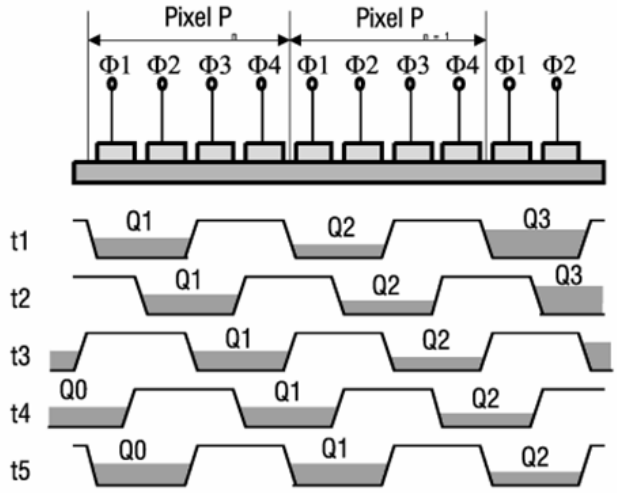


Рис. 3.2. сканування пікселів CCD лінійки

Для створення двовимірної матриці формується набір лінійок. Зсув на крок з лінійки у лінійку по одній з координат (стовпчиках зображення) формує для зчитування наступний рядок, який можна також просканувати відповідною сукупністю тактів зсуву. Зрозуміло, що поки зсувається рядок, зсув стовпчиків не виконується. Для зручності кожен з рядків фотодатчиків може мати дублюючий. При цьому одним кроком весь кадр переписується у дублюючі комірки, я яких виконується подальше порядкове зчитування. Така структура

матриці дозволяє реалізувати електронний затвор, тобто обмеження часу реєстрації пікселів кадру забезпечується командою, а не відповідним механічним пристроєм, що перекриває потік світла на матрицю.

CMOS-датчик (рис. 3.3) для перетворення енергії фотонів в електричний сигнал використовує фотодіод (1) – напівпровідникову структура на основі p-n переходу. Для накопичення заряду використовується інший елемент – конденсатор (3). Для зведення комірки до початкового стану є спеціальний ключ (7), час експозиції задається ключем (2). Для доступу до отриманого заряду піксельного датчика використовується ключ (6). Відповідно електрод вибору рядка (5) дозволяє забезпечити паралельне зчитування цілого рядка. У комірці використовується попередній підсилювач (4), що дозволяє покращити співвідношення сигнал/шум і при певних додаткових умовах забезпечити виграш за чутливістю в порівнянні з CCD датчиками.

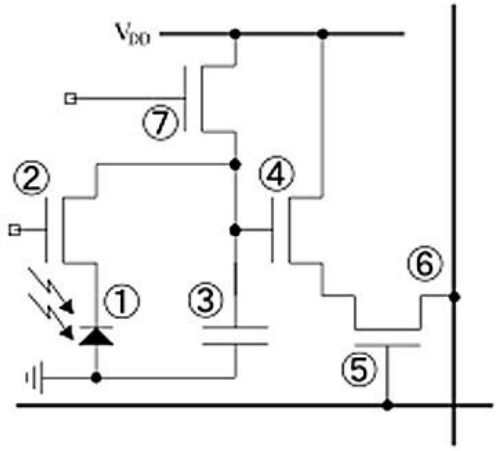


Рис. 3.3. Структура CMOS датчика

Описана структура CMOS-датчика має деякі недоліки в порівнянні із своїм конкурентом, а саме складність формування матриці однакових датчиків. Крім того площа фотодатчика займає значно менший відсоток повної площі пікселя за рахунок потреби формувати на поверхні кристалу комірку з більшою кількістю елементів. Тобто є програш також за чутливістю, оскільки значна частина фотонів, що потрапляє на ділянку пікселя, не потрапляє на призначений для них датчик. Розвиток технологій виготовлення мікросхем та цифрова обробка, яка використовує калібрування, дозволили нівелювати перший з недоліків. А технологія, яка

дозволила формувати над поверхнею напівпровідника матрицю мікролінз, кожна з яких збирає у потрібну ділянку фотодатчика майже усе світло, що потрапляє на квадратик пікселя, забезпечила кращі характеристики за чутливістю. Такий виграш призвів до майже повного витіснення CCD-технології при виготовленні матриць фотоапаратів та відеокамер.

Є ще одна цікава технологія, яка дозволяє отримати кольорові сигнали без фільтрації світла поглинанням зайвого – Foveon X3. За цією технологією світлодіоди кольорових каналів пікселя формується один над одним (верхній – синій, найглибший – червоний). Завдяки цьому кванти з найбільшою енергією («сині») поглинаються з утворенням електричних носіїв першими, глибше з тих, що не поглинулись, відпрацьовують «зелені», ще глибше – все що залишилось інтерпретується як «червоний». Теоретично така структура повинна мати суттєві переваги – немає втрат світла за рахунок поглинання у світлофільтрах, немає потреби в інтерполяції і роздільна здатність точно визначається кроком пікселів на матриці. Проте реальність виявилась складнішою і на поточний момент такі матриці в порівняння з баєрівськими замість приблизно триразового виграшу у чутливості мають програш у декілька разів. Складність технології не дозволила наздогнати і досягнуту конкурентами роздільну здатність. До того ж – не вдалось повністю побороти проблеми з якісним виділенням червоної складової (найглибший шар!). Через це такі матриці не отримали широкого розповсюдження, хоча в майбутньому можлива зміна ситуації.

Описані датчики використовується в багатьох цифрових пристроях – фотоапаратах, відеокамерах, смартфонах, веб-камерах, сканерах (в останньому випадку застосовують одновимірну піксельну лінійку) тощо. За основними фотооптичними характеристиками в останні роки ці матриці наздогнали старі аналогові технології, і разом з цим значно спростили і прискорили процес отримання остаточного зображення, придатного для перегляду людино. Цифрове зображення є готовим через долі секунди замість мінімум декількох хвилин, які забезпечувала «миттєва фотографія» фірми Polaroid, або години-двох в лабораторії при звичайній хімічній обробці. Це призвело до справжнього перевороту у використанні зображень. На принципово новий рівень вийшла професійна обробка фотографій, як художнього так і науково-технічного призначення. У телебаченні та кінематографі використовуються значно складніші технології монтажу. Використання фотографій, в тому числі мільйонів панорам в ГІС (графічних інформаційних системах) роблять роботу з картами значно зручнішою. Тисячі особистих фотографій і сотні відеороликів

зберігаються у приватному зібранні користувача, далекого від рівня фото- та відео-професіонала.

3.2. Технології роботи з зображеннями

3.2.1. Види обробки зображень

Ще до появи комп'ютерів при роботі з зображеннями виявилась низка проблем, вирішити які досить складно. Введення зображень раніше розробленими технологіями (як фотохімічними, так і електронними) забезпечувало якість, недостатню для комфортного спостереження людино. Аналогові телекомунікаційні технології призводили до помітного рівня шуму та появи артефактів (деталей на зображенні, яких не повинно бути). Певна робота із зображеннями, наприклад побудова топографічних карт на основі фотографій, вимагала надзвичайно великої кількості рутинних ручних операцій.

Розробка методів покращення характеристик зображень та виділення окремих їх характеристик визначили нову галузь – обробку зображень, яка значною мірою спиралась на методи, раніше розроблені для часових сигналів. Проте реалізація схожих методів для зображень вимагала врахування специфіки такого виду сигналів, в тому числі значне зростання складності апаратної реалізації пристроїв та методів.

Саме тому поява достатньо швидкої обчислювальної техніки одразу була використана для роботи з графічною інформацією. Більш того, пізніше розвиток архітектури засобів обробки інформації значною мірою став залежати від потреб саме роботи з зображеннями. Навіть з'явився спеціальний термін – «графічна станція».

Через схожість методів обробки інформації для часових і просторових сигналів надалі будемо ці дві задачі розглядати у взаємозв'язку.

Під цифровою (комп'ютерною) обробкою сигналу (зображення) можна розуміти будь який розрахунок вихідного сигналу (зображення), характеристики кожного значення якого є деякою функцією відповідних характеристик значень вхідного сигналу (зображення) а також одержання деяких числових величин, які є функцією значень вхідного сигналу (зображення).

В даному визначенні зображення розглядається як сукупність значень деякої функції від координат. При використанні комп'ютерних розрахунків така функція повинна бути дискретною, а незлічена кількість значень замінюється обмеженою за розмірами

матрицею відліків. У даному визначенні не випадково використана термінологія неперервних функцій. Це пов'язано з тим, що математичні формули, які описують методи обробки виводяться для неперервних функцій, а випадок дискретної графіки виводиться дискретизацією до таких «точкових» формул. Зауважимо, що положення пікселів вихідного зображення може не співпадати з сіткою відліків вхідного. Тобто є потреба крім формул для неперервних функцій використати інтерполяцію для дискретної сітки.

Найпростішою обробкою зображення є випадок, коли для розрахунку кожної точки вихідного зображення необхідна тільки одна точка вхідного. Така обробка називається ***точковою***.

Випадок, коли для обрахунку точки вихідного зображення потрібно врахувати деякий окіл точки вхідного називається ***локальною*** обробкою. Зрозуміло, що така обробка суттєво повільніша за точкову, особливо для великого околу, але і призначення таких методів в порівнянні з точковими є зовсім іншим.

Найресурсоємнішою обробкою є така, коли для кожної точки вихідного зображення доводиться обраховувати усе вхідне. Назвемо таку обробку ***нелокальною***.

До точкової обробка відноситься корекція яскравості та контрастності. У цьому випадку навіть зберігається сітка дискретного зображення, а перераховується значення яскравості або значення субкольорів за наперед заданою формулою. Трохи складнішою є корекція кольору, оскільки зміна окремого субкольору може призвести до колірного розбалансу Відповідно попередньо треба розрахувати правильні функції загального перерахунку. При такій обробці зв'язування сіток вхідного та вихідного зображень також зберігає точну прив'язку до дискретного пікселя, тобто все одно маємо точкове перетворення.

Точковим є поворот або деформація зображення. При такій обробці для відліків дискретного випадку потрібно використовувати інтерполяцію. Зазначимо, що для алгоритмічної реалізації перетворення при потребі інтерполяції є зручним, коли геометричне перетворення площини зображення має обернене (до таких перетворень відносяться досить популярні у використанні афінні). У цьому випадку заповнення даних вихідної сітки забезпечується двома вкладеними циклами по відповідним координатам вихідного кадру. Геометричне положення відліку (відліків) вхідного зображення розраховуються оберненим перетворенням, а значення вихідного пікселя розраховується інтерполяцією даних з врахуванням взаємного положення по відношенню до сусідніх відліків на вхідному. Таким

чином стає зрозумілішим, чому використовуються формули саме для неперервних функцій.

3.2.2. Фільтрація зображень

У загальному випадку недоліки отриманого сигналу (часового або просторового) призводять до потреби використання обробки, яка змінює певні характеристики сигналу, що отримало назву «фільтрація». Частіше за все використовується лінійна фільтрація, яка математично описується застосуванням деякого лінійного оператора. Обмеження випадком лінійності значно спрощує математичний опис фільтра. У випадку лінійної системи вхідний сигнал може бути розкладений як сума інших сигналів, кожен з яких надалі обробляється незалежно від інших, а на виході результати підсумовуються. Одразу зауважимо, що в реальних аналогових пристроях (наявність у системі електронних елементів, використання обробки зображення хімічними процесами) досягнення лінійності часто є досить проблематичним. Тобто лінійність в більшості випадків роботи з аналоговими технологіями розглядається тільки як деяке наближення. Використання розрахунків для реалізації фільтра дозволяє застосувати «правильні» формули, що забезпечує коректність лінійності. При цьому загальна похибка є контрольованою, оскільки фактично формується похибкою оцифровування (точність виконання розрахункових дій завжди може бути зроблена меншою за неї).

Розглянемо використання лінійної фільтрації для зображень на основі аналогії з радіотехнікою. Часові сигнали, які є традиційними для електроніки та телекомунікації, зазвичай зручно розкласти по частотам. При цьому періодичний сигнал може бути представлений сумою синусоїд кратних частот (гармонік, рис. 3.4), неперіодичний – формальним розкладенням по усім можливим частотам. На рис. 3.5. як приклад наведено розклад меандру. Таке представлення відповідно до цих двох випадків є розкладенням в ряд або інтеграл Фур'є. Оскільки в нашому випадку цікавішим є дискретний випадок, наведемо пряме та обернене перетворення у вигляді ряду:

$$F(\omega_n) = \frac{1}{N}\sum_{k=0}^{N-1} f(t_k) e^{-j\frac{2\pi}{N}nk}$$

$$f(t_k) = \sum_{n=0}^{N-1} F(\varpi_n) e^{j\frac{2\pi}{N}nk}$$

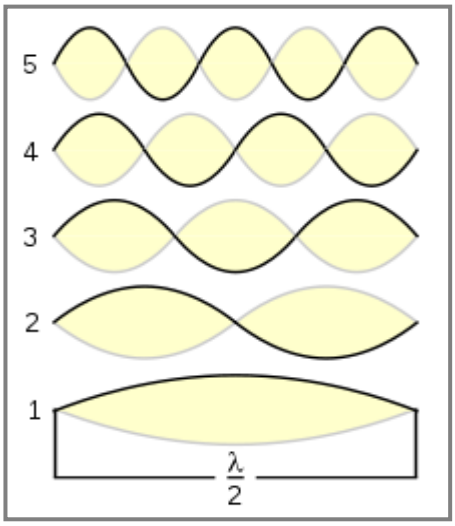


Рис. 3.4. Гармоніки, по яких розкладається періодичний сигнал

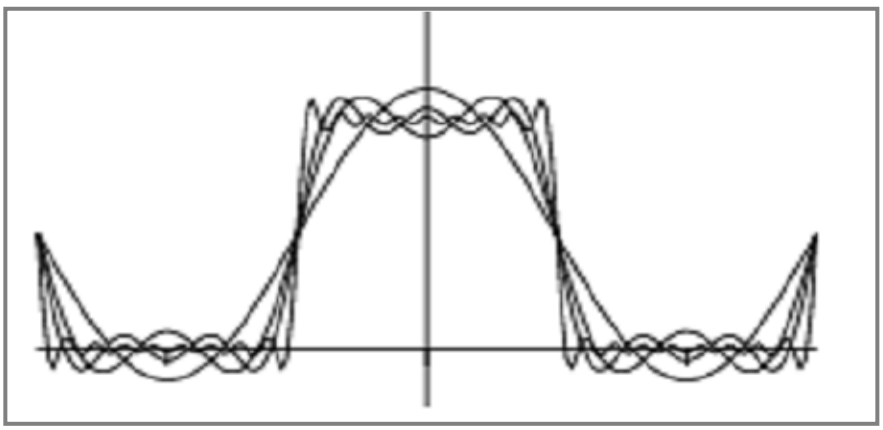


Рис. 3.5. Представлення меандру сумою гармонік

У радіотехніці фільтрація зазвичай використовується для виділення корисного сигналу на фоні стохастичного шуму. Можливість отримання покращених характеристик пов’язана з тим, що зазвичай корисні сигнали мають певні обмеження, наприклад за частотами, присутніми у них. В цьому випадку можна спробувати придушити ті частотні складові, які відповідають в основному шуму. Крім широкосмугового (у широкому діапазоні частот) може

накладатись вузько смугова завада. Логічно, що ця завада може бути послаблена придушенням цього вузького діапазону. Звичайно така обробка деяким чином змінює і форму корисного сигналу, що вимагає шукати деякий компроміс за характеристиками фільтра.

Співвідношення сигналу до шуму, близьке до одиниці, може вимагати як мінімум простого виявлення імпульсів на фоні шумів. Певні методи фільтрації дозволяють зробити і це. При такій фільтрації значно змінюється форма імпульсу, але головне – стає можливим виявити його положення.

Зображення також може бути розкладене по гармоніках. Але на відміну від часового сигналу, який замінюється на суму часових, в даному випадку функціями, по яким виконується розкладення зображення також є зображеннями (рис. 3.6).

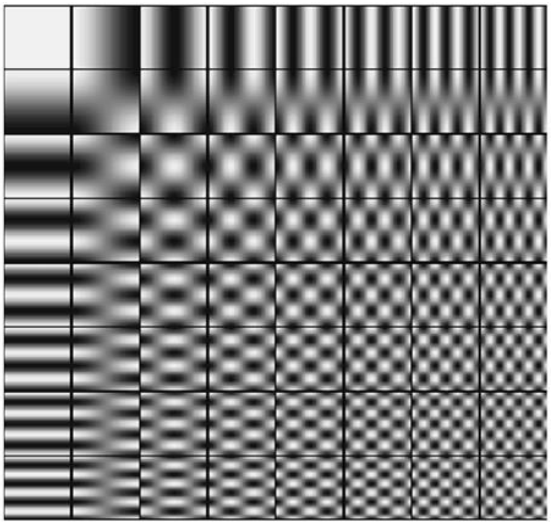


Рис. 3.6. Просторові гармоніки, по яких розкладається зображення

Такі функції отримали назву просторових гармонік. І, якщо гармоніка характеризується одним числом – частотою, просторова гармоніка визначається парою чисел – просторовою частотою.

$$F(u_n v_m) = \frac{1}{NM} \sum_{k=0}^{N-1} \sum_{l=0}^{M-1} f(x_k y_l) e^{-j(\frac{2\pi}{N} nk + \frac{2\pi}{M} ml)}$$

$$f(x_k y_l) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f(u_n v_m) e^{j(\frac{2\pi}{N} nk + \frac{2\pi}{M} ml)}$$

І в цьому випадку розкладення по просторовим частотам може використовуватись для фільтрації, як для часових сигналів. Потреби фільтрації зображень охоплюють:

- придушення шумів (фільтр низьких частот);
- виділення країв (фільтр високих частот);
- підкреслення країв (комбінація оригіналу з дією фільтра високих частот).

Загальна схема фільтрації для зображень співпадає з аналогічною для часових сигналів (рис. 3.7). Фільтрація безпосередньо задається деякою функцією в області просторових частот, що задає, які з гармонік будуть сильніше придушуватись. Ця функція у загальному випадку має бути визначеною у всій частотній області. Відповідно потреба розрахунку прямого і зворотного перетворення Фур'є та множення на фільтр є доволі ресурсоємним, особливо з врахуванням двовимірності та кількості відліків за частотами, яка визначається кількістю відліків зображення. Це є причиною того, що для зображень частіше використовується не частотна фільтрація, а згорточка.

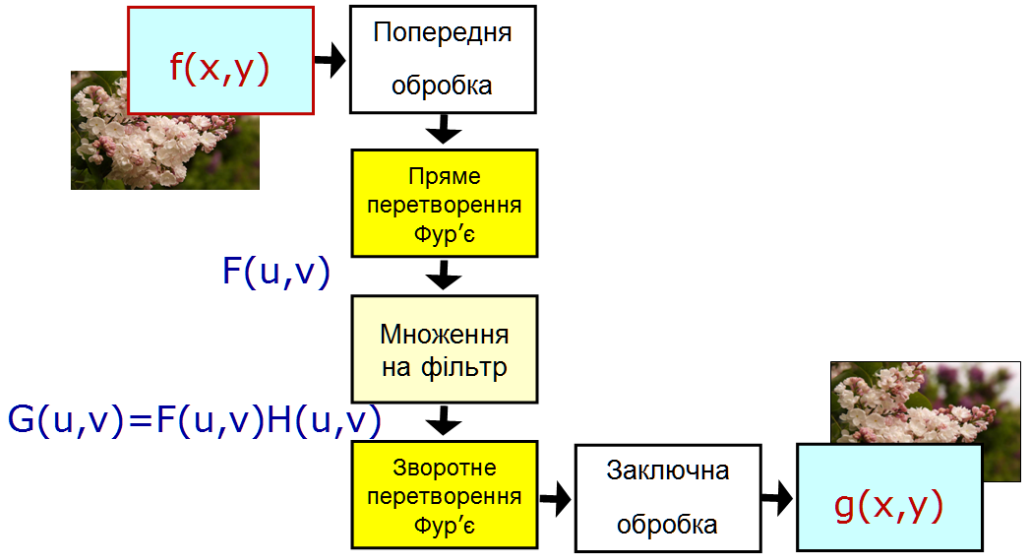


Рис. 3.7. Загальна схема фільтрації просторових частот при обробці зображень

З точки зору математики без використання додаткових способів зменшення ресурсоємності згорточка та частотна фільтрація є еквівалентними. Про це каже ***теорема про згортку – фур'є-образ згортки функцій дорівнює добутку їх фур'є-образів.*** Тобто для переходу від частотної фільтрації до еквівалентної згорточної є

достатнім розрахувати обернене перетворення Фур'є для функції частотного фільтра, при цьому формальна ресурсоємність фільтрації не зменшиться. Але для більшості задач фільтрації зображень, апертуру ядра згортки можна суттєво обмежити та замінити значення поза апертурою нулями. Зрозуміло, що така модифікація фільтра дещо погіршить результат його застосування, але саме такий підхід є розумним компромісом між якістю обробки зображення та швидкодією. Ілюстрація фільтрації на прикладі деякого модельного зображення наведений на рис. 3.8 – 3.12.

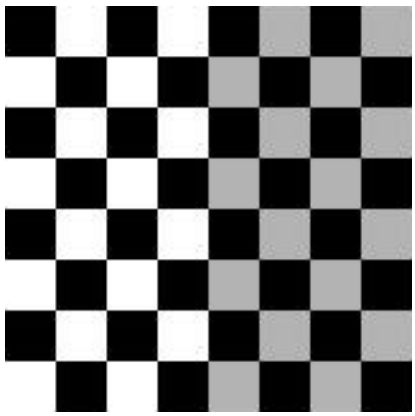
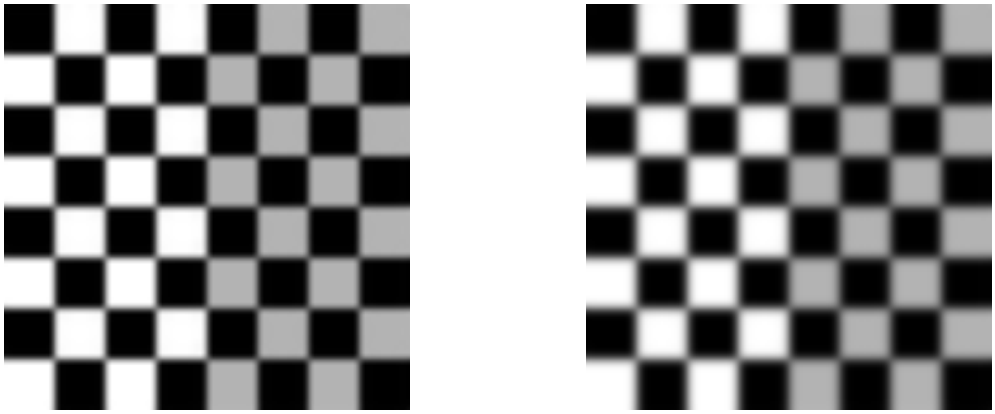
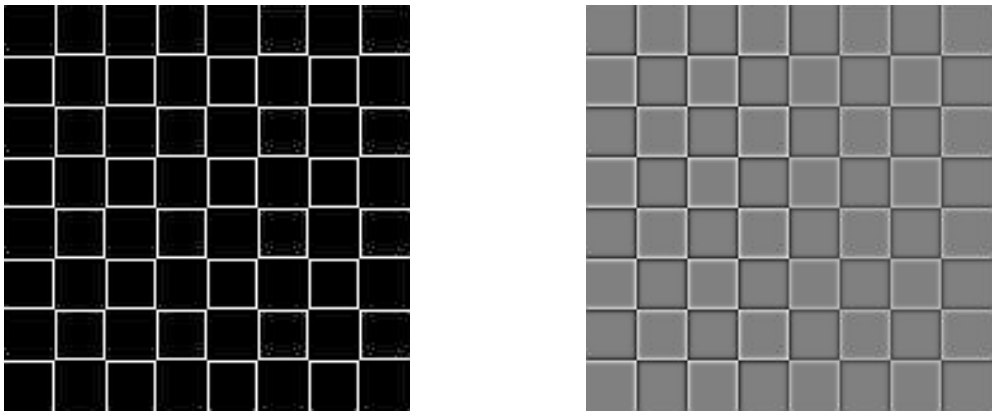


Рис. 3.8. Оригінал модельного зображення для демонстрації фільтрації



а *б*

Рис. 3.9. Результат згорточної (а) та частотної (б) фільтрації низьких частот зображення рис. 3.8.



а *б*

Рис. 3.10. Результат згорточної (а) та частотної (б) фільтрації високих частот зображення рис. 3.8.

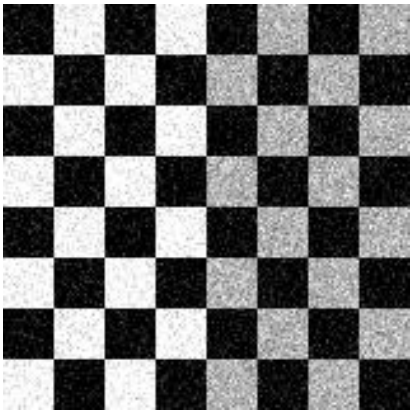
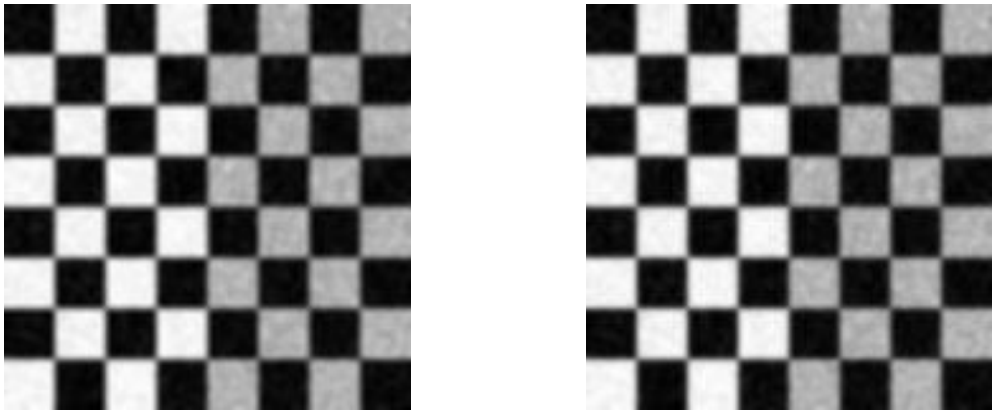


Рис. 3.11. Оригінал модельного зображення з доданими шумами для демонстрації фільтрації.



а *б*

Рис. 3.12. Результат згорточної (а) та частотної (б) фільтрації низьких частот зображення з шумами (рис. 3.11).

3.2.3. Сегментація зображення

При роботі з інформацією, яку представляє деяке зображення, часто є потреба її аналізу на основі визначення окремих об'єктів та їх характеристик. Серед найбільш популярних задач цього класу можна виділити:

- розпізнавання тексту;
- введення даних в географічні інформаційні системи (побудова топографічних карт);
- медичні діагностичні та експертні системи;
- системи спостереження;
- комп'ютерний зір, в тому числі для потреб робототехніки.

Відповідно першим етапом аналізу зображення як з сукупністю певних об'єктів є виділення його ділянок, які відповідають цим об'єктам. Таке виділення називають сегментацією. Тобто ***сегментація зображення – процес розділення цифрового зображення на декілька сегментів (множин пікселів). Результатом є сукупність сегментів, які разом покривають все зображення. Всі пікселі в межах сегмента схожі за деякою властивістю. Сусідні сегменти істотно відрізняються за цими властивостями.***

Задача сегментації є складною і неоднозначною. Її ефективність та точність зазвичай пов'язана з конкретним видом зображення та задачами, що розв'язуються. Відповідно було розроблено велику кількість різних методів, які можна поділити на дві групи:

- виділення контурів та визначення їх внутрішньої області;
- безпосереднє розбиття на області.

Виділення контурів може бути виконане фільтрацією (п. 3.2.2). При цьому краї зазвичай виходять далекими від ідеальних ліній, можуть формуватись паразитні розгалуження, розриви, дрібні острівці. Відповідно до фільтрації треба додати певні алгоритми подальшого покращення границь. Границя не є самим сегментом, що вимагає розробки та реалізації відстеження належності точки внутрішній частині по відношенню до замкненої границі. Певна складність такого підходу призводить до того, що більш розповсюдженими є методи сегментації другої групи. Серед них найбільш популярними є:

- порогова бінаризація;
- кластеризація;
- нарощування областей;
- розділення та злиття областей;
- морфологічний водоподіл.

Порогова бінаризація є найпростішою. В цьому випадку ознакою належності до того, чи іншого сегменту є перевищення порогу за певною скалярною ознакою, яка може бути розрахована з пікселя або певного його околу. Часто для визначення цього порогу використовують наявність локального мінімуму обраної ознаки (наприклад, яскравості) на гістограмі зображення.

Кластеризація забезпечує результат, що практично не залежить від початкового вибору настроювання на відміну від порогової бінаризації. Один з варіантів кластеризації має такий ітераційний алгоритм:

- початковий стан визначається довільним завданням центрів кластерів;
- кожен з пікселів відноситься до кластеру, центр якого є найближчим за наперед заданою метрикою в просторі ознак;
- центри кластерів перераховуються усередненням усіх пікселів, які є в кластері в результаті даної ітерації.

Метрика ознак може враховувати досить широку сукупність характеристик (яскравості, кольору, текстури, геометричного положення, тощо) При цьому підсумовування ознак може виконуватись із завданням певних вагових коефіцієнтів. Вибір метрики та її налаштування робить цей метод досить гнучким по відношенню до задач і типів зображень.

Метод нарощування областей спирається на об'єднання (злиття) в кожній з ітерацій сусідніх ділянок, які відрізняються найменше з усіх сусідів. Цей метод є доволі проблемним через те, що фінальний результат може залежати від початкового розділення на найменші ділянки – «насіння».

Метод розділення та злиття областей є своєрідним розширенням попереднього. Він додатково включає в себе поділ сегментів на кожній з ітерацій за певними ознаками з подальшим злиттям. Для розбиття визначається певний поріг неоднорідності ознаки, тобто розділяється тільки неоднорідний сегмент. За рахунок цього частина сегменту в даній ітерації може бути зарахована в інший сегмент, що і забезпечує покращення сегментації від ітерації до ітерації.

Метод водоподілу спирається на ідеї математичної морфології. У географії вододіл – хребет, що ділить області збору води різних річкових систем. Відповідно проводиться певна аналогія зображення з ландшафтом (яскравість чи складніша комбінована ознака – висота), а лінії вододілу – границі, що розділяють ділянки зображень.

Основною проблемою цього алгоритму є надмірність сегментації, особливо для зображень із помітним рівнем шумів.

3.2.4. Розпізнавання образів

При використанні комп'ютерів часто «виникає бажання» частину операцій, які традиційно виконує мозок людини, передати комп'ютеру. Подібні задачі досить часто вдається розв'язати, іноді навіть методами, які імітують роботу мозку. В цьому випадку часто кажуть про «штучний інтелект», хоча реально з інтелектом навіть у найскладніших подібних системах не так багато спільного. Але для практичного застосування головне результат, а не те як само він досягнутий. На практиці ж класифікація сигналів за допомогою комп'ютера є надзвичайно важливою. Ця задача (її називають розпізнаванням образів) стосується розпізнавання інформативних складових мови, наявності рухів, класифікації об'єктів на зображенні, побудови змістового зв'язку між об'єктами тощо. Серед популярних практичних задач, для яких розпізнавання образів є надзвичайно актуальним, можна виділити:

- введення друкованого тексту;
- введення рукописного тексту (значно складніша за попередню задача);
- аналіз зображень аеро- та космічної фотографії (наприклад, для картографії);
- комп'ютерний зір (автотранспорт, робототехніка, військове використання, тощо);
- наукові застосування (рутинні дії з введення та попередньої обробки результатів досліджень):
- медичні застосування (автоматизація діагностики);
- розпізнавання облич (системи спостереження і безпеки, деякі задачі автоматизації фото- та відеозйомки);
- автентифікація.

Загалом розпізнавання образів (розглянемо його на прикладі графічних об'єктів) розпочинається з результатів сегментації (п. 3.2.3). Далі з загального складного зображення вибирається по черзі один з об'єктів (наприклад одна літера з тексту) і порівнюється з усіма відомими (класифікованими раніше) об'єктами. Таким чином даний досліджуваний об'єкт класифікується, тобто відноситься до одного з наперед заданих класів, наприклад, досліджувана (ще невідома) літера відноситься до можливих реалізацій літери «В».

Одним з варіантів розв'язання цієї задачі є розрахунок кореляційної функції з усіма зразками і визначення, з яким із зразків максимум цієї функції є найбільшим. Проте такий метод може використовуватись тільки коли характеристики зразків є слабо варійованими. Якщо повернутись до прикладу з літерами, існує велика кількість зовсім різних зображень, які ми відносимо до однієї літери – поліморфізм. Саме тому просте попіксельне порівняння самих зображень (досліджуваного і одного з класифікованих) є неефективним. До відмінностей може призводити розмір, орієнтація (поворот), деформація (навіть невелика), різний шрифт. І навіть використання певних математичних методів, що уніфікують орієнтацію та масштаб, не забезпечує високу надійність розпізнавання. Саме тому зазвичай від «простору об'єктів» переходять до «простору ознак», до яких, наприклад можуть бути віднесення особливості форми границі, топологічні властивості, текстури, співвідношення спектральних складових тощо.

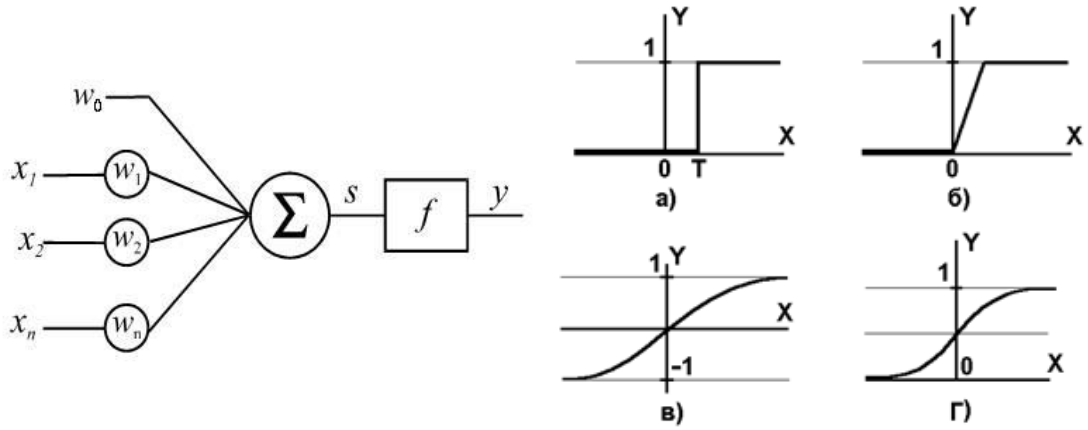
Зазначимо, що класифікація об'єкту відбувається за певним набором ознак, досить великим за кількістю складових. Можна побудувати деякий багатовимірний гіперпростір, у якому значенню кожної з ознак буде відповідати одна вісь. Відповідно кожен з об'єктів відповідає точці у цьому гіперпросторі, а між різними класами (множинами об'єктів, визнаних спільними) можна визначити границі. Тобто досліджуваний об'єкт при класифікації потрапляє в одну чи іншу частину в межах границі ознак певного класу.

Є багато алгоритмів розпізнавання, але останній час доволі популярним і досить ефективним стало використання штучних нейронних мереж – примітивної імітації біологічного мозку, точніше деякої математичної моделі, яка від біологічного прототипу використовує певні ідеї:

- одиночний обробник (вузол) виробляє вихідне збудження на основі обробки деякої кількості вхідних збуджень (рис. 3.13);
- одиночний обробник має настроювання чутливості для кожного з вхідних збуджень;
- одиночні обробники об'єднуються певною структурою зв'язків у мережу (одна з поширених топологій такої мережі - персептрон, рис. 3.14).

На цьому копіювання прототипу закінчується. Математична модель штучного нейрона, сильно відрізняється від біологічного (справжнього) нейрона, робота якого взагалі вивчена ще недостатньо. Штучний нейрон є об'єднанням зваженого суматора, рис. 3.13.І (тобто до підсумовування кожний доданок множиться на ваговий коефіцієнт)

та функції активації (рис. 3.13.II), яка визначає умову переведення виходу в збуджений стан.



I *II*

Рис. 3.13. Схема штучного нейрона (вузла штучної нейтронної мережі) – I та приклади вихідної функції (функції активації) - II.

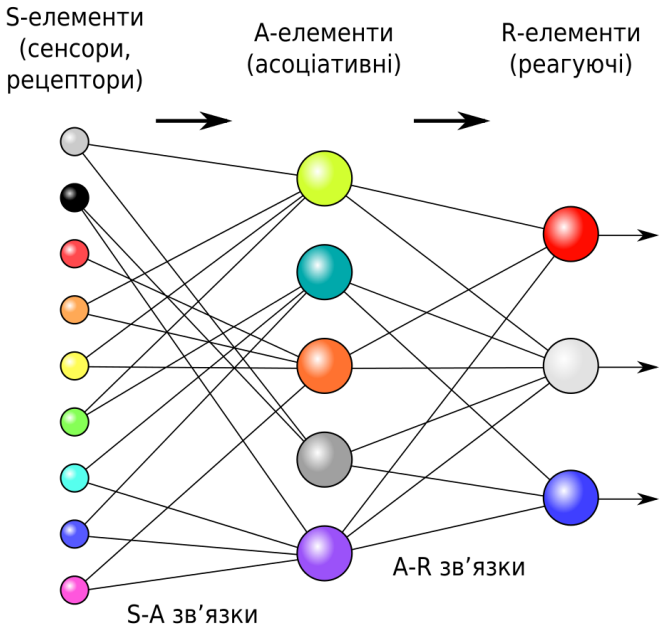


Рис. 3.14. Приклад топології штучної нейтронної мережі – тришарового персептрона.

Топологія штучної нейтронної мережі зовсім не відповідає топології мережі біологічних нейронів. Штучна мережа для задачі розпізнавання (персептрон) має малу кількість шарів, кількість входів, що відповідає розмірності простору ознак, та кількість виходів, що визначається кількістю класів. Для забезпечення класифікації на кожному нейроні повинні бути задані коректні

значення усіх вагових множників, що забезпечується їх формуванням в процесі навчання мережі, зазвичай на основі попередньо класифікованої множини навчальних об’єктів.

3.3. Стиснення даних

3.3.1. Загальні відомості про стиснення даних

Будь-яка інформація (приладові дані, текст, зображення, звук, відеопотік) представляється для збереження, накопичення та обробки у формі деякого потоку даних, тобто числових значень. За це представлення відповідає такий метод обробки, як кодування. Оскільки воно не є однозначним (можуть використовуватись різні системи кодів та методи кодування), та ж сама інформація може бути представлена різними даними. Більш того, ці дані можуть мати різну довжину (кількість байтів).

Наприклад, довільний текст можна закодувати однобайтовим кодом, відповідно до якого літері відповідає один байт. Для літерного алфавіту (не ієрогліфічного!) однієї мови такої кількості даних достатньо, але потреба використання в єдиному файлі декількох алфавітів викликає певні ускладнення. Той самий текст може бути представлений юнікодом, тобто кодуванням літери двома байтами, за рахунок чого кодування стає універсальним, для будь-якого алфавіту існує свій піддіапазон кодів в рамках загального (що неможливо забезпечити при однобайтовому кодуванні). Але при такому кодуванні при збереженні об’єму інформації об’єм даних збільшується вдвічі. Саме це є ціною забезпечення певної універсальності в даному випадку.

Можна навести й інші приклади, коли об’єм даних значною мірою залежить від способу кодування. Але існує ще одна проблема, вирішення якої є дуже принциповим для забезпечення ефективності використання комп’ютерних технологій. Досить часто дані при кодуванні формуються як результат розбиття потоку інформації на послідовність невеликих порцій інформації. В наведеному прикладі такою порцією є літера, при кодуванні звуку це може бути один відлік сигналу, для зображень – один піксель, до послідовності яких зводиться зображення. Причиною такого способу кодування є надзвичайна простота алгоритмізації процесу кодування і швидкість його роботи. Але таке просте кодування часто призводить до появи надлишкових («зайвих») даних у потоці коду, тобто таких даних, які можна було б замінити на більш короткі.

Як приклад розглянемо кодування на рівні слова замість окремої літери. У мовах спілкування людини досить обмежена кількість слів (близько 10^5). З врахуванням специфічної форми узгодження закінченнями отримаємо 10^6. Якщо для кожного з можливих слів виділити окремий код, то для такого кодування вистачить трьох байтів на одне слово, а якщо не вирівнювати код на довжину байту, то і двох з половиною. Якщо взяти до уваги, що середня довжина слова (залежить від мови) сягає 5-7 літер, і навіть більше, при кодуванні на рівні слова отримаємо зменшений, приблизно вдвічі об'єм даних. Зрозуміло, що отриманий таким кодуванням потік даних є неприйнятним для людини і підлягає на фінальному етапі перекодуванню у звичний літерний потік.

Відповідно, ***стиснення даних – це процедура перекодування даних, яка дозволяє зменшити їх обсяг при умові збереження можливості подальшого відновлення (зворотного перетворення) даних***.

Зазвичай процедуру стиснення застосовують не безпосередньо до деякої інформації у незручному для цифрової обробки вигляді, а вже після певного простого кодування, яке, наприклад, застосовується для інших видів обробки інформації даного типу. Відповідно, при відновленні стиснених даних (будемо називати це розгортанням) отримують потік даних, аналогічний за структурою та інформаційним змістом до того, що стискався.

Процедура стиснення вимагає додаткових ресурсів (часу на виконання обробки, додаткового об'єму пам'яті для збереження проміжних даних), але для запису або передачі інформації вона може бути надзвичайно корисною. Будь-який канал має обмеження за швидкодією, тобто за кількістю даних, які можна передати в одиницю часу. У багатьох випадках буде більш ефективним витратити певні обчислювальні ресурси для перекодування даних з боку відправника та після отримання адресатом, але більш ефективно використати ресурси самого каналу. Такою ж є ситуація з виділенням ресурсів для накопичення інформації, коли стиснення дозволяє на тому ж об'ємі носія зберігати більшу кількість інформації.

Ефективність визначається коефіцієнтом стиснення, який визначається як відношення обсягу вхідних (нестиснених) даних до обсягу стиснених, тобто

$$C_R = \frac{S_o}{S_c}.$$

Часто використовують такий термін, як надлишковість даних (точніше буде надлишковість по відношенню до певного методу

стиснення), яка слідує з цього виразу. Надлишковість фактично характеризує, яку частину даних можна видалити в процесі стиснення.

$$R_d = 1 - \frac{1}{C_R}.$$

Тепер розглянемо зворотну операцію – відновлення (розгортання) даних. Для значної кількості типів інформації обов'язковою вимогою є повне збереження (на рівні співпадіння кожного біту) блока даних при послідовному застосуванні стиснення та розгортання. Такий спосіб стиснення називається ***безвтратним***. Зрозуміло, що безвтратне стиснення є універсальним, тобто може бути застосовано до будь-якого типу інформації – виконуваного файлу програми, тексту, приладових даних, звуку, зображення. Але для типів інформації, які відрізняються найбільшими об'ємами, таке безвтратне стиснення зазвичай є недостатньо ефективним для забезпечення більшості потреб використання цих типів інформації.

До найбільш ресурсоємних типів інформації відносяться (впорядковано за зростанням):

- звукова інформація;
- статичне зображення;
- динамічне зображення (відеопотік).

Розглянемо роботу з цими видами інформації детальніше. Звукова інформація за вимоги забезпечення високої якості вимагає запису двох (або навіть більшої кількості) каналів з 12-бітовим відліком та з частотою дискретизації 36–44 кГц, що відповідає (за теоремою Котельникова, п. 1.4.1) граничній частоті 18–22 кГц. Найчастіше використовується 44 кГц. Таким чином, секунда нестисненого звукового потоку займає 132 КБ, тобто звичайний компакт-диск (CD), який розроблявся саме з вимог до запису звуку, дозволяє записати приблизно 80 хвилин стереозвуку високої якості.

Високоякісна фотографія має розміри не гірше, ніж 2000x3000 пікселів, для запису кожного з яких потрібні 3 байти. Тобто таке зображення буде вимагати обсягу даних приблизно 20 МБ, той самий компакт-диск дозволить зберігати до 30 фотографій.

Відеопотік якості аналогового телебачення передбачає 25 кадрів (окремих зображень), кожен з яких займає приблизно 1 МБ (різниться відповідно до вибору стандарту аналогового телебачення), тобто секунда нестисненого телевізійного потоку вимагає 25 МБ, а компакт-диск буде вміщувати тільки приблизно півхвилинний ролик. Якщо ж зробити подібну оцінку для сучасного стандарту кодування відео (HD), для якого кадр у форматі 1080p займає 6.4 МБ, взагалі отримаємо приблизно 4 с. Навіть якщо замість CD використати DVD,

який розроблявся з врахуванням вимог запису відеопотоку телевізійної якості, отримаємо тільки декілька хвилин звичайного відео. Стає зрозумілим, що для зазначених типів інформації, особливо для відеопотоку, ефективність стиснення стає критичною. При цьому заради забезпечення високої ефективності стиснення є сенс навіть пожертвувати частковою втратою інформації. Цей спосіб обробки даних визначив такий потужний клас алгоритмів як ***втратне стиснення***.

При обробці звукового потоку та графічної інформації слід брати до уваги такі моменти:

- нестиснені дані зазначених типів інформації отримують з певним обмеженням співвідношення сигнал/шум (приладові обмеження перетворення);
- при спостереженні статичного зображення дрібні деталі несуть додаткове змістове навантаження в порівнянні з елементами зображення великого розміру;
- при сприйнятті динамічного потоку (звуку та відео) високої якості частина інформації не встигає бути обробленою мозком людини, тобто такий потік зазвичай є надлишковим щодо сприйняття людиною.

Таким чином, втрати інформації, які не перевищують рівня шуму, є взагалі непринциповими. При відтворенні дрібних деталей навіть для статичного зображення можна використати деяке погіршення якості. При стисненні відеопотоку є сенс використати особливості сприйняття людиною зображення та звуку, що супроводжує відеопотік. Слід брати до уваги також і можливість використання навіть помітних втрат інформації в кадрі заради забезпечення меж пропускної здатності каналу, що використовується. Це пов’язано з меншою критичністю для спостерігача певних дефектів зображення, ніж відставання від реального масштабу часу або стрибкоподібні пропуски потоку інформації, які виникають через недостатню швидкість передачі даних.

При використанні втратних методів є потреба у введенні деякого критерію вірності відновлення, який дозволяє визначати прийнятність втратності інформації:

$$e(x,y) = \hat{f}(x,y) - f(x,y),$$

де $f(x,y)$ та $\hat{f}(x,y)$ відповідно вхідне та оброблене зображення. Наприклад, для дискретного просторового сигналу (цифрового зображення) його можна задати таким чином:

$$e_{скв} = \left[\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left(\hat{f}(x,y) - f(x,y) \right)^2 \right]^{1/2}.$$

Розглянемо основні джерела надлишковості даних для різних видів інформації, які можна використати для реалізації стиснення даних.

На прикладі текстового повідомлення як сукупності слів, в порівнянні з кодуванням окремими літерами вже було видно, що певні подовжені фрагменти потоку інформації можуть забезпечити стиснення. Причина такої можливості криється фактично у наявності повторення слів у тексті, що і робить його надлишковим у порівнянні з випадковою послідовністю літер. Крім цього слід брати до уваги, що одні літери у тексті зустрічаються частіше за інші (порівняйте, наприклад, «о» та «ш»), і це також можна використати.

У звуковому потоці може спостерігатись схожість як окремих фрагментів у різні моменти часу, так і між каналами у випадку стерео.

На статичних зображеннях можуть зустрічатись схожі фрагменти, блоки з повторення однакових пікселів. І в цьому випадку нерівномірність статистичного розподілу значень пікселів (як у прикладі з літерами) також може бути використана.

Для відеопотоку можна додатково до цього використати певну схожість сусідніх кадрів між собою. І, як вже було зазначено, є певні особливості сприйняття, які дозволять розділити інформацію звуку чи зображення на більш та менш важливу.

3.3.2. Безвтратні методи стиснення

Групове стиснення. Найбільш простим для розуміння ідеї безвтратного стиснення є алгоритм групового стиснення, з якого і почнемо розгляд. Одразу зауважимо, що цей метод є досить специфічним і прийнятний тільки для стиснення зображень, при чому досить вузького діапазону застосування. З іншого боку групове стиснення використовується як один з етапів високоефективних втратних методів, які будуть розглянуті у наступному параграфі.

Зображення може бути представлене послідовністю пікселів за допомогою растеризації – послідовного приєднання наступного рядка до кінця попереднього. Для деякого класу зображень (креслень або анімаційних кадрів з великими площами, заповненими однаковими значеннями пікселів) у такому потоці часто будуть зустрічатись послідовності однакових байтів. Зрозуміло, що зображення для цього повинне бути саме з кодуванням пікселя байтом. Таким чином, справжнє кольорове зображення з трибайтовим (RGB) кодуванням

пікселя для застосування такого алгоритму не підходить. Формально можна було б реалізувати цей алгоритм окремо для трьох кольорових шарів, але подібне зазвичай не використовується через непридатність метода для стиснення таких класів зображень, які вимагають високоякісного відтворення кольору, наприклад, фотографій.

Найчастіше серед можливих реалізацій групового стиснення використовуються дві:

- PackBits (розробка Apple);
- RLE (використовується у стандартизованих графічних форматах для стиснення індексованих зображень).

Нагадаємо, що індексацією (коли йде мова саме про зображення) називають представлення кольору кожного пікселя кодом, що дорівнює номеру рядка палітри – таблиці кольорів, кожен рядок якої вміщує RGB тріаду. При цьому палітра формується за найбільш часто застосованими у зображенні значеннями RGB-тріад, а кольори, які не потрапили у палітру, замінюються на найближчі з наявних в ній.

Кожен з двох вище згаданих алгоритмів спирається на вимогу того, що кодувальник повинен закодувати послідовність N однакових байтів так, щоб забезпечити найкоротший запис цієї послідовності. Але при цьому необхідно передбачити можливість запису без перекодування деякої послідовності байтів, у якій немає повторень, бо інакше такі послідовності кодуються із значним «розбуханням». Розглянемо реалізацію алгоритму на прикладі RLE.

Нехай вхідна послідовність має вигляд

aaaabccccccaaaccccccddddaaaaa.

Послідовність однакових байтів запишемо як кількість повторень та значення байта після нього. Таким чином, уся послідовність прийме вигляд:

4a1b7c3a7c4d5a.

Декодувальник розглядає пару байтів також як кількість повторень та значення, що дозволяє виконати коректне розгортання. Але у випадку, коли зустрінеться деяка послідовність байтів, що не повторюються (вона називається літеральною групою)

aaaabccc**abdcadad**ccccaaaccccccddddaaaaa,

такий простий спосіб запису буде її розтягувати вдвічі, група байтів **abdcadad** перетвориться у **1a1b1d1c1a1d1a1d**. Загальна ефективність стиснення буде визначатись двома факторами:

- стисненням груп байтів, що повторюються;
- розбуханням літеральних груп.

Відповідно, що з цього буде переважати і визначає, чи взагалі стиснення буде досяжним. Для забезпечення компактності запису літеральної групи вона записується двома спеціальними байтами та самою групою, що йде за ними. Перший спеціальний байт є нульовим і забезпечує розпізнавання декодером ознаки літеральної групи, другий байт – довжина літеральної групи. Відповідно, наведений ускладнений приклад потоку, що стискається, буде закодований таким чином:

4a1b7c**08abdcadad**3a7c4d5a.

В алгоритмі PackBits на відміну від RLE для розпізнавання літеральної групи для групи повторень і літеральної групи використовуються відповідно додатні та від'ємні числа формату «знаковий байт».

Статистичне стиснення. Значно більш універсальним є метод Хаффмана, який є представником так званих статистичних алгоритмів. Сама назва статистичних алгоритмів говорить про те, що використовується деяка статистика значень байтів потоку, що стискається. В даному випадку використовується нерівномірність статистичного розподілу значень байтів, тобто те, що одні байти зустрічаються частіше, інші рідше. Зрозуміло, що такій ситуації відповідає і зміст виконуваного файлу програми, і текст, і зображення. Тобто на відміну від групового стиснення методи цієї групи є універсальними.

Можливість стиснення полягає у тому, що результатом кодування є значення, не вирівняні на довжину байту. Таким чином, для кодування довільного байту вхідного потоку будемо використовувати коди, частина яких є коротшою за байт (8 бінарних розрядів). Зрозуміло, що більш коротких кодів буде менше, ніж 256 можливих значень байту, тобто для певної частини значень доведеться використовувати коди, довші за 8 бінарних розрядів. Це означає, що частина потоку запишеться більш короткими фрагментами бітового потоку, а частина – більш довгими. Якщо короткі коди будуть відповідати значенням, які зустрічаються частіше, стиснення буде більшим, ніж розбухання, що забезпечує бажаний позитивний баланс.

Для реалізації цього алгоритму необхідно сформувати спеціальну бітову таблицю кодування на основі частотного аналізу вхідних даних та приєднувати цю таблицю до стисненого потоку.

Таблиця кодування будується у вигляді асиметричного бінарного дерева (рис. 3.15.), тобто гілки вліво та вправо відповідають двом значенням біта, а шлях з вершини по значенням нулів та

одиниць виводить до вузла дерева, який відповідає одному з байтових значень вхідного потоку. Таким чином, коли шлях з елементів бітового потоку завершується, вибирається відповідний код, а новий шлях для наступного (ще нерозглянутого біта) стиснутого потоку відстежується знову з вершини дерева. Саме це дозволяє декодеру аналізувати бітовий потік з невизначеною довжиною елементів.

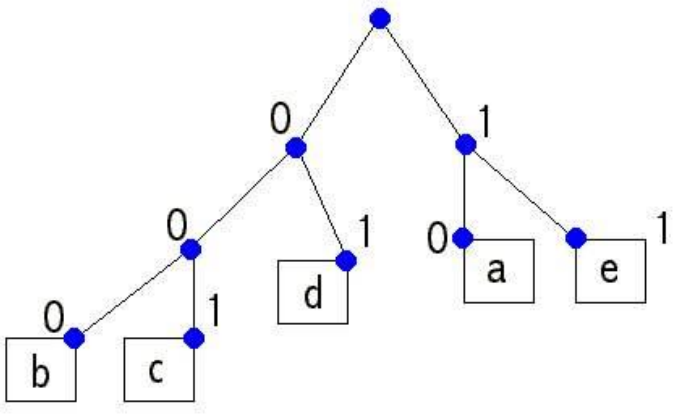


Рис. 3.15. Приклад дерева в алгоритмі Хаффмана

Залишилось з’ясувати, яким чином можна побудувати таке дерево. Для цього розраховується ймовірність кожного байтового значення у вхідному потоці. Далі розгортається ітераційний процес. У рамках кожної ітерації вибираються два значення, ймовірність яких є найменшою при поточному стані процесу. Цим значенням приписуються відповідно «0» та «1» як бінарний розряд їх коду (один крок шляху на дереві Хаффмана). Для переходу до наступної ітерації ця пара об’єднується у єдиний елемент, а ймовірність пари складається як сума двох ймовірностей. Таким чином, дерево нарощується знизу догори з подовженням вже створених раніше шляхів. Процес повторюється до перебору усіх можливих значень байтів вхідного потоку. При цьому елементи з найбільшою ймовірністю включаються у розгляд на останніх ітераціях, і їм будуть відповідати коротші шляхи (коди), а найменш ймовірним – довші.

Ще одним яскравим представником статистичних алгоритмів є арифметичне стиснення. Воно використовує кодування деякої послідовності байтів двійковим дробом мінімально можливої довжини. В даному випадку, як і в методі Хаффмана, першим кроком є дослідження статистики розподілу байтів у потоці, що стискається. Оскільки сума цих ймовірностей дорівнює одиниці, можна побудувати відрізок одиничної довжини та розбити його на фрагменти, довжини яких будуть відповідати ймовірностям окремих значень байтів вхідного потоку(рис. 3.16., лівий відрізок).

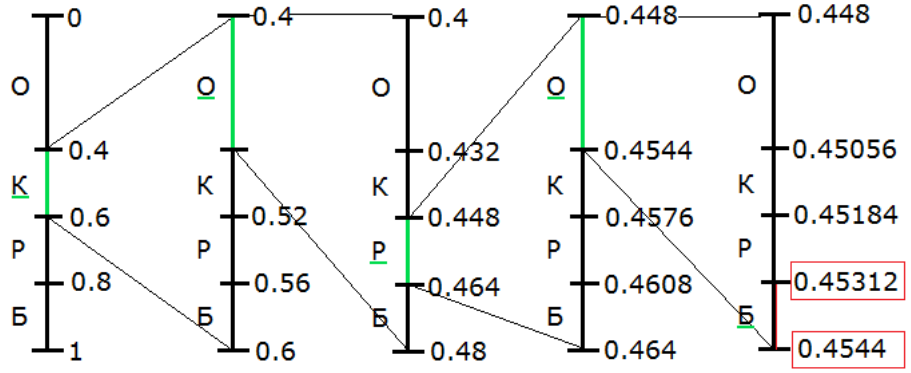


Рис. 3.16. Арифметичне стиснення

Тепер перейдемо до послідовного побайтового аналізу вхідного потоку. Оберемо той фрагмент, який відповідає значенню першого байта. Цей фрагмент можна також розбити пропорційно ймовірностям (аналогічне вже описаному кроку), на рис. 2.17., другий зліва відрізок. Для зручності на малюнку він розтягнутий до розмірів попереднього випадку. Вибираємо на ньому фрагмент, що відповідає другому символу у потоці.

Далі описані кроки повторюємо. Відповідно, деякій вже розглянутій послідовності байтів вхідного потоку відповідає певний фрагмент першого одиночного відрізка. Довільний дріб, значення якого вкладається у цей фрагмент, однозначним чином буде описувати розглянуту послідовність. Якщо додати, що чим більшим є фрагмент, тим менша кількість бінарних розрядів буде відповідати найкоротшому бінарному дробу, стає зрозумілим механізм роботи цього алгоритму.

Словникові алгоритми. У п. 3.3.1 як приклад стиснення був використаний словник, а саме справжній словник деякої мови, що робить такий спосіб стиснення занадто вузькоспеціалізованим. Але існують алгоритми, які створюють власний словник безпосередньо на основі оброблюваного потоку даних, що забезпечує універсальність та високу ефективність. Фактично, принцип стиснення збігається з тим, що було розглянуто, тобто використовується деяке кодування слів, які зустрічаються у потоці багато разів. Це і забезпечує стиснення, але з динамічним формуванням самого словника.

Як приклад розглянемо дуже популярний алгоритм LZW (Lempel–Ziv–Welch), тобто його назва походить від перших букв прізвищ вчених, які внесли певний вклад у створення цього алгоритму. Алгоритм був опублікований Велчем у 1984 році, на основі розвитку алгоритму LZ78, (Лемпел і Зів, 1978 рік).

На відміну від розглянутих вище статистичних алгоритмів даний алгоритм не вимагає приєднання до стиснутого потоку таблиці кодування завдяки тому, що і кодер і декодер можуть створити цю таблицю тільки з аналізу потоку даних, який вони обробляють. Розглянемо, яким чином можна цього досягти.

Введемо необхідні терміни:

- процес стиснення та розгортання виконують **«кодер»** і **«декодер»**, у випадках, коли немає алгоритмічної різниці в їх роботі, будемо використовувати загальний термін **«обробник»**;
- байт потоку, що стискається (формальну одиницю цього потоку), назвемо **«літерою»**;
- сукупність бітів, які є формальною наступною одиницею стиснутого потоку, назвемо **«кодом»**. Код не є вирівняним на довжину байту, а його довжина змінюється у процесі роботи обробника;
- **«словом»** будемо називати довільну послідовність літер, тобто елементів потоку що стискається;
- усю сукупність відомих на поточний момент слів будемо вважати **«словником»**. При цьому словник являє собою таблицю, кожен з рядків якої містить слово та код, що відповідає цьому слову. В процесі роботи стан словника змінюється додаванням нових рядків та збільшенням бітової довжини кодів;
- **«акумулятором»** вважаємо спеціальний буфер літер (масив), у якому кодер складає нову послідовність «дописуванням» літери в кінець послідовності, яка вже є в акумуляторі;
- для кодера **«вхідним потоком»** є сукупність літер нестиснутого повідомлення, для декодера – послідовність кодів (не вирівняних на довжину байту, тобто наступний елемент виділяється відповідно до довжині кодів при поточному стані словника).

Початковий стан обробника створюється очисткою буферів та формуванням початкового стану словника («за домовленістю») як сукупності усіх можливих однолітерних слів та бітових кодів, що повторюють байти цих літер. Оскільки код співпадає зі словом (поки що), зрозуміло, що початковий стан словника для кодера і декодера однаковий.

У процесі стиснення (до кінця потоку) кодер у циклі виконує наступну послідовність дій:

а. Виділяється наступна літера X вхідного (нестиснутого) потоку;

б. Робиться пошук AX (змісту акумулятору A разом з літерою X) в словнику. Якщо є, то елемент X додається до акумулятора і керування передається на крок «а», інакше:

б1. У вихідний потік відповідно до змісту акумулятора A зі словника передається код;

б2. AX (зміст акумулятора A та літера X) вноситься як нове слово у словник з новим (наступним за зростанням) кодом. При нестачі кодів через обмеженість довжини до усіх кодів словника дописується нульове значення біту **зліва** (тобто з цього моменту довжина коду стає більшою);

б3. Літера X записується в акумулятор замість старого змісту, керування передається на крок «а».

Таким чином, кодер виконує кодування за поточним станом словника та одночасно з цим «нарощує» його з перспективою на повтори тих сполучень літер, які знайдені у поточному тексті. Тобто словник формується тільки з тих слів, які є присутніми в даному потоці.

Алгоритм декодування є дещо відмінним, оскільки вхідний потік складається не з літер, а з кодів, а доповнювати словник треба аналізуючи саме слова, тобто сукупності літер. При цьому декодер, так як і кодер, повинен формувати словник на крок пізніше декодування, спираючись у декодуванні на поточний стан словника. Для цього декодер має два буфери кодів – назвемо їх «перший буфер» B1 та «другий буфер» B2. Спочатку (одноразово) з вхідного (стиснутого) потоку в другий буфер виділяється код B2 і відправляється у вихідний потік (через те, що стан словника є початковим, код співпадає зі словом і його перекладати у слово немає потреби, а довжина коду 8 бітів). Далі (до кінця потоку) виконується наступне:

а. Виділяється наступний код із вхідного потоку та записується у перший буфер B1, при цьому його довжина (в бітах) визначається довжиною коду в словнику при його поточному стані;

б. За допомогою поточного стану словника робиться зворотний переклад коду з першого буфера у слово S(B1) і виводиться у вихідний потік;

в. Виділяється перша літера X слова S;

г. За допомогою поточного стану словника робиться переклад коду з другого буфера в слово A(B2);

д. У словник додається слово AX (слово A, що відповідає попередньому коду з додаванням першої літери з останнього коду), код у словнику формується аналогічно тому, як це відбувається при роботі кодера;

е. В другий буфер переносяться дані з першого буфера (B2=B1).

Приклад декодування « A W E D 256 E 260 261 257 B 260 T », що було отримане з кодування потоку «AWEDAWEAWEEAWEBAWET», наведено у табл. 3.1.

Таблиця 3.1. Приклад декодування (розгортання) потоку даних за алгоритмом LZW

B1	B2	S (вихід)	A(B2)	X	Занесення у словник
A	A	A			
W	A	W	A	W	256= AW
E	W	E	W	E	257=WE
D	E	D	E	D	258=ED
256	D	AW	D	A	259=DA
E	256	E	AW	E	260=AWE
260	E	AWE	E	A	261=EA
261	260	EA	AWE	E	262=AWEE
257	261	WE	EA	W	263=EAW
B	257	B	WE	B	264=WEB
260	B	AWE	B	A	255=BA
T	260	T	AWE	T	255=AWET

Словникові методи стиснення саме завдяки універсальності за даними, які стискаються, і високій для безвтратних методів ефективності активно використовуються в архіваторах (у різних реалізаціях) та у стандартних графічних форматах (зазвичай LZW).

Алгоритми з передбаченням. При кодуванні сигналів (наприклад, звуку або зображення) об’єм даних перед застосуванням інших безвтратних алгоритмів можна зменшити використанням передбачення. Така технологія отримала назву Adaptive Differential Pulse Code Modulation (ADPCM).

Візьмемо деякий періодичний сигнал, наприклад, зображення, рядок якого має яскравість, що залежить від горизонтальної координати періодичним чином. Якщо, починаючи з другого періоду, віднімемо від поточного значення сигналу (в даному випадку

просторового) сигнал, зсунутий на період, отримаємо нульові значення в усій частині рядка, що залишилась. Зрозуміло, що цей приклад виключно модельний, але подібне передбачення може для багатьох реальних випадків забезпечити збільшення ефективності. Такий спосіб стиснення використовується у деяких ускладнених методах стиснення зображень, наприклад безвтратному JPEG.

3.3.3. Втратне стиснення зображення

Для стиснення мультимедійної інформації розглянуті вище безвтратні методи у багатьох випадках неприйнятні через недостатню їх ефективність. Відповідно, доводиться жертвувати частковою втратою інформації заради компактності даних. При цьому, як вже зазначалось, повинні бути враховані особливості сприйняття такої інформації людиною.

Найпопулярнішим алгоритмом стиснення статичних зображень фотографічної якості, які серед зображень є найбільш ресурсоємними, є JPEG. Цей алгоритм стиснення був розроблений з врахуванням контенту таких зображень та особливостей його сприйняття. Він підтримує повноцінні кольорові зображення (RGB) та зображення у градаціях сірого. Загальна схема алгоритму показана на рис. 3.17.

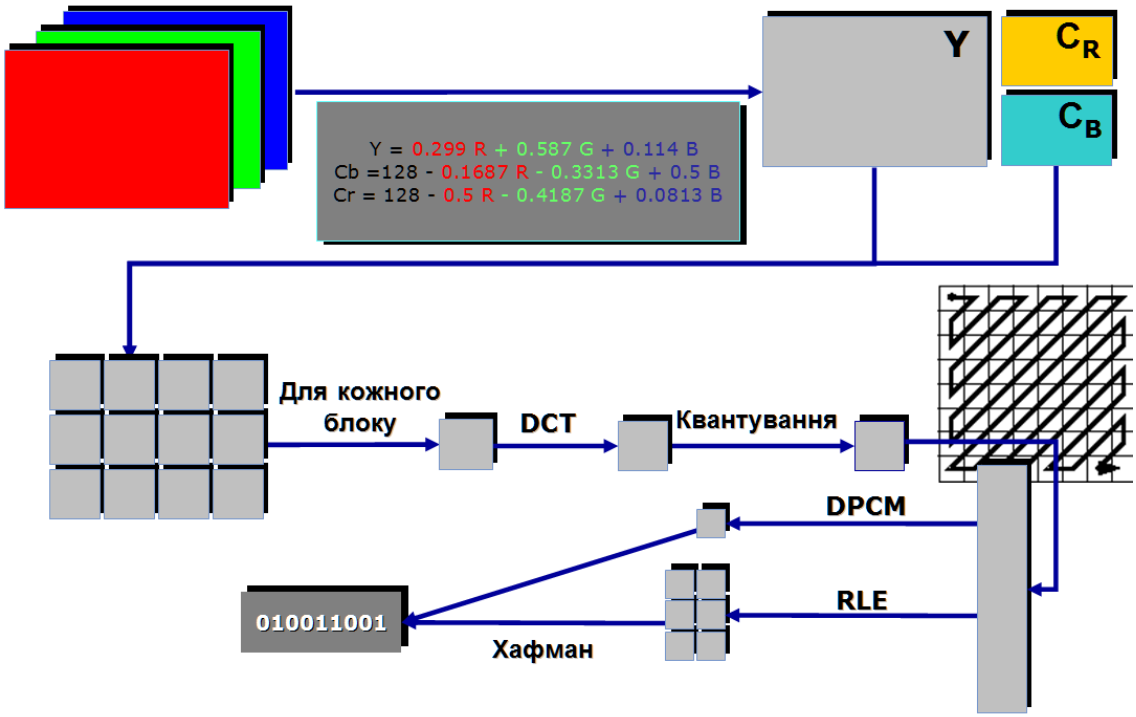


Рис. 3.17. Загальна схема кодування за алгоритмом JPEG

Кольорове зображення може бути представлене як сума трьох одноколірних градаційних зображень (червоного R, зеленого G, синього B). Але око інформацію у цих шарах сприймає не зовсім рівноцінно, оскільки серед трьох видів світлочутливих клітин, які сприймають кольори, найбільш чутливими є ті, що сприймають саме зелений. Більш того, при аналізі зображення більш суттєвими для мозку людини є перепади яскравості, ніж кольору. З врахуванням таких особливостей для кольорового зображення в даному алгоритмі передбачений перехід до іншого представлення кольору $RGB \to YC_BC_R$ з використанням отриманих емпірично вагових множників при RGB складових:

$$\mathrm{Y} = 0.299\,\mathrm{R} + 0.587\,\mathrm{G} + 0.114\,\mathrm{B}$$
$$\mathrm{C_B} = 128 - 0.1687\,\mathrm{R} - 0.3313\,\mathrm{G} + 0.5\,\mathrm{B}$$
$$\mathrm{C_R} = 128 + 0.5\,\mathrm{R} - 0.4187\,\mathrm{G} + 0.0813\,\mathrm{B}$$

Крім цього, опціонально (за потреби забезпечення високої ефективності стиснення при умові можливості внесення суттєвіших втрат) на даному етапі може бути використаний ще один потужний інструмент – проріджування даних. При використанні проріджування кількість пікселів у рядку та кількість самих рядків різницевих шарів, які відповідають саме за кольоровий контраст, зменшуються вдвічі. Саму це враховує меншу значимість кольорового переходу в порівнянні з перепадом яскравості, про яке вже згадувалося. Відповідно, ця опція є першим принципово втратним перетворенням в даному алгоритмі.

Далі кожен з отриманих в результаті попереднього перетворення шарів даних обробляється незалежно. Для «сірих» зображень алгоритм починається з цього моменту, оскільки шар Y фактично і відповідає такому зображенню.

Наступний етап обробки враховує те, що дрібні елементи зображення є менш критичними для сприйняття. Математично це може бути представлено розкладом по просторовим гармонікам (рис. 3.18.), тобто зображенням, які мають певну кількість хвилеподібних варіювань по горизонтальній та вертикальній осям (кількість варіювань по осям є індексами просторової гармоніки).

Для зменшення ресурсоємності обробки розкладається не усе зображення цілком, а квадратні блоки 8x8 пікселів, на які воно попередньо розбивається. Розміри блока обрані емпіричним чином як компроміс між якістю та швидкістю обробки Відповідно кожен із блоків представляється як деяка зважена сума 64 зображень показаних на рис. 3.18. У вигляді формули це перетворення (DCT) може бути записане таким чином:

$$F(u,v)=\frac{1}{4}C(u)C(v)\left[\sum_{x=0}^{7}\sum_{y=0}^{7}f(x,y)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}\right],$$

де

$$C(z)=\begin{cases}1/\sqrt{2}, & z=0\\ 1, & z\neq 0\end{cases}.$$

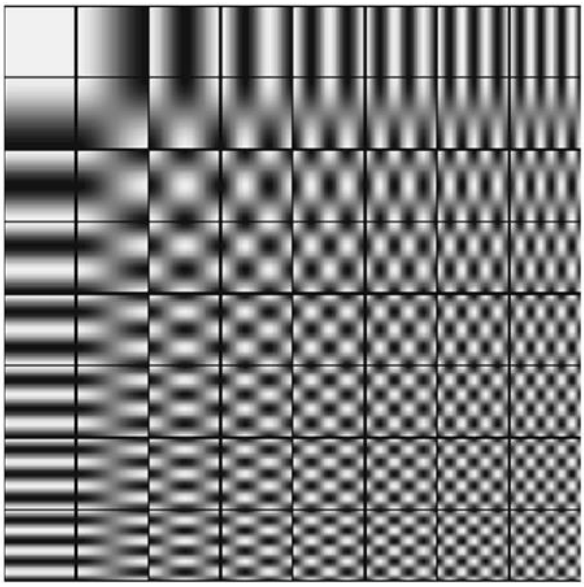


Рис. 3.18. Сукупність функцій, по яких виконується розклад при дискретному косинусному перетворенні (DCT)

В результаті блок пікселів з 64 скалярних значень утворює нові 64 скалярних значення, які є вагами (амплітудами гармонік) у вказаному розкладі. Більш того, перетворення виконується зі збільшенням розрядності з 8 бітів до 12, тобто кількість даних при виконанні даного кроку навіть збільшується. Але саме цей крок перерозподіляє інформацію на більш корисну, яку необхідно записувати максимально докладно, і таку, яку можна частково відкинути. Відповідно, амплітуди гармонік з меншими значеннями індексів повинні зберігатись з більшої кількості бітів, ніж ті, що мають більші індекси. Цю операцію відкидання частини розрядів називають квантуванням.

При квантуванні використовується основний параметр настроювання алгоритму, який відповідає якості стиснутого зображення. Тобто менше значення цього параметра відповідає погіршенню якості, але більшому стисненню. Фактично, при меншому значенні цього параметра зі зростанням індексів просторової

гармоніки швидше, ніж при більшому значенні, зменшується кількість бітів, що зберігаються при квантуванні. Саме квантування є обов'язковим джерелом втратності даного алгоритму. Якщо згадати про розглянуте вище проріджування, то воно є лише опцією і, зазвичай, вмикається при умові, що параметр якості є меншим за деяке порогове значення.

Квантування визначає специфічні артефакти стисненого за допомогою даного алгоритму зображення, а саме – появу так званого "ефекту Гіббса", коли навколо елементів зображення з різким переходом кольору або контрастності утворюється ореол. Крім того, при дуже малому значенні параметру якості взагалі на зображенні може проявитись характерна мозаїка з клітинок-блоків (8x8).

Крім зазначених кроків для досягнення високої ефективності стиснення виконуються ще деякі додаткові перетворення даних кожного з блоків, зведених до косінусного представлення. При цьому застосовується зміна порядку проходження растра таким чином, що поруч розміщуються амплітуди гармонік приблизно однакової значимості (рис. 3.19.).

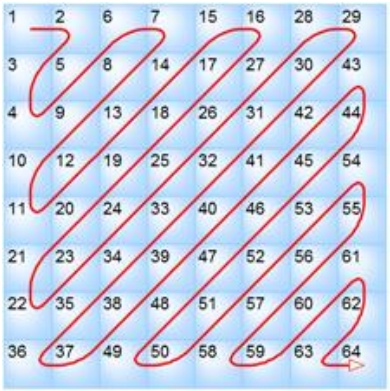


Рис. 3.19. Зміна порядку проходження растру блоку для збільшення ефективності наступного застосування групового стиснення

Далі використовуються безвтратні методи – групове стиснення в рамках вже пересортованого блока та метод Хафмана взагалі для зображення. Все це разом забезпечує дуже високу ефективність стиснення для більшості зображень навіть при непомітних на око втратах. Але слід пам'ятати, що алгоритм реалізує найкраще співвідношення якість/розмір саме для фотографій, а на синтезованих зображеннях, особливо таких, що мають чіткі контрастні лінії, можуть з'являтись небажані артефакти, особливо коли зменшення параметра якості включає механізм проріджування даних.

Серед інших найвідоміших методів, які використовують втратність для забезпечення високої ефективності стиснення, можна виділити використання Wavelet-перетворення. Цей метод розглядається як один із перспективних, перевищує JPEG за ефективністю при аналогічній якості, хоча значно програє за часом кодування, що дещо заважає його розповсюдженню на сучасному етапі. Особливо якщо взяти до уваги, що виграш в порівнянні з конкурентом не такий потужний, як той, що вивів JPEG у лідери в конкурентній боротьбі форматів у порівнянні з попередниками, що використовують безвтратне стиснення.

Більша ефективність використання Wavelet забезпечується тим, що, на відміну від розглянутого базису просторово-періодичних функцій, цей метод фактично використовує базис аперіодичних хвилькових функцій, які мають просторову обмеженість, що краще узгоджується з просторовими особливостями елементів оброблюваного зображення. Таке перетворення, наприклад, використовується у форматі JPEG2000, стандартизованому, хоч поки що і не такому популярному, як більш традиційний JPEG,.

3.3.4. Втратне стиснення відеопотоку

У п. 3.3.1 було зазначено, що найбільш критичним щодо забезпечення ефективності стиснення є кодування відеопотоку, тобто динамічного зображення. Саме тому так багато уваги приділяється оптимізації методів роботи з цим типом інформації. Найбільш поширеними засобами стиснення відеопотоку є алгоритми групи MPEG, точніше декілька поколінь розвитку загального алгоритму, який, як і розглянутий у попередньому параграфі JPEG, є деяким послідовним ланцюжком перетворення даних, більшість з яких є способами оптимізації даних для остаточного застосування безвтратного стиснення. Як і аналогічний алгоритм стиснення статичної графіки, даний алгоритм враховує особливості сприйняття інформації людиною та максимально можливим чином використовує корельованість сигналу, в цьому випадку вже не просторового (статичне зображення), а часово-просторового.

Відеопотік являє собою сукупність кадрів, кожен з яких може бути закодований незалежним чином, тобто з врахуванням тільки просторової корельованості в межах кадру. Але у цьому випадку не буде використана часова корельованість, тобто схожість даного кадру на попередній. Найпростішим способом врахування такої схожості є використання вже розглянутого (п. 3.3.2) віднімання попередніх даних (ADPCM), для чого можна використати попередній кадр. Такий метод

є основою алгоритму moving JPEG. Цей спосіб стиснення є досить простим, але далеким від оптимального.

Основою оптимізації часової корельованості при MPEG-стисненні є розбиття потоку на декілька видів кадрів:

- I-кадр (Intraframe);
- P-кадр (Predictive);
- B-кадр (Bi-directional).

I-кадр є повністю незалежним від інших зображенням, кодується схожим до JPEG чином. Саме завдяки цьому цей кадр може бути точкою входу у відеопотік, що відтворюється.

P-кадр включає у себе зміни, що відбулися з часу попереднього кадру, тому розраховується на основі кадру, на який має посилання, не обов'язково I-кадру, тобто можлива реалізація послідовного відтворення P-кадрів. Зрозуміло, що цей кадр буде компактнішим за Intraframe (приблизно вдвічі). Гіпотетично можна було б реалізувати весь потік зі стартового I-кадру і подальшої послідовності P-кадрів. Реально така структура потоку не використовується, в основному через накопичення похибки при ланцюжковому обчисленні. Крім того деяким додатковим ускладненням алгоритму можна досягти ще більшої ефективності стиснення.

Більшу ефективність в порівнянні з P-кадром забезпечує B-кадр. Для його розрахунку треба мати як попередній (не обов'язково найближчий), так і наступний кадри (тобто використовується не одне, а два посилання). Звідси і виникла назва цього типу кадру. Таким чином у кадрі цього типу кодуються відмінності не від попереднього, а від інтерпольованого за двома кадрами, що і забезпечує більшу компактність (приблизно у два рази). Відповідно, є сенс використовувати послідовність типу

$$I_0 - P_1 - B_2 - B_3 - P_4 - B_5 - B_6 - P_7 ...$$

Таким чином, P_1 посилається на I_0; P_4 на P_1; B_2 та B_3 на пару (P_1, P_4) і далі за ланцюжком аналогічним чином. Якщо цей ланцюжок подовжити, стає зрозумілим, що за рахунок того, що при розгортанні поточного кадру у нього вносить похибку той кадр, на який цей кадр має посилання (у якому вже є похибка відновлення), похибка буде накопичуватись. З метою її обмеження потік розбивається на фрагменти певної довжини (задається за деякою домовленістю), які починаються незалежними I-кадрами. Сукупність кадрів від I-кадру до наступного I-кадру (не включаючи останній) створює так звану групу кадрів – GOP (Group of pictures). Її довжина обмежується від 8 до 24.

Загальна схема кодування P-кадру показана на рис. 3.20.

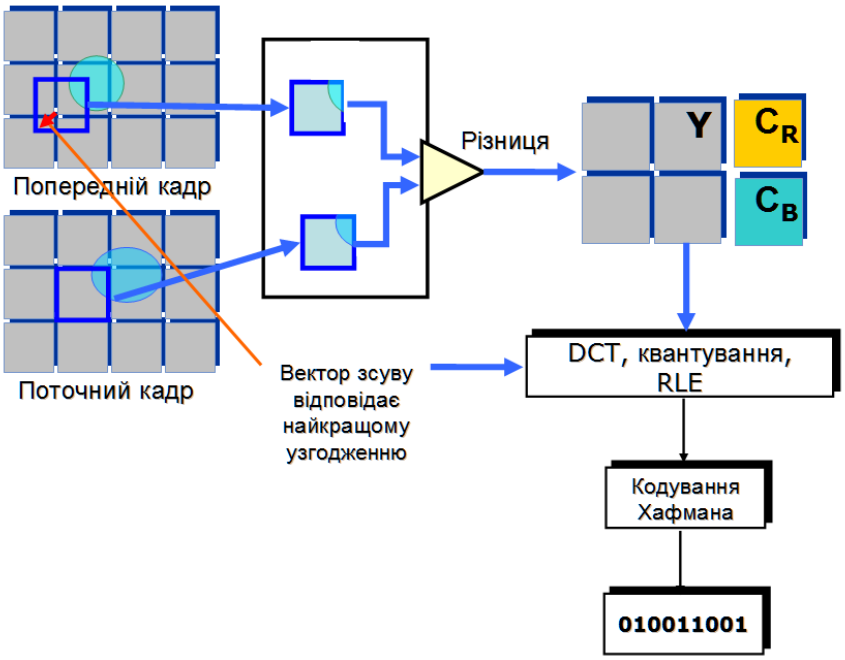


Рис. 3.20. Загальна схема кодування P-кадру в алгоритмі MPEG

Як і в алгоритмі JPEG, використовується розбиття на блоки, DCT та квантування, але розмір блоку є більшим, 16x16. На першому етапі, як видно з рисунку, для забезпечення максимальної корельованості даного блоку з тим, який віднімається, підбирається вектор зсуву за умовою максимуму кореляційної функції.

Для B-кадру алгоритм схожий з тією відмінністю, що замість різниці з блоком одного кадру (на який є посилання), різниця береться з результатом інтерполяції блоків двох кадрів і, відповідно, підбираються два вектори зсуву.

Сучасні реалізації, наприклад, кодек H.264 (в рамках MPEG-4) мають деякі додаткові засоби збільшення ефективності (правда, ціною суттєвого зростання ресурсоємності). Для цього кодека слід виділити:

- більш гнучке багатокадрове передбачення (до 32 посилань);
- компенсацію руху зі змінним розміром блоку (від 16x16 до 4x4), що дозволяє більш точно виділяти області руху;
- екстраполяцію векторів руху за межі кадру (за потреби);
- покращені методи фільтрації з метою зменшення зубчастості країв;
- більш гнучке застосування проріджування даних.

3.3.5. Стиснення звукового потоку

Крім стиснення графічної інформації доцільно одразу ознайомитись із стисненням звуку, який зазвичай супроводжує відеопотік.

Для кодування звуку серед втратних методів стиснення можна виділити два напрямки:

- логарифмічне кодування;
- спектральне кодування.

Логарифмічне кодування. Найпростішою операцією стиснення звукової інформації при фактичному збереженні її якості є перехід від рівномірної шкали квантування до нерівномірної. Така операція є аналогічною використанню перед записом на деякий носій (наприклад, магнітну стрічку) обробки пристроєм із нелінійною передаточною функцією, що активно використовується в аналогових технологіях для зменшення шумів у випадку, коли динамічний діапазон носія гіршій за системи перетворення (відома технологія Dolby).

Такий метод стиснення інформації спирається на нелінійну (близьку до логарифмічної) шкалу сприйняття звуку людиною і неприйнятний, наприклад, для графічної інформації. Він не дуже ефективний, але простий для реалізації і добре вписується у потокову технологію, в тому числі з використанням ADPCM (п. 3.2.2.).

Спектральне кодування. Найбільш поширеним представником такого кодування є MP3. Цей метод був розроблений фірмою Fraunhofer IIS. Європейська корпорація THOMSON активно підтримала формат і за її підтримкою він став одним зі стандартів аудіостиснення сімейства MPEG1, MPEG2 і одержав назву MPEG Layer3, звідки пішло більш звичне скорочення MP3.

В основі забезпечення високоефективного стиснення лежить сприйняття звуку семплами, тобто неможливість розпізнавання надто коротких фрагментів, та використання так званої психоакустичної моделі обробки звукових даних. Вона базується на тому, що:

- При наявності гучного сигналу крива порогу чутності вуха піднімається вище, погіршуючи сприйняття слабких складових;
- Вухо не в змозі розрізнити звуки з близькими частотами (цей ефект називається частотним маскуванням);
- В рамках маскування зазвичай виділяють 25 частотних смуг, які називають критичними. В області низьких частот критичні смуги приблизно сягають 100 Гц, на середніх частотах – 300 Гц, на високих (звукових) частотах – 4 КГц;

- Гучний семпл маскує наступний (цей ефект називають передмаскуванням) на час 20–200 мс.

Алгоритм MP3 є досить гнучким за налаштуванням, в основу якого покладене принципове для потокових методів обробки інформації обмеження за пропускною здатністю системи. Степінь стиснення визначається настроюванням кодера. Ширина потоку даних, bitrate (тобто кількість бітів в одиницю часу), при кодуванні сигналу, аналогічного стандарту нестиснутого звукового потоку CD Audio (44 кГц 16 біт, стерео) варіюється від 320 кб/с до 96 кб/с і навіть нижче. При цьому за вимоги забезпечення високої якості в рамках даного алгоритму при бітрейті 320 кб/с для кодування застосовується тільки безвтратне стиснення.

Розглянемо особливості процесу кодування. Першим етапом кодування є розбиття сигналу на ділянки – фрейми, кожний з який кодується і додається у кодований потік незалежно від інших. При цьому кожен фрейм може навіть кодуватися з різним настроюванням параметрів. Інформація про це настроювання вноситься у заголовки фреймів. Послідовність при відтворенні визначається порядком розташування фреймів.

Наступним етапом є кодування окремого фрейму. Спочатку сигнал за допомогою фільтрів розділяється на декілька частотних діапазонів. Сума отриманих субканалів еквівалентна вхідному сигналу. Слід зауважити, що частотний розклад обмеженої у часі ділянки сигналу, якою є фрейм, супроводжується помітною втратою якості на краях ділянки. Для зменшення таких втрат фрейми виділяються з деяким перекриттям по краях.

При потребі високоефективного стиснення надалі (як опція) використовується обробка з врахуванням психоакустичної моделі. При цьому для кожного отриманого раніше частотного субканалу визначається величина ефекту маскування сигналом сусідніх діапазонів і сигналом попереднього фрейму. Якщо потужність сигналу в ньому виявляється нижче деякого порогу чутності, який при розробці алгоритму був визначений емпіричним шляхом, для даного фрейму цей субканал відкидається.

Для даних, що залишилися, для кожного субканалу визначається умова округлення, тобто яку бітову розрядність треба залишити для забезпечення рівня втратності, нижчої за ефект маскування (втрата одного біта еквівалентна шуму квантування на рівні приблизно 6 дБ). Таким чином оптимізується відкидання непринципової для сприйняття інформації. Далі використовується безвтратне стиснення за алгоритмом Хаффмана.

Досить цікавими є й особливості кодування стереосигналу, які враховують часткову корельованість сигналу в цій парі каналів. За потреби забезпечення певного бітрейта (кількості бітів в одиницю часу) можуть використовуватись декілька режимів кодування стерео:

- dual channel;
- stereo;
- joint stereo.

При використанні режиму dual channel на кожен канал виділяється половина потоку, і канали кодуються повністю незалежно (як моносигнал). Цей режим є ефективним у випадку, коли канали містять принципово різну інформацію.

В режимі stereo канали кодується окремо, але кодер в процесі роботи перерозподіляє частки бітрейта по каналах в залежності від того, який з них у цей час є більш інформативним.

Joint stereo (MS stereo) – сигнал розкладається на середній між каналами і різницевий. При цьому другий кодується з меншим бітрейтом. Це дозволяє дещо збільшити якість кодування в звичайній ситуації, коли канали є близьким за фазою, але приводить до різкого його погіршення, якщо кодуються сигнали з великим взаємним фазовим зсувом.

Joint Stereo (MS/IS Stereo) вводить ще один метод спрощення стереосигналу, що підвищує якість кодування на особливо низьких бітрейтах. У цьому випадку для деяких частотних субканалів відкидається інформація про фази, тобто залишається тільки співвідношення потужностей між лівим та правим каналами.

3.4. Засоби виведення зображення.

3.4.1. Загальні вимоги до засобів виведення зображення

Для передачі інформації від комп'ютера до людини зір людини є найзручнішим каналом (п. 2.1.1) завдяки його високій пропускній спроможності (кількості введеної інформації за одиницю часу). Найпершим в комп'ютерних технологіях реалізованим для цього способом був друк літер на папері за допомогою керованої комп'ютером електричної друкарської машинки або телетайпа. Хоча ефективність такої взаємодії комп'ютера з людиною не дуже висока, така технологія фактично визначила один з двох типів виведення графічної інформації – ***друк на носії***, який досить довго зберігає

виведену інформацію. Більш динамічний обмін був вперше реалізований на основі використання телевізійного монітору, пристосованого для відтворення сторінки літер. Надалі на основі цього типу засобу відтворення був реалізований графічний монітор. Загалом ***монітор*** започаткував засіб для відтворення інформації, що динамічно змінюється у часі. Сучасні засоби цифрової обробки інформації активно використовують обидва класи виводу візуальної інформації.

Для динамічного виведення виходячи з основних потреб для якісного сприйняття людиною можна визначити наступні вимоги:

- висока контрастність (найкритичніший параметр легкого зчитування інформації оком);
- висока яскравість (особливо для роботи в умовах сильної фонової освітленості);
- якісна кольоропередача (особливо для потреб професійних дизайнерів та інших професій, пов'язаних з обробкою зображень);
- мала інерційність (для мінімізації викривлень динамічного зображення);
- комфортність перегляду зображення (великі кути перегляду, відсутність мерехтіння, можливість зручного розташування тощо);
- мінімально можливе енергоспоживання (особливо критично для мобільних пристроїв, які повинні якомога довше працювати від однієї зарядки акумулятора).

Для статичного виведення основні вимоги дещо інші:

- висока контрастність (як і в попередньому випадку);
- якісна кольоропередача, до якої додається наступний пункт –
- стійкість відбитку (особливо при наявності ультрафіолетового випромінювання, проблемного для багатьох барвників;
- ціна відбитку;
- комфортність (простота користування та мінімум потреб технічного обслуговування засобу друку);
- енергозбереження (в даному випадку найменш критичний з показників за виключенням одного специфічного способу статичного виведення – *e-ink,* п. 3.4.2*)*.

Відсутність вимоги до яскравості в даному випадку визначається тим, що статичне зображення спостерігається при наявності зовнішнього джерела освітлення, яке і визначає яскравість.

Детально розглянемо популярні технології, які використовуються для візуалізації в цифрових засобах та перспективи їх розвитку. Усі ці технології значно сильніше зв'язані з сучасними підходами в інформатиці, ніж цього, наприклад, вимагає виведення звуку. При цьому враховується особливість сприйняття зображення людиною і виконується значна адаптація даних для того, щоб інформація зображення сприймалась якомога повно та правильно.

3.4.2. Монітори

Першими розглянемо більш критичні для комп'ютерних технологій засоби динамічного виведення, до яким можна віднести:

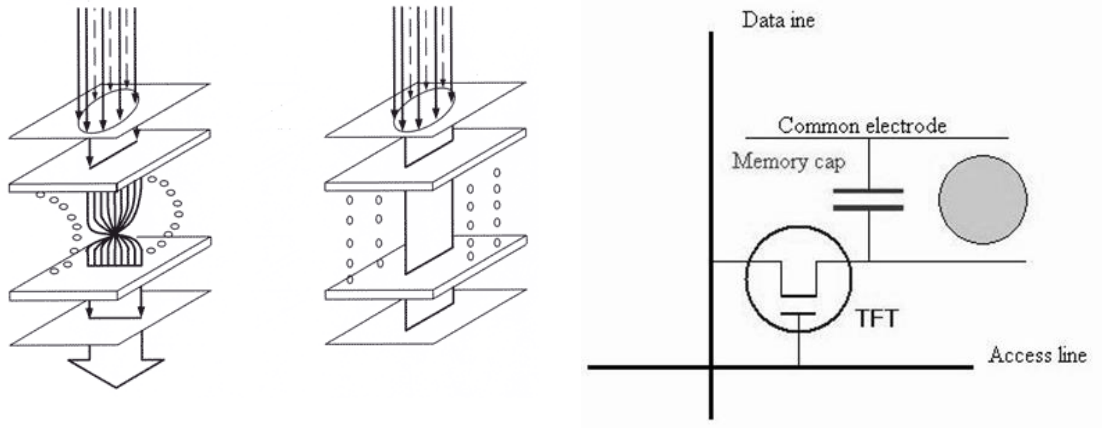
- монітори або екрани (засоби для формування зображення цифровими методами на робочій площині самого пристрою);
- проектори (засоби для створення методом оптичної проекції зображення на зовнішній поверхні, тобто проекційному екрані, або предметі, що його заміняє).

З врахуванням того, що використання кінескопу на основі електроннопроменевої трубки (англійською мовою CRT, cathode ray tube) фактично пішло у минуле, можна сказати, що сучасний монітор довільного типу є двовимірною матрицею пристроїв виведення пікселя. При цьому кожен піксель складається з трьох комірок для виведення субпікселя одного з базових кольорів, що як в засобах введення та запису зображень використовує особливості сприйняття кольору оком.

Найбільш поширеною зараз технологією реалізації монітору є рідкокристалічна. Такі матриці зазвичай називають LCD від англійського liquid crystal device або TFT від thin film transistor, тобто тонкоплівковий транзистор, призначення якого в подібних пристроях розглянемо незабаром.

Ідея рідкокристалічної матриці полягає у можливості створення електрокерованого затвору для світла на основі сандвіча з шару рідкого кристалу та двох поляризаційних плівок (поляроїдів). Рідкий кристал являє собою особливу фазу деяких речовин, що об'єднує властивості рідин (текучість) та кристалів (анізотропія та впорядкованість). Для комп'ютерних технологій найважливішими властивостями холистеричних кристалів є переорієнтація кристалів в електричному полі та обертання площини поляризації світла, що проходить через шар такої речовини.

Таким чином субпіксельні комірки (рис. 3.21) можна реалізувати помістивши шар рідкого кристалу між двома скляними пластинами, на поверхні яких нанесені надзвичайно тонкі і за рахунок цього прозорі для світла електроди, у вигляді набору паралельних ліній рядків (з одного боку рідкого кристалу) і стовпчиків (з іншого боку). Вибором електроду рядка можна подати керуючу напругу на усі комірки рядка (через усю сукупність електродів колонок), що дозволяє у кожній з комірок плавним чином міняти кут обертання площин поляризації. Оскільки око не сприймає різницю у поляризації, потрібен перетворювач поляризація/яскравість, роль якого виконують два шари поляроїдної плівки. Оскільки описані комірки є тільки затвором для світла, необхідне його джерело. Для цього використовуються або декілька люмінесцентних ламп з холодним катодом (застаріла технологія), або «білі» світлодіоди, які зараз отримують надалі ширше застосування практично в усіх системі підсвітки. Рівномірність підсвітки забезпечується спеціальною конфігурацією світловода-розсіювача позаду матриці.



а б

Рис. 3.21. Принцип дії субпіксельної комірки LCD матриці (а) та активна матриця TFT (б).

Колір кожного субпікселя (червоний, зелений або синій) забезпечує смугова фільтрація за рахунок використання світлофільтра, як і у випадку реалізації світлочутливої матриці, п. 3.4.4.

Описана конфігурація формування матриці монітора є цілком працездатною і використовувалась на початковому етапі розвитку подібних технологій, але вона має суттєвий недолік – недостатню контрастність. Це пов’язано з тим, що напруга керування на комірки даного рядка подається тільки під час звернення саме до цього рядка,

тобто використання електроду вибору цього рядка. Протягом іншого проміжку часу, коли виконується керування іншими рядками (скважність визначається кількістю рядків матриці) відбувається релаксація рідких кристалів до вільного стану. Таким чином середній (в часі) кут обертання площини поляризації значно менший за бажаний, що і відповідає малій контрастності. І саме транзисторний ключ, інтегральне виготовлення якого за технологією тонких плівок дало назву сучасній модернізованій матриці (TFT), дозволив зробити майже незмінною керуючу напругу комірки в той час, коли до неї немає звернення Деяка незначна зміна визначається саморозрядом за рахунок скінченого опору закритого ключа, але для польового транзистора цей опір є надзвичайно великим.

Історія розвитку технологій рідкокристалічних моніторів зводиться до боротьби з основними його недоліками – інерційністю та значним погіршенням якості зображення (контрастності та правильності відображення кольору) при перегляді під великими кутами. Щодо інерційності, прийнятні параметри були досягнуті досить швидко, проблеми з кутами виявились складнішими. На поточний момент є три основні конструкції рідкокристалічних матриць:

- TN+Film (або для скорочення – TN, twisted nematic);
- MVA (Multi-Domain Vertical Alignment) або PVA (те ж саме);
- IPS (In-Plane Switching).

TN матриці в доповнення до описаного раніше мають спеціальний плівковий шар для покращення кутів огляду. За інерційністю ці матриці є найшвидшими.

MVA матриці мають субпіксель, що складається з двох половинок, протилежно орієнтованих одна до одної. Відповідно при відхиленні від перпендикулярного напрямку перегляду в один чи інший бік призводить до того, що одна з цих половинок зменшує прозорість при іншому куті, а друга збільшує, що разом забезпечує приблизно стале значення.

IPS матриця має обидва керуючих електроди з одного боку шару рідкого кристалу, що і відображено у назві («в площині»). Це значно покращує кутові характеристики зміни орієнтації кристалів та, відповідно, кути перегляду, які для цього типу матриці є найкращими в порівнянні з конкурентами.

Загалом рідкокристалічні матриці мають такі переваги:

- малий розмір пікселя, що дозволило реалізувати екрани, на яких зернистість неозброєним оком практично непомітна;
- хороша кольоропередача та кути огляду (IPS, MVA);

- прийнятна енергетична економічність при умові світлодіодної підсвітки;
- відсутність мерехтіння матриці (при цьому може спостерігатись мерехтіння підсвітки при зменшенні яскравості від максимального значення, оскільки для цього використовується широтно-імпульсна модуляція, п. 4.1.4, але з частотою, що значно перевищує максимальну частоту, яку реєструє око);
- великий час можливої експлуатації (десятки тисяч годин, що визначається навіть не матрицею, а деградацією джерела підсвітки).

Останні десять років певних успіхів досягли у реалізації ще однієї технології, якій пророкує велике майбутнє – активній матриці органічних світловодів (англійською мовою – Active Matrix of Organic Light-Emitting Diode, AMOLED). Ідея її створення полягає у використанні «випромінювальної» технології замість «поглинальної», яку використовують рідкокристалічні матриці. Світлодіод випромінює саме стільки світла, скільки є потреба для відтворення поточного стану пікселя, що забезпечує більшу економічність. Крім того легко реалізуються дуже великі кути перегляду. Але для використання матриці світлодіодів треба було вирішити дві принципові проблеми – створення надійного синього світлодіоду та матрицювання діодів різних кольорів. Це вдалось реалізувати і на окремих діодах (така технологія використовується для створення великих інформаційних екранів), більш того існують серійні RGB діодні збірки. Але достатньо компактну для індивідуального використання і доступну за ціною систему відтворення зображення змогли реалізувати тільки на основі певних органічних сполук з напівпровідниковими властивостями. Для керування такою матрицею, як і у випадку рідкокристалічної матриці використовуються тонкоплівкові транзистори (по одному на субпіксель) та аналогічна адресація рядків та колонок. Як результат отримали пристрій з можливістю виготовлення субпікселів дуже малих розмірів, економічніший та легший за конкурентів. Така матриця є надзвичайно тонкою і, навіть, гнучкою, з надмалою інерційністю і дуже якісною передачею кольору. Основним недоліком такої технології, який ще не усунуто повністю, є досить швидка деградація, що призводить до зміни кольоропередачі і суттєвого скорочення часу життя приладу.

На поточний момент AMOLED матриці вже охоплюють досить широкий діапазон засобів. Першими з них були екрани смартфонів та електронні відеошукачі фотоапаратів, тобто пристрої з відносно невеликим часом світіння, хоча вже з’явились і серійні телевізори.

Є ще одна цікава технологія реалізації монітору, яка займає проміжне положення між динамічним та статичним відображенням – так звані «електронні чорнила» (*e-ink, e-paper*). Ця технологія використовує електричне перенесення (електрофорез) частинок барвника в мікрокапсулі, розмір якої менший за розмір пікселя. В залежності від полярності керуючого сигналу чорний барвник в окремій капсулі збирається в передній (видимій), або задній її частині. З певної кількості окремих капсул формуються пікселі і створюється зображення. Такий пристрій має надзвичайно мале енергоспоживання (практично відсутнє між моментами перемикання), що є дуже привабливим для компактних акумуляторних пристроїв з не дуже частим перемиканням зображення. До таких пристроїв відносяться, наприклад, так звані електронні книжки. Недоліками електронних чорнил є мала контрастність, погана передача напівтонів та кольору, дуже велика інерційність, яка взагалі є неприйнятною для універсального монітора.

3.4.3. Проектори

Реалізація проекторів також має за основу матрицю для формування зображення. Відмінністю їх від моніторів є оптичне проеціювння цього зображення на екран за допомогою спеціального високоякісного об’єктива з метою забезпечення його великого розміру. Оскільки оптичне проеціювання забезпечує багаторазове збільшення зображення, відбувається аналогічне послаблення його яскравості (за рахунок збереження кількості випроміненої енергії). Таким чином зображення на матриці проектора повинне мати надзвичайно велику яскравість, що вимагає спеціальних проекційних джерел світла та застосування засобів відведення тепла від пристрою формування зображення.

Є дві основні технології, які на поточний момент використовуються для створення проекторв:

- 3LCD (рис. 3.22);
- DLP (рис. 3.23).

3LCD проектор є певним перенесенням вже розглянутої технології рідкокристалічної матриці на випадок проекційного пристрою, але є суттєві відмінності. В даному випадку є вигідним замість реалізації різнокольорових субпікселів викоритати три окремі матриці для формування повного кадру у кожному з кольорів адитивної тріади. Це забезпечує розподіл енергії нагріву матриць по більшій площі, що спрощує охолодження, конче необхідне через те, що значна частина потужного світлового потоку поглинається

оптичними затворами – піксельними комірками. Крім того виділення кожного з кольорових потоків світла з широкосмугового («білого») від проекційного джерела можна реалізувати не поглинанням зайвого смуговими кольоровими світлофільтрами, а «кольороділенням» на основі резонансного відбиття від діхроічного дзеркала (багатошарової тонкоплівкової структури).

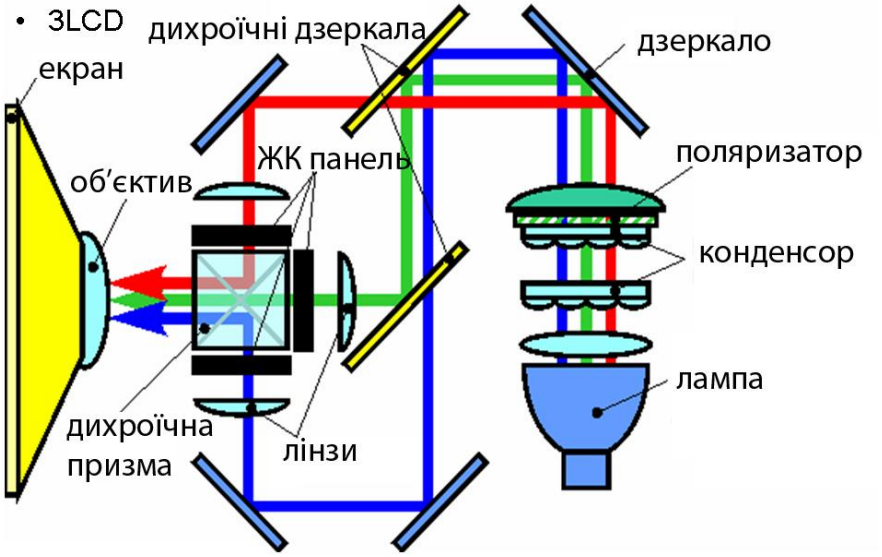
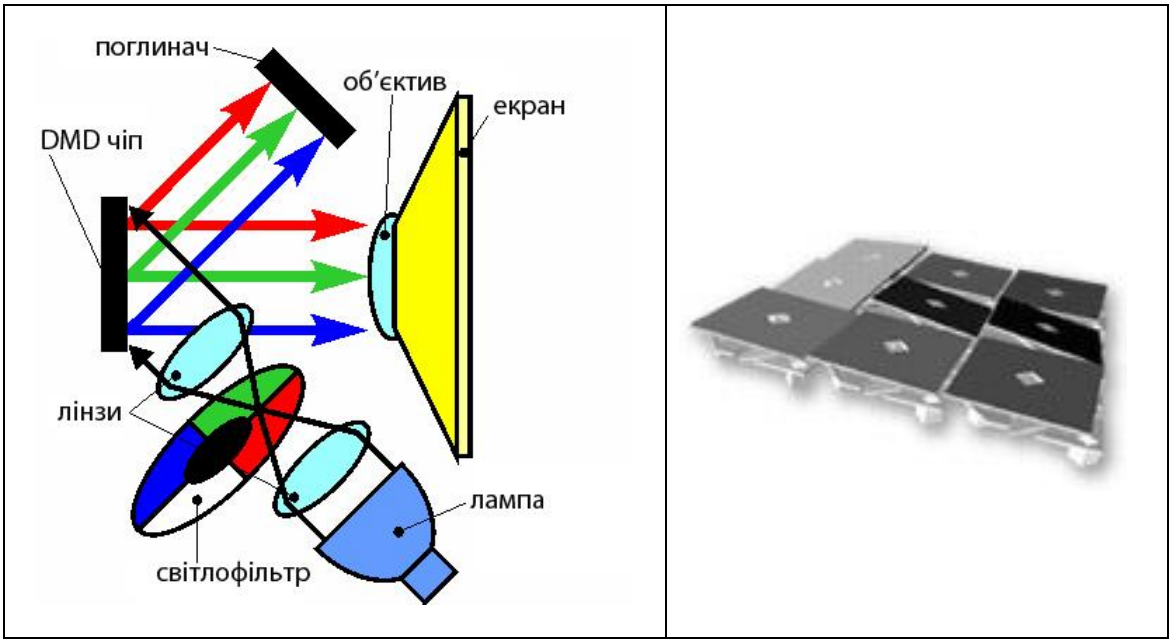


Рис. 3.22. Принцип дії 3LCD проектора



а б

Рис. 3.23. Принцип дії DLP проектора (а) та структура матриці (б)

Такий спосіб формування кольорових пучків має суттєво менші втрати в порівнянні з поглинанням. Суміщення трьох різнокольорових

кадрів у цілісне кольорове зображення забезпечується діхроічною призмою з такими ж тонкоплівковими шарами. Суміщення трьох зображень призводить до того, що піксель на екрані формується не у вигляді трьох субпіксельних смужок основних кольорів, як це відбувається на моніторі, а як квадратик, однорідно заповнений необхідним складним кольором.

DLP (Digital Light Processing) є принципово іншою технологію створення зображення для проекції. Кадр на проекційному екрані в даному випадку також створюється суміщенням субкадрів кожного з трьох основних кольорів. Для формування субкадру використовується спеціальна матриця мікродзеркал, рис. 3.23.б (по одному дзеркалу на піксель). За допомогою окремого електричного сигналу, що подається до кожного мікродзеркала, забезпечується зміна кута його нахилу, що направляє віддзеркалений промінь, що повинен формувати піксель, або в об'єктив, тобто на екран, або на спеціальний поглинач збоку від об'єктива. Оскільки керування яскравістю пікселя таким чином можлива реалізація виключно у ключовому (бінарному) режимі, для формування напівтонів використовується широтно-імпульсна модуляція (п. 4.1.4). Для формування субкадрів необхідного кольору використовується обертальний диск-світлофільтр з секторами відповідних кольорів. Таким чином по черзі формуються субкадри кожного з кольорів, які за повний оберт диску накладаються на проекційному екрані. У більш дорогих пристроях використовують три матриці субкадрів. Крім збільшення світлового потоку це забезпечує відсутність основного недоліку описаної одоматричної конструкції – райдужного ефекту, тобто сприйняття глядачем артефакту, що збільшує втомлюваність очей – райдужної кайми на контрастних краях при рухах голови.

DLP проектор забезпечує більшу контрастність зображення, менш помітну сітчасту рамку між піселями, більш насичені кольори. Загалом вважається, що ця технологія має більші перспективи для розвитку.

Окремо слід зупинитись на проекційному джерелі світла. На поточний момент основним джерелом є ртутна лампа високого тиску, яка забезпечує достатньо потужний потік світла прийнятного спектру та досить великий час життя. Деградація цієї лампи визначається втратою потоку (вдвічі за 4-5 тис. годин). Зауважимо, що лампа замінюється разом з оптичним модулем формування пучка і коштує приблизно половину ціни усього проектора. Останній час для компактних пристроїв отримує розповсюдження використання світлодіодних освітлювачів, які мають значно більший коефіцієнт

корисної дії, але поки-що програють у світловому потоці приблизно на порядок.

Загалом проекційні комп'ютерні технологій використовуються не тільки як один з способів виведення зображень з комп'ютера для великого приміщення, вони дозволили перевести на рейки цифрових технологій кінематограф, що є особливо важливим у зв'язку з поширенням у цій галузі складних методів цифрової обробки зображень і, навіть, цифрового введення та комп'ютерного синтезу.

3.4.4. Друк зображення

Тепер розглянемо особливості формування статичного зображення, тобто друк на деякому носії, частіше за все – папері.

Забезпечення технології друку передбачає:

- технологію формування мікроплями фарби на поверхні паперу (або у його приповерхневому шарі);
- алгоритм передачі напівтонів;
- спосіб передачі кольорів;
- спосіб формування растру усього зображення.

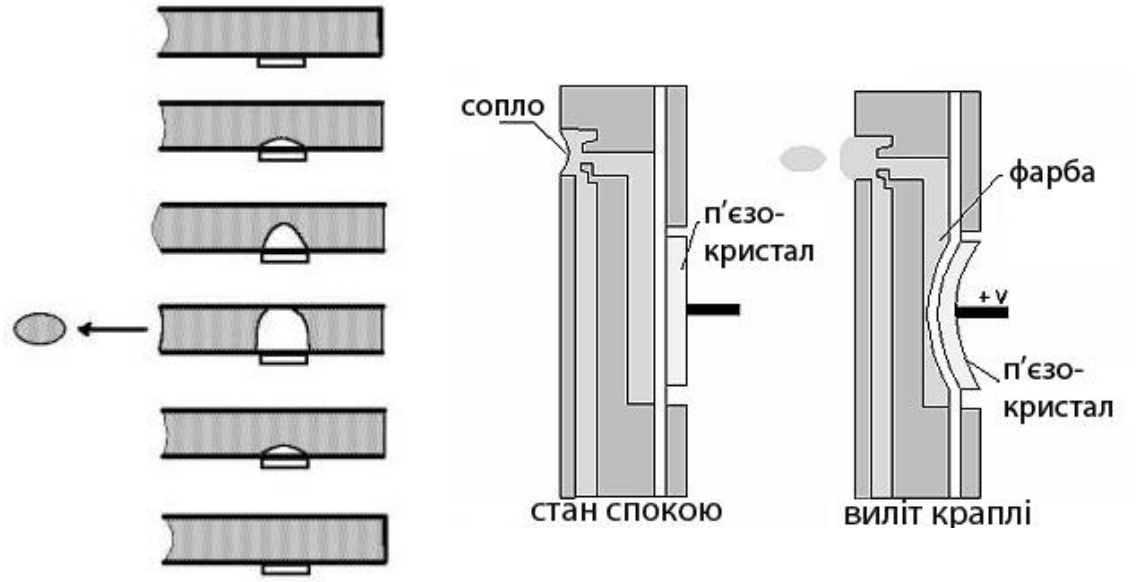
Нанесення фарби на папір комп'ютерні технології значною мірою запозичили у раніше розроблених аналогів. На поточний момент найбільш поширеними є дві:

- добре відоме використання чорнил (та туші);
- електроперенос дрібних частинок твердого тонера, розроблений свого часу як основа ксерографії (ще аналогової).

Чорнила є водяним розчином фарби (барвника) у воді. Туш на відміну від чорнила є дрібними твердими частинками, звішеними у рідині, яка їх не розчиняє. При нанесенні на папір подібна рідка фарба всмоктується у її приповерхневий шар або залишається практично повністю на поверхні паперу (більш характерно для використання туші) з подальшим висиханням рідини. Пристрій друку, тобто принтер, який спирається на цю технологію, повинен забезпечити формування надзвичайно малої краплі чорнил. Розмір краплі разом з використанням паперу певної якості (який менше чи більше всмоктує чорнила, тобто призводить до певного розпливання краплі) буде визначати мінімальний розмір отриманої мікроплями, що і обмежує роздільну здатність друку.

Для формування краплі використовується:

- короткочасний нагрів капіляра з чорнилом, що призводить до вибухоподібного випаровування і появи у ньому мікробульбашки (рис. 3.24.а);
- електричний імпульс, що подається на п'єзоелектричний елемент, який є стінкою капіляру і працює як мікроскопічний поршень(рис. 3.24.б).



а б

Рис. 3.24. Бульбашковий (а) та п'єзоелектричний (б) чорнильний друк

Растеризацію зображення у чорнильних принтерах (їх зазвичай називають струменевими, іноді бульбашковими, що підкреслює належність саме до бульбашкової технології) виконує пересування вздовж рядка друкуючої голівки та прокруткою паперу впоперек рядка. Голівка має певну кількість сопел, розташованих впоперек рядка (за рахунок цього одразу друкується смужка замість окремої лінії).

Електроперенос тонера використовують лазерні та світлодіодні принтери (рис. 3.25). Основним елементом формування зображення є фотобарабан, тобто алюмінієвий барабан з напівпровідниковим шаром з малою власною провідністю. При обертанні барабану його поверхня рівномірно заряджається за допомогою ролика перенесення заряду. Саме на цій однорідно зарядженій поверхні можна за допомогою променя лазера формувати зображення, яке потім буде перенесене на поверхню паперу.

Рис. 3.25. Принцип дії лазерного принтера

При освітленні деякої ділянки променем світла за рахунок явища фотопровідності поверхня барабану у цьому місці втрачає опір, і заряд з неї стікає через металевий барабан. Промінь лазера за рахунок дії спеціальної оптико-механічної системи відхилення (на основі обертальної призми) сканує рядок вздовж фото барабану. Вмикання та вимикання променя створюють потрібну для даного рядка структуру розряджених ділянок, а обертання самого барабана забезпечує послідовне формування усього зображення рядок за рядком. Світлодіодний принтер відрізняється тим, що замість відхилення променя лазера використовується лінійка світлодіодів, яка за допомогою об’єктиву одночасно формує весь рядок.

Далі створене електричне зображення на поверхні фотобарабану за рахунок вибіркового налипання порошкоподібного тонера стає майже повноцінним. Його залишається перенести електричним полем на папір, та закріпити на папері плавленням за допомогою спеціального нагрівача. При цьому барабан зробив майже оберт. Для завершення повного циклу залишається видалити з поверхні барабану в окремий бункер дрібні залишки тонера спеціальним лезом – ракелем.

Описані принципи друку є придатними для зображення без передачі напівтонів, оскільки після друку на поверхні паперу залишаються або ділянки чистого паперу, або повністю зафарбовані ділянки. Це підходить, наприклад для друку літер, але не годиться для відтворення якісних напівтонових зображень. Таким чином доводиться доповнювати апаратну технологією деякою програмною з метою формування напівтонів.

За передачу напівтонів в лазерних та струменевих принтерах відповідає розрахунок фактично іншого зображення, у якому кожному

«сірому» пікслелю вхідного зображення відповідає матриця, ступінь заповнення якої визначає рівень (градацію) сірого. Саме точки (пікселі) обробленого зображення відповідають мікроточкам принтера.

Зауважимо, що результат відтворення зображення на папері певною мірою залежить від алгоритму заповнення матриці, тобто яким чином розподіляються зафарбовані точки по поверхні матриці. Як приклад, може застосовуватись:

- випадковий розподіл точок (рис. 3.26.б);
- максимально рівномірний розподіл (рис. 3.26.в);
- нарощування приблизно круглої центральної плями (рис. 3.26.г).

Ця матриця, призначена для передачі півтонів на дврівнемому пристрої, яким є принтер, називається матрицею халфтонингу (halftoning) або дизерингу (dithering).

а б в г

Рис. 3.26. Халфтонинг (дизеринг) зображення у градаціях сірого: а – оригінал, б – випадкове заповнення, в – впорядковане заповнення, г – варіювання розміру кружечка.

Друк кольорового зображення, в тому числі формування напівтонів є аналогічним до друку градацій сірого з тою різницею, що замість одного чорного кольору фарби (чорнила або тонера) використовується декілька різних. Створення кольору вже було розглянуто при описі матриць моніторів. У монітора кольоровий піксель формувався додаванням сигналів трьох основних кольорів до початкового умовно чорного, тобто деякого рівня загального сигналу, який за рахунок обмеження динамічного діапазону ока в результаті його адаптації до середнього рівня усього зображення буде сприйматись як чорний. У цьому випадку використовуються кольори, які відповідають максимумам чутливості відповідних клітин сітківки

ока. Така тріада кольорів (RGB – red, green, blue) називається адитивною. У випадку друку початковий рівень навпаки є умовно білим (перевищення динамічного діапазону ока, адаптованого до зображення) від якого віднімаються базові кольори іншої тріади – субтрактивної (CMY – cyan, magenta, yellow). Таким чином для друку кольорового зображення виконується перерахунок кольорів з адитивної тріади в субтрактивну, а для передачі напівтонів кольорових каналів розраховується заповнення матриця халфтонінгу кожного з кольорових каналів усіх пікселів зображення.

При реалізації друку є ще одна дуже принципова проблема – правильність передачі кольорів при накладанні фарби одна на одну. Найбільш складною в цьому випадку є проблема створення якісного чорного тільки за рахунок змішування кольорових фарб. Для її вирішення до трійки кольорових фарб додали четверту чорного кольору і замість розрахунку тріади CMY розраховують четвірку CMYK (від blacK). Відповідно у чорний канал (точніше «сірий», оскільки забезпечуються напівтони) виносять деякий наперед визначений відсоток частини сигналів з інших кольорових каналів, що є в усіх каналах одночасно. Тобто у «сірий» канал виділяють певну частину загальної «сірої» складової пікселя.

Зауважимо, що адитивна та субтрактивна тріади є формально рівноцінними, але за рахунок обмеження діапазону беззнакових цілих чисел (0–255), якими кодуються канали кольорів в обох випадках, перерахунок з однією тріади в іншу є втратною операцією за рахунок потреби заміни значень, що виходять за межі допустимого діапазону значень, на граничні.

Для забезпечення достатньо широкого динамічного діапазону (достатньо великої кількості градацій) у кольорових принтерах (зазвичай струменевих) у пурпурному та блакитному каналах використовуються пари чорнил одного кольору але різної насиченості. Це відповідає найбільш розповсюдженому для забезпечення фотографічної якості 6-колірному друку. Іноді використовуються ще більш складні комбінації кольорів. Відповідно для такого високоякісного друку виконується більш складній розрахунок значень по каналах.

3.5. Питання для самоконтролю

1. Чому матриця Баєра має вдвічі більше зелених пікселів, ніж червоних та синіх?
2. Звідки для кожного з пікселів при введенні зображення отримуються відсутні кольори?

3. Чи призводить конфігурація Баєра для введення кольору до втрати роздільної здатності?
4. Чим забезпечується конкурентоспроможна чутливість CMOS датчика сучасних матриць в порівнянні з CCD ?
5. Чому у CCD піксельному датчику відсутній конденсатор для накопичення заряду?
6. Які переваги забезпечила цифрова фотографія у порівнянні з плівковою?
7. Чи можна забезпечити підкреслення країв на зображенні точковою обробкою?
8. Чому при комп'ютерній реалізації точкових перетворень зображення зазвичай використовують не пряме, а обернене перетворення?
9. Що є просторовою гармонікою та просторовою частотою?
10. Чи є згорточка фільтрація повним аналогом фільтрації просторових частот?
11. Чим відрізняється ядро згортки для підкреслення країв від ядра для виділення країв?
12. Чи може в результаті розв'язання задачі сегментації піксель бути віднесений до двох сегментів?
13. Чому використання кореляційної функції для розпізнавання образів є неефективним?
14. Чому розпізнавання рукописних літер є складнішим за розпізнавання друкованих?
15. Яким чином комп'ютерний зір використовується в автотранспорті?
16. Чи є сенс використовувати розпізнавання обличчя для автентифікації?
17. Чому TFT монітор забезпечує гірше відтворення чорного кольору в порівнянні з AMOLED ?
18. Для чого потрібен транзистор у піксельному елементі TFT монітору?
19. Для яких зі складових проектора найнебезпечніший вихід з ладу вентилятора ?
20. Для чого в DLP проекторі використовується широтно-імпульсна модуляція ?
21. Чому в 3LCD проекторі відсутній ефект райдуги?
22. Чи можна реалізувати принтер з набором кольорів RGB ?
23. Чи може словниковий метод стиснення використовуватись для стиснення файлу виконуваної програми?
24. Чи враховується в алгоритмі JPEG корельованість окремих блоків між собою?

4. Автоматизація вимірювань

4.1. Основи автоматизації експерименту

4.1.1. Потреби автоматизації

Розробка певних інженерних проектів вимагає дослідження працездатності попередніх реалізацій, яке значною мірою пов'язане з вимірюваннями фізичних величин. При забезпеченні запуску у виробництво вже готового проекту та при експлуатації виробів контрольні вимірювання також є обов'язковими. Десятиріччями подібні вимірювання робились із значною кількістю ручних операцій, як при отриманні даних, так і при їх обробці. Але поява цифрових засобів обробки інформації суттєво змінили підхід для розв'язання зазначеної задачі. Відповідно наступна глава буде присвячена використанням комп'ютерів та мікроконтролерів для реалізації вимірювальних засобів та автоматизації досліджень.

При проведенні будь якого експерименту необхідно чітко розуміти його алгоритмізацію, тобто покрокову послідовність дій, які забезпечують отримання результату з необхідною точністю. Алгоритм експерименту звичайно передбачає коректне використання методів та методик вимірювання та обладнання, що відповідають цим методикам. При традиційному (ручному) керуванні ходом дослідження керує експериментатор, саме він дотримується алгоритму. Але в цьому випадку можливі певні відхилення від жорсткої послідовності дій, наявність галужень у алгоритмі і навіть його корекція. Ці особливості значною мірою нівелюють сприйняття дослідження як алгоритму, особливо при використання відносно простих методик. У випадку автоматичного дослідження його коректність значно більше залежить від правильності алгоритмізації.

Автоматизація (не обов'язково цифрова) в експериментальних дослідженнях наукового та інженерного призначення сприяє підвищенню точності вимірювань, збільшенню їх швидкості і надійності, звільняє експериментатора від рутинних операцій. Головним причинами необхідності автоматизації експерименту в наукових дослідженнях та інженерній роботі (як при проектуванні, так і при експлуатації) є:

- спостереження надшвидких (людина не встигає) і край повільних процесів (рутинні дії з великими інтервалами провокують помилки);
- необхідність застосування саморегульованих систем (автомат часто керує краще за людину);
- повторення великої кількості однотипних спостережень (знову рутина);
- одночасна робота з великою кількістю параметрів (можна не встигнути);
- необхідність позбутись суб'єктивності спостерігача (на жаль є психологічний фактор ненавмисних поправок);
- неможливість перебування дослідника поруч з досліджуваним об'єктом (віддаленість об'єкту або неприйнятність або шкідливість для людинри умов, у яких перебуває об'єкт).

Використання комп'ютерів та мікроконтролерів для автоматизації досліджень дозволяє реалізувати досконалішу автоматичну систему, накопичувати дані в більш зручній для подальшої обробки формі.

Навіть при простих вимірюваннях певної величини виникає ряд задач, які є бажаним передати від людини до автомату. При цьому не зважаючи на видиму простоту цих задач якісну їх реалізацію без застосування цифрових засобів виконати складно, або, навіть, неможливо. До таких задач можна віднести:

- автоматичний вибір діапазону вимірювання (навіть у нескладному мультиметрі);
- автокалібрування (періодична процедура калібрування передбачається для прецизійних приладів і чим частіше вона виконується, тим краще);
- врахування нелінійності і неоднорідності передаточних і частотних характеристик (якщо характеристика виміряна та табульована, нескладно зробити перерахунок);
- застосування більш складних методів та алгоритмів вимірювання (які при відсутності автоматики вимагають великої кількості послідовних дій від експериментатора).

Використання засобів цифрової автоматизації експерименту є дуже корисними при отриманні величин, які є результатом певної обробки даних прямого вимірювання. В цьому випадку обробка даних може вбудовуватись разом з їхнім отриманням і розгалуження алгоритму керування експериментом може виконуватись з використанням розрахованих даних.

Загалом алгоритмізація автоматичних досліджень є схожою на роботу з неавтоматичними приладами. Хоча слід зазначити, що саме використання автоматики (особливо цифрової) часто дозволяє використати більш складну методику дослідження, якою не в змозі в реальному часі керувати людина. Це часто забезпечує більшу надійність та точність дослідження.

При розробці автоматизованої вимірювальної системи доводиться розв'язувати і певні додаткові задачі. Оскільки сам експеримент програмується фактично до його проведення, експериментатор (точніше програміст) більш чітко і детально повинен розуміти послідовність дій під час досліду, вміти на рівні написання керуючої програми передбачати і виконувати такі традиційні операції, як перемикання з'єднань, підбір масштабування вимірюваного сигналу, тощо.

При розробці програми, яка керує експериментом, як вже було зазначено, строгість алгоритму має значно більше значення, ніж при керуванні людиною. Крім того ідеологія економного і гнучкого формування автоматичної вимірювальної системи передбачає відмову від використання «повноцінних» приладів, які включають такі зайві для автоматичного режиму роботи елементи, як ручні органи керування, дисплей, самостійний блок живлення, тощо. Замість цього використовуються модульні системи вимірювання. Відповідно установка утворюється з окремих блоків, кожен з яких виконує відносно вузьку функцію. До сукупності таких модулів додається один, чи декілька контролерів, роль яких часто відіграє комп'ютер або мікроконтролер. Ці контролери повністю беруть на себе логіку взаємодії окремих модулів. Для зручності модулі об'єднуються на спеціальній рейці, або у каркасі. До них підводяться стандартизовані інформаційні магістралі та живлення. Все це дозволяє розробляти достатньо потужні та зручні вимірювальні системи.

Програма для керування експериментом за структурою нагадує програму для математичних обчислень, тобто послідовний ланцюжок команд з розгалуженнями та циклами. Але крім традиційних команд алгоритмічної мови в даному випадку додаються ще команди-звернення з програми до зовнішніх пристроїв (модулів вимірювальної системи). Часто є потреба коректного врахування зворотних зв'язків та потребу у необхідних місцях програмного коду часових затримок.

4.1.2. Вимірювальні операції

Загалом за реалізацію та коректне використання засобів вимірювання відповідає метрологія. Одразу зауважимо, що специфіка

автоматичного обладнання, особливо з використанням програмного керування, вимагає певної відмінності від деяких трактовок класичної метрології.

Відповідно до основних визначень метрології відносяться:

- ***метод – сукупність прийомів використання засобів вимірювання для визначення числового значення величини*** (абстракція, в якій не визначається природа фізичної величини);
- ***методика вимірювання – детально намічений порядок процесу вимірювань, який регламентує засоби, алгоритми, які забезпечують вимірювання з заданою точністю*** (на цьому рівні виконується конкретизація фізичної природи величини, що вимірюється);
- ***алгоритм – покроковий опис дій, необхідних для одержання результату*** (в класичній метрології опис дій для експериментатора, тобто людини).

Загалом, ***вимірювання – процес визначення числового значення вимірюваної величини, дослідним шляхом на основі порівняння її з одиницею вимірювання за допомогою засобів вимірювальної техніки.*** В цьому визначенні фігурує порівняння, що диктує потребу того, щоб досліджувана величина і величина, яку формує пристрій, що задає одиницю вимірювання, були однаковими за фізичною природою. Відповідно до такого підходу визначається сукупність елементарних метрологічних операцій (на ці елементарні дії розкладається уся процедура вимірювання):

- вимірювальне перетворення;
- масштабування;
- відтворення величини заданого розміру;
- порівняння.

Вимірювальне перетворення – процес перетворення вхідного сигналу у вихідний, інформативний параметр якого з заданою точністю функціонально зв'язаний з інформативним параметром першого. Використовується для уніфікації типу сигналів та перетворення до фізичної природи, яка є зручною для вимірювання або подальшої обробки;

Масштабування – утворення сигналу, інформаційна характеристика якого зв'язана з аналогічною вхідного сигналу пропорційним законом. Пристрій, що забезпечує масштабування зветься масштабним перетворювачем. Його роль відіграють дільник та підсилювач. Масштабування використовується як уніфікація рівня сигналів, узгодження динамічного діапазону сигналу з динамічним

діапазоном подальшої частини вимірювальної системи, або частини приладу.

Відтворення величини заданого розміру – утворення сигналу з наперед заданим розміром інформаційного параметру. Засіб, що служить для відтворення заданої величини зветься мірою. Найбільш точно можна відтворювати напругу, струм, частоту, час, довжину, масу. Формально цю операцію можна представити як перетворення коду у фізичну величину заданого типу. Міра може бути одноканальною та багатоканальною, некерованою та керованою.

Порівняння – операція визначення співвідношення між розмірами однорідних величин з ціллю одержання відповіді – більше / менше. Засіб, що служить для порівняння, зветься компаратором.

До цих основних вимірювальних операцій можна додати певні допоміжні, такі як:

- ***запам'ятовування*** (аналогового або цифрового сигналу);
- ***передача***;
- ***усереднення***;
- ***комутація***.

Продовжуючи «цитування класичної метрології», можна виділити два основних методи вимірювання:

- метод безпосередньої оцінки;
- метод порівняння з мірою.

Якщо метод порівняння з мірою в межах реалізації системи як сукупності елементарних дій не викликає запитань, то метод безпосередньої оцінки визначається як метод, коли вимірювана величина «зчитується безпосередньо з шкали», і, сама шкала трактується просто як «набір деяких поділок». Таке трактування виникло з потреби припису порядку дій для людини, яка виконує вимірювання. Саме тому і було використано термін «зчитування», характерне для дій людини, не враховуючи, що ця ніби то проста дія реально включає в себе певну сукупність елементарних операцій. Крім того, з точки зору розкладення на сукупність елементарних операцій шкала є багатоканальною некерованою мірою (кожна поділка є окремою мірою), з якою виконується порівняння. Для забезпечення можливості порівняння (шкали з положенням стрілки) реалізується деякий ланцюжок вимірювальних перетворень для узгодження природи фізичних величин. Людина ж виконує саме порівняння, тобто відіграє роль багатоканального компаратора. Така трактовка є більш універсальною саме для наступної програмної (цифрової) реалізації вимірювання, коли роль людини покладається на

комп’ютер або мікроконтролер, для яких сам термін «безпосереднього зчитування» загалом є абсурдним.

4.1.3. Аналогово-цифровий перетворювач

При використанні цифрових засобів збирання та обробки даних ланцюжок елементарних операцій, розглянутих в п. 4.1.2 повинен призвести до результату, прийнятного для комп’ютера. Як було зазначено у п. 1.4, для представлення інформації у комп’ютері для подальшої її обробки необхідно її представлення у цифровій формі. Відповідно для введення до комп’ютера природних сигналів, аналогових по своїй суті (наприклад напруги, звуку, зображення, руху комп’ютерної мишки чи рухомих частин верстату, тощо), їх треба замінити на схожий цифровий – квантований дискретизований сигнал (п. 1.4.1). Саме тому прилади, які виконують ці дві операції, сукупність яких часто називають оцифровуванням, є надзвичайно важливиму для комп’ютерних технологій. Ці прилади називаються ***АЦП*** – аналогоцифровими перетворювачами (англійською мовою – Analog-to-digital converter, ***ADC***). Якщо врахувати, що вже зараз значна частина електронних приладів по суті є цифровими і їх частка у порівнянні з аналоговими постійно збільшується, для радіотехніки і електроніки є надзвичайно важливим розуміння особливостей аналого-цифрового перетворення.

Зазвичай сигнал довільної фізичної природи для зручності передачі та подальшої обробки автоматизованими засобами перетворюють в електричний. Відповідно для оцифоровувння зазвичай використовується саме сигнал електричної природи. В результаті перетворення повинен формуватись відповідний код. Перетворення не робіться миттєво, що призводить до обмеження кількості відліків (цифрових значень), які можна отримати з АЦП за одиницю часу.

Розглянемо основні параметри, які характеризують подібні прилади. Найбільш важливими з них є діапазон вимірювань та точність перетворення. Максимальне значення відповідає повному заповненню розрядів перетворювача. Мінімальний вимірюваний сигнал визначається рівнем шумів та кроком квантування і не завжди може бути коректно визначений. З іншого боку розрядність та максимальне значення, дають змогу визначити крок квантування. Відповідно як і для аналогових вимірювачів обмежуються максимальним допустимим значенням. Оскільки вхідним для таких приладів є електричний сигнал, цю величину зазвичай визначають напругою. Це пов’язано з тим що сучасні АЦП частіше за все мають

вхідну електронну схему, яка має великий вхідний опір, що спрощує його узгодження з іншими електричними пристроями та компонентами саме при вимірюванні напруги. Ця ж схема дозволяє уніфікувати діапазон допустимих сигналів, наприклад, ±5 В або ±12 В. Для потреби узгодження з іншим діапазонам сигналу може використовуватись додатковий масштабний перетворювач – підсилювач чи дільник.

Складнішою є ситуація з точністю перетворення. Основною похибкою АЦП є похибка квантування (або шум квантування), тобто відмінність квантованого сигналу від неперервного. При наявності округлення він дорівнює 1/2 кроку квантування, без округлення – цілому кроку, (як максимальне можливому відхиленню). Але слід пам'ятати, що прилад може мати додаткову похибку, співрозмірну з кроком квантування, що визначається конкретною реалізацією (вказується в описі перетворювача). Комерційні АЦП відповідно до потреб використання мають розрядність від 6 до 24 бітів (зазвичай 6, 8, 12, 16, 24).

Операція дискретизації вносить власну похибку, її можна визначити як максимальна зміна сигналу за крок дискретизації. З врахуванням того, що частота дискретизації вибирається зазвичай таким чином, що ця величина менша за крок квантування, цією похибкою можна знехтувати. Але нестабільність тактового генератору і, що більш суттєво, можливе варіювання часу перетворення від відліку до відліку, призводять до джитеру (англійською jitter - тремтіння) – фазових спотворень, тобто деяких коливань фази від відліку до відліку. Ще одною похибкою перетворення може бути паразитна нелінійність, у сучасних приладах досить мала. У деяких випадках спеціальним чином реалізують нелінійні АЦП, наприклад з логарифмічною характеристикою.

Іншою важливою характеристикою АЦП є час перетворення, оскільки саме він обмежує частоту дискретизації, а відповідно до теореми Котельникова (п. 1.4.1), є потреба забезпечити частоту дискретизації вдвічі вище за потрібну для подальшої обробки. Тобто для якомога більш якісного представлення сигналу, що вводиться до комп'ютера або до іншої системи цифрової обробки інформації, необхідно забезпечити достатньо високу для кожного конкретного випадку частоту дискретизації.

При реалізації АЦП у протиріччя вступають три параметри:

- точність;
- швидкість;
- складність.

Відповідно при розробці перетворювача доводиться шукати певний компроміс. Саме тому для різних задач використовуються різні типи АЦП, які мають відмінності у принципі роботи.

Найпростішим за структурою та принципом дії є ***перетворювачі з пилкоподібною напругою*** (рис. 4.1). Цей тип перетворювача може бути доповнений інтегруванням за періодом коливання мереженого живлення з метою придушення цієї завади. Такі модифіковані АЦП називаються ***інтегруючими***.

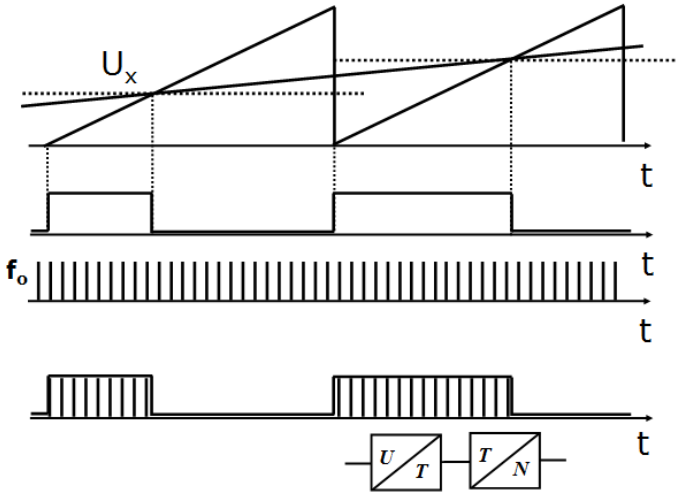


Рис. 4.1. АЦП з пилкоподібною напругою

Для такого пристрою основою є перетворення напруга/час, за рахунок формування прямокутного імпульсу, довжина якого визначається часом від початку заданої пилкоподібної напруги до сигналу від компаратора, що порівнює досліджувану напругу з миттєвим значенням пилкоподібної. Для вимірювання довжини цього прямокутного імпульсу він заповнюється прямокутними імпульсами відомої частоти з подальшим їх рахуванням. Основним недоліком цього типу перетворювача є мала швидкодія та недостатня для високої точності лінійність пилкоподібної напруги.

Останній недолік можна виправити використовуючи замість генератора пилкоподібної напруги деякий ЦАП (цифро-аналоговий перетворювач, буде розглянуто детальніше у п. 4.1.5) з подачею на його вхід цифрового коду, що інкрементується у заданому діапазоні. При цьому навіть не потрібно подальше вимірювання довжини прямокутного сигналу, оскільки значення перетворення буде дорівнювати числу на вході ЦАП при зміні сигналу від компаратора. Цей тип перетворювача має назву ***послідовного АЦП***. Але швидкодія залишиться аналогічною і визначається часом зміни пилкоподібного сигналу в межах потрібного динамічного діапазону, тобто

обмежується швидкодією елементів, з яких реалізований прилад. Наприклад для 8-бітового АЦП на основі ЦАП і компаратора час дискретизації буде дорівнювати 256 тактам.

Основним фактором обмеження швидкодії у такому типі АЦП є те, що порівняння починається фактично з молодшого розряду. Використання більш складної логіки керування для реалізації алгоритму ділення відрізку навпіл (аналог одного з популярних чисельних методів уточнення пошуку кореня рівняння) дозволяє виконати порівняння починаючи з найбільшого бінарного розряду, що значно скорочує час перетворення. Наприклад, для 8-бітового перетворення з 256 до 8 тактів. Такий тип перетврювача отримав назву ***АЦП порозрядного врівноваження*** (рис. 4.2).

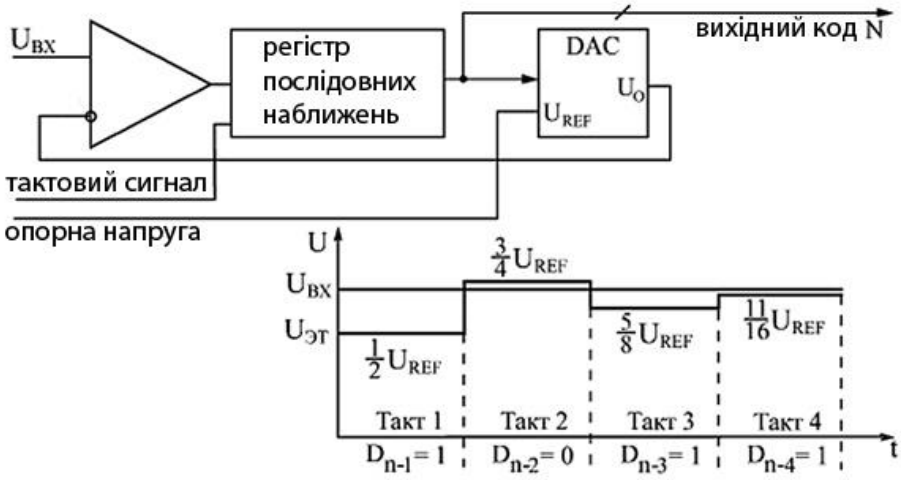


Рис. 4.2. АЦП порозрядного зважування

Радикальним способом скорочення часу перетворення є паралельність обробки. ***Паралельний АЦП*** (рис. 4.3) забезпечує максимально досяжну швидкодію для даної граничної частоти складових елементів. Основою такого перетворювача є лінійка компараторів, кожен з яких порівнює досліджуваний сигнал з власним опорним значенням (рівнем можливого квантованого сигналу), яке формується багаторівневим дільником. Таким чином на виході частина компараторів видає значення «більше», інші – «менше», границя розподілу між «більше» та «менше» визначає результат. Залишається цей результат перекодувати у двійкове число, що може забезпечити відповідний шифратор. За виняткову швидкодію (один такт на перетворення) такого АЦП доводиться платити відповідною складністю (і ціною) приладу. Так 8-бітовий перетворювач вимагає використання 256 (!) компараторів.

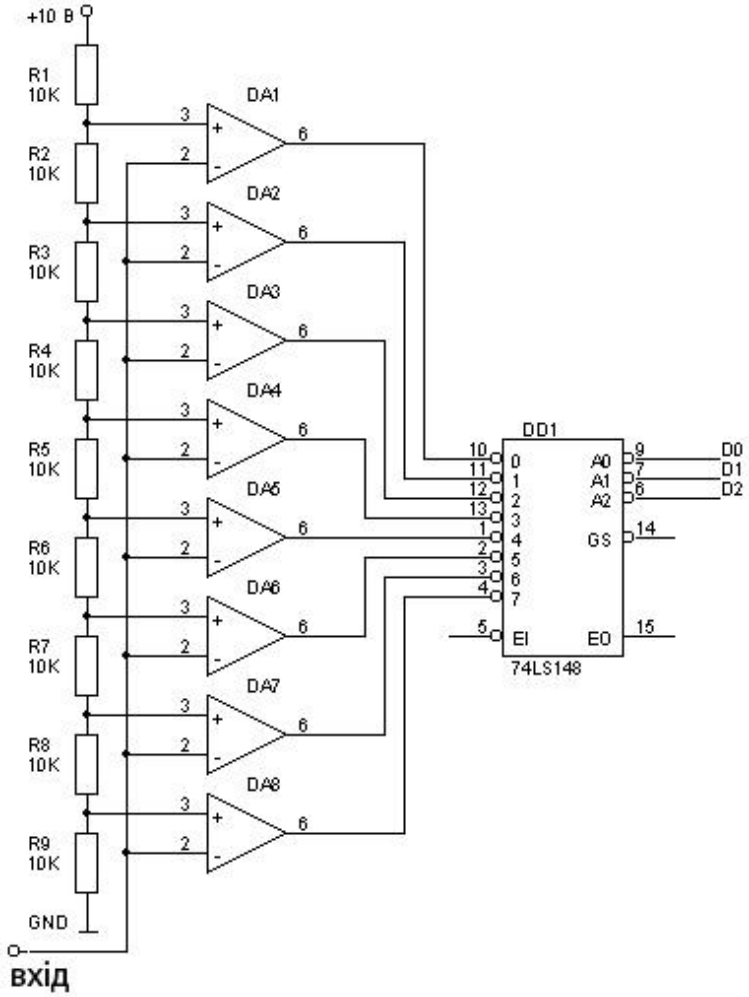


Рис. 4.3. Паралельний АЦП

Компромісом між складністю та швидкодією є ***паралельно-послідовні АЦП***. У цих приладах загальне перетворення відбувається у 2-4 такти, кожен з яких забезпечує послідовне визначення частини розрядів паралельним перетворенням. При цьому можна реалізувати, наприклад, 12-розрядний АЦП, який за 2 такти забезпечує перетворення при потребі 128 (2 по 64) компараторів (замість 1024 тактів для послідовного АЦП, або такої ж кількості компараторів для паралельного). Деяке ускладнення перетворювача подібного типу на основі принципу конвеєризації (п. 5.1.1) дозволяє скоротити перетворення до одного такту. Такі паралельно-послідовні АЦП отримали назву ***конвеєрних***.

Ще одним цікавим компромісом між складністю та швидкодією є ***сигма-дельта АЦП*** (рис. 4.4). Принцип дії цього перетворювача, який реалізується інтегруванням сигналу однобітового перетворювача, заснований на тому, що він виконує лінійне інкрементування або дикрементування коду (залежить від того, яким

був результат останнього порівняння компаратора) від попереднього відліку до поточного. І хоча нібито це схоже на роботу повільного послідовного перетворювача, за рахунок того, що відстежується відмінність наступного значення сигналу не від умовного нуля а від попереднього значення, час перетворення дорівнює тільки декільком тактам при високій розрядності АЦП. Цей факт зробив такий тип перетворювача досить популярним, наприклад, для якісного оцифровувння звукового сигналу.

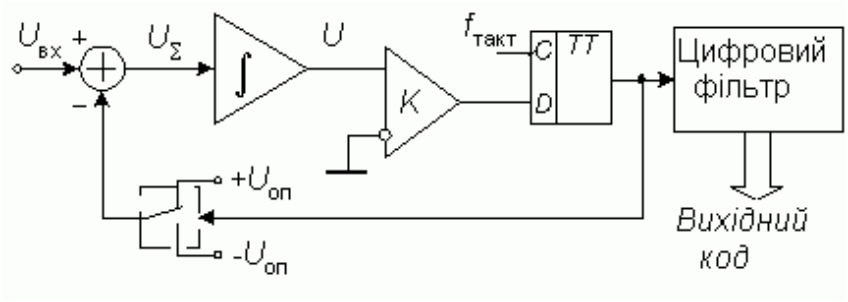


Рис. 4.4. Сигма-дельта АЦП

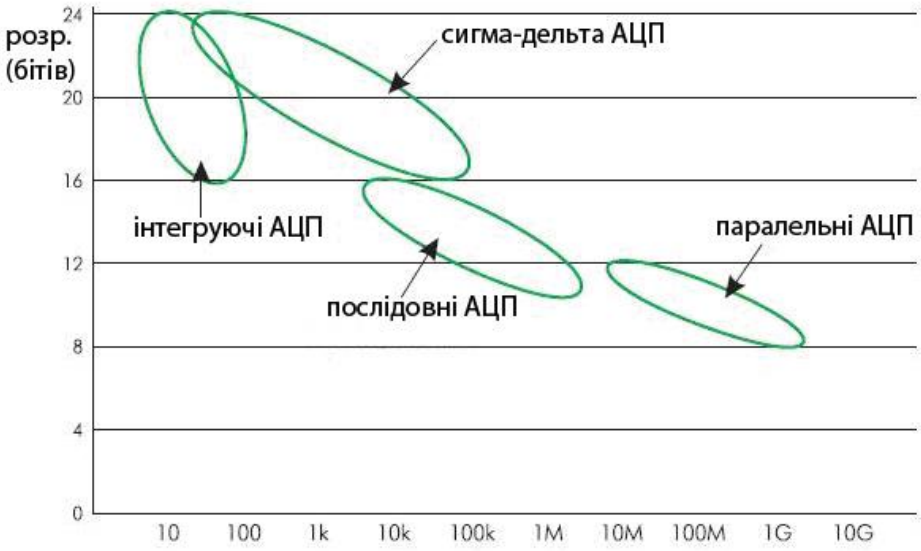


Рис. 4.5. Співвідношення різних типів АЦП [22] (частота – розрядність)

4.1.4. Цифро-аналоговий перетворювач

Виведення інформації з комп'ютера або іншого цифрового пристрою обробки інформації у найпростішому (історично першому) варіанті зводиться до друку на папері імпульсним керуванням механічним важільним засобом друку або формуванням літерного зображення на екрані монітора. При цьому використовується бінарний

режим керування і у достатньо точному формуванні аналогового сигналу потреби немає. Однак розвиток цифрових технологій призвів до необхідності більш складного впливу цифрового керуючого пристрою на аналогові засоби, звичні для людини. Навіть засоби відображення для відтворення напівтонів почали вимагати аналогового сигналу. Таким чином з'явилась потреба у перетворенні, зворотному до аналого-цифрового (п. 4.1.3). Пристрій, який відповідно до цифрового коду, що подається на вихід, формує на виході аналоговий сигнал, (зазвичай електричний), величина (напруга) якого, пропорційна величині вхідного коду, називаються ЦАП – цифро-аналоговим перетворювачем (англійською мовою – Digital-to-analog converter, ***DAC***).

Аналогічно до АЦП найважливішими параметрами ЦАП є лоступний максимальний аналоговий сигнал, який як і в попередньому випадку уніфікується додатковою електронною схемою (в даному випадку на виході), та розрядністю, яка фактично визначає точність перетворення. Одразу зауважимо, що ЦАП на виході за рахунок квантованості буде видавати ступінчастий сигнал. Тому за потреби його додатково згладжують низькочастотним фільтром, який придушує частоти, більші за частоту дискретизації даного сигналу.

Наступним важливим параметром АЦП є швидкодія, тобто час перетворення, Проте компроміс між складністю та досягнення високих частот перетворення, характерний для АЦП (п. 4.1.3), в даному випадку не такий жорсткий. Розглянемо найпопулярніші типи ЦАП та особливості їх реалізації.

Одним з найперших реалізованих методів цифро-аналогового перетворення є ***широтно імпульсна модуляція*** ШІМ). Зауважимо, що цей метод ефективно використовується попри деякі його недоліки і в сучасних приладах. ШІМ спирається на зміну ширини імпульсів незмінної амплітуди та періоду повторення пропорційно до потрібної для формування напруги (рис. 4.6) з наступним інтегруванням отриманого імпульсного сигналу. Швидкодія такого ЦАП визначається вимогою реалізувати керовані прямокутні імпульси з частотою повторення, більшою за граничну частоту вихідного сигналу не менш ніж у 3-5 разів.

Ще одним популярним ЦАП є дельта-сигма ЦАП (інша назва – ЦАП передискретизації). Він дозволяє реалізувати достатньо велику розрядність перетворення на основі меншої розрядності самого перетворювача за рахунок накопичення у часі. Зазвичай, як і для реалізації сигма-дельта АЦП, використовується найпростіший однобітовий перетворювач.

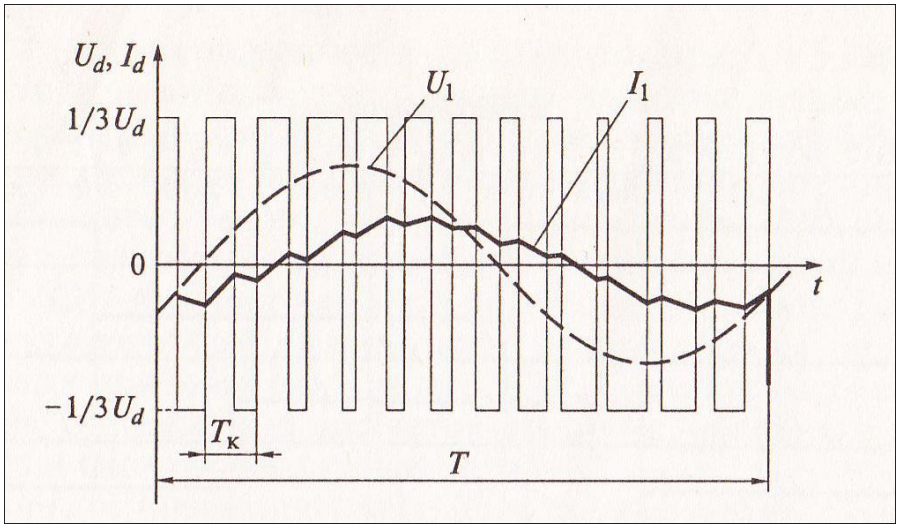


Рис. 4.6. Широтно-імпульсна модуляція

Інший принцип роботи ЦАП спирається на створення деякої лінійки напруг (рис. 4.7) або струмів (рис. 4.8), значення який пропорційні степені двійки, тобто представляють двійкові розряди та відповідно до вхідного коду комутуються у сумарний вихідний сигнал (відповідно підсумовуються напруги або струми).

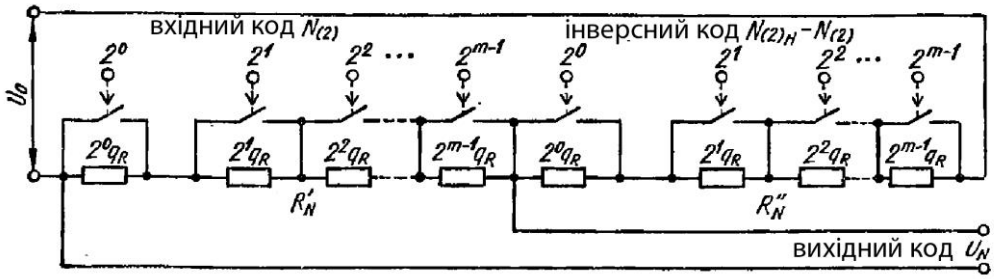


Рис. 4.7. ЦАП з підсумовуванням напруг

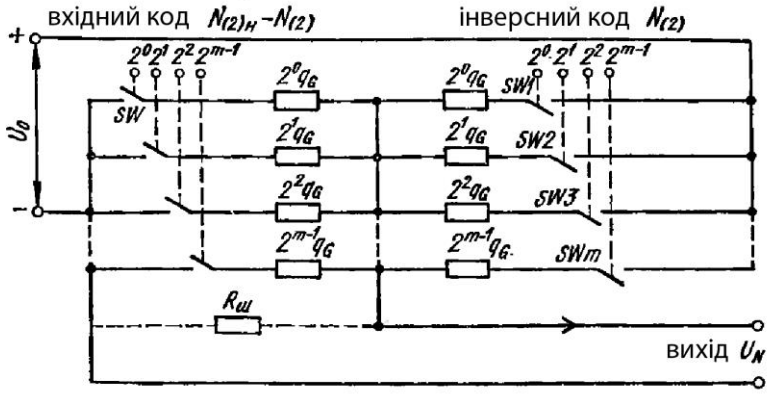


Рис. 4.8. ЦАП з підсумовуванням струмів

Може використовуватись також спеціальним чином сформована драбиноподібна матриця опорів (рис. 4.9), яка дозволяє порозрядною комутацією формувати необхідний опір вихідного дільника. Така матриця є зручною для інтегрального виконання.

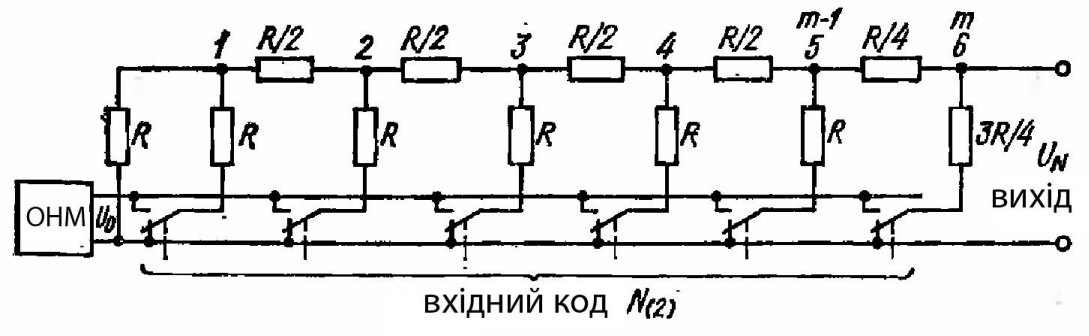


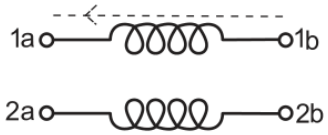
Рис. 4.9. ЦАП з матрицею опорів

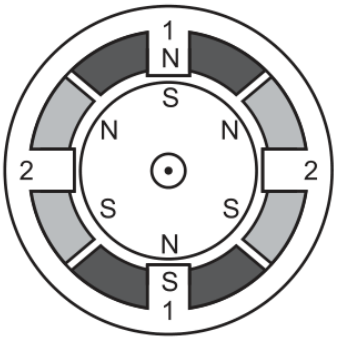
4.1.5. Крановий двигун.

Доволі популярна потреба керування з боку цифрового пристрою (комп'ютера або контролера) механічним зсувом призводить до необхідності реалізації різного типу перетворювачів, тобто пристроїв, що змінюють фізичну природу сигналу. Оскільки дані в цифровій системі формуються і зберігаються у вигляді двійкового коду і легко зводяться до деякої послідовності електричних імпульсів, найбільш ефективним і найбільш поширеним став пристрій, який виконує безпосереднє перетворення коду у зсув. Цим пристроєм є крановий двигун в парі з відповідним контролером. Крановий двигун реалізує механічну частину цього перетворення, а контролер відпрацьовує процедуру перетворення цифрового коду в керуючу імпульсну послідовність та електричне узгодження з обмотками двигуна.

Крокові двигуни є пристроями з деяким набором стійких станів, перехід між якими відбувається за рахунок електричного керування (рис. 4.10). Вони мають статор, на якому розташовані обмотки збудження, і ротор, виконаний з магніто-м'якого (феромагнітного) або з магніто-твердого (магнітного) матеріалу. Другий варіант конструкції забезпечує більший обертальний момент і фіксацію ротора при відсутності струму в обмотках.

Кількість обмоток визначає кількість сталих положень ротору і, відповідно, його кутовий крок. Швидкодія зазвичай визначається його механічною та електричною інерційністю.





	1a	1b	2a	2b
1	+	-	-	-
2	-	+	-	-
3	-	-	+	-
4	-	-	-	+
5	+	-	-	-
6	-	+	-	-
7	-	-	+	-
8	-	-	-	+

а б

Рис. 4.10. Біполярний крокoвий двигун (а) та діаграма його керування (б)

Для редукції (зменшення) кроку двигуна може використовуватись статор та ротор спеціальної зубчастої форми. При цьому активація наступної обмотки призводить не до повної переорієнтації ротору на активну обмотку, а на зсув до найближчого суміщення зубців (рис. 4.11).

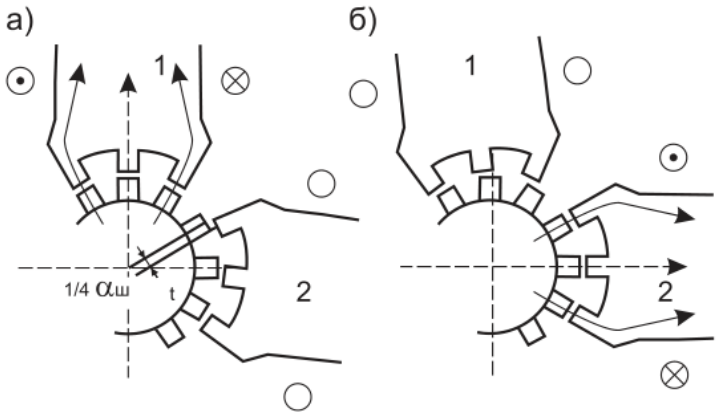


Рис. 4.11. Редукція кроку двигуна

Розглянута конструкція крокового двигуна забезпечує обертальні (кутові) кроки механічного руху. Часто є потреба у лінійному русі. Загалом для перетворення обертального руху у лінійний можна застосувати зубчасту передачу типу шестерня/рейка, але замість цього можна безпосередньо реалізувати лінійний крокoвий двигун (рис. 4.12).

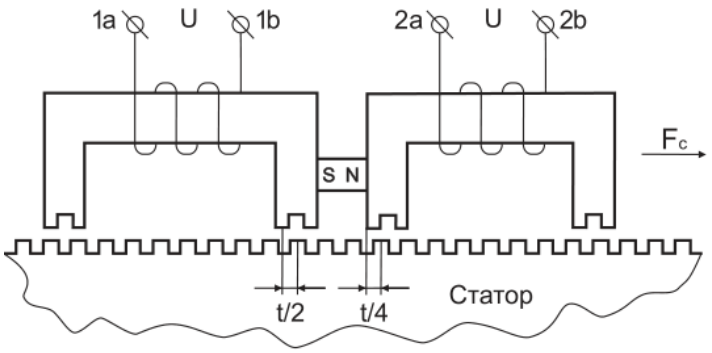


Рис. 4.12. Лінійний кроковий двигун

До переваг крокового двигуна можна віднести:

- високу точність позиціонування (3-5 %) та відсутність накопичення похибки при багатокороковору русі;
- лінійність (кут повороту пропорційний кількості вхідних імпульсів);
- можливість реалізації системи без зворотного зв’язку завдяки високій надійності та точності відпрацьовування заданого руху;
- можливість варіювання в широкому діапазоні швидкості обертання від дуже низьких до досить високих;
- двигун при умові, що обмотки підключені до живлення забезпечує повним момент у стані спокою;
- висока довговічність завдяки відсутності щіток.

До головних недоліків можна віднести:

- доволі малу питому потужність;
- відносно складну схему керування;

4.1.6. Алгоритмізація автоматизації вимірювань

Зазвичай науковий експеримент являє собою отримання зв'язку між сукупністю вхідних і вихідних параметрів, що дозволяє визначити шукані внутрішні властивості досліджуваної системи. Значна частина досліджень зводиться до одержання залежностей однієї фізичної величини для досліджуваної системи від одного вхідного параметру при фіксованих інших. Наявність людини під час експерименту з одного боку гарантує гнучкість керування експериментом, а з іншого боку є додатковим джерелом похибок, особливо коли знімаються динамічні часові залежності. Ці похибки складаються з помилок

експериментатора, паралаксу спостереження при використанні стрілочних приладів, неідеальності синхронізації тощо.

При використанні комп'ютерного керування таким експериментом можуть використовуватись аналогічні алгоритми, що і при ручному дослідженні. Але на відміну від ручного вимірювання, коли експериментатор має змогу за потреби в конкретному дослідженні перемкнути, наприклад, чутливість приладу або полярність напруги, алгоритм експерименту повинен бути запрограмований і перевірений до його початку. Саме коректності алгоритму і врахуванню усіх можливих варіацій початкових умов досліду треба приділити максимальну увагу. Тому при формуванні вхідного параметру має аналізуватись вихідний параметр, як дані для встановлення режимів приладів, що повинно бути включене у програму керування.

Для формування установки частіше за все використовуються ЦАП як джерело сигналу, АЦП як основний вимірювальний пристрій. Для розширення динамічного діапазону вказаних приладів застосовують підсилювачі та дільники. У випадку, коли необхідно забезпечити перемикання, використовують комутатори аналогових сигналів – релейні або електронні. Перші з них мають переваги практично нульового опору з'єднання і двонаправленості передачі сигналу, другі – кращу швидкодію і надійність. При необхідності використовуються різноманітні перетворювачі напруга / іншій фізичний параметр та навпаки.

В найпростішому варіанті алгоритмізація вимірювання функціональних залежностей зводиться до циклу, тіло якого включає інкрементування вхідного параметру і процедуру вимірювання вихідного. При необхідності очікування завершення перехідних процесів між цими операціями додається необхідна пауза. Цикл формує потрібний діапазон вхідного параметра. При такому алгоритмі треба знати можливий діапазони зміни вихідного параметра щоб забезпечити максимальну точність, або виконати програмування автоматичного підбору масштабування перед вимірювальним модулем (ітераційний підбір коефіцієнта підсилення в залежності від отриманого результату вимірюванні). Нагадаємо, що мінімальна похибка приладу відповідає максимальному коефіцієнту використання діапазону вимірювального приладу.

При можливості варіювання параметрів об'єкта дослідження аналогічним чином треба формувати діапазон вхідного параметра виходячи з отриманих результатів для запобігання можливого ушкодження об'єкту дослідження. Це може виконуватись будь-яким із способів дострокового завершення циклу.

При виконанні багатьох наукових експериментів і автоматизації виробничих процесів доводиться забезпечувати утримання одного чи декількох вхідних параметрів експериментальної системи у наперед заданих межах. В найпростішому випадку задача керування зводиться до одержання параметру керування, як функції контрольованого параметра. В найпростіших системах використовуються звичайні аналогові системи із зворотнім зв'язком, аналогічні системі автоматичного регулювання частоти, відомої з радіоелектроніки. В таких системах для формування сигналу зворотного зв'язку використовується елемент, що виконує перетворення відхилення регульованого параметра в напругу, яка після множення на коефіцієнт зв'язку використовується як параметр керування системою. Такі системи досить непогано працюють у випадках однопараметричної задачі в системах з малим часом реакції.

Цифрове керування також може використовувати подібний порядок дій, але при цьому можна реалізувати значно складніші алгоритми розрахунку керуючого сигналу, які можуть спиратись на екстраполяцію та передбачення. Такі особливості можуть суттєво покращити керованість та стабільність результатів.

4.2. Засоби взаємодії з користувачем

4.2.1. Засоби вводу для користувача

Крім безпосередньої взаємодії з технічним обладнанням, яке обслуговує обчислювальний пристрій, (п. 4.1), значна частина пристроїв обробки інформації потребує активної взаємодії з людиною. Це потребує розвитку засобів вводу інформації та її представлення, зручних саме для роботи користувачів. При цьому значна увага приділяється ефективності роботи непрофесійних користувачів, тобто тих, для кого робота з комп’ютером не є основною діяльністю (потрібні простота та зрозумілість використання). Введення та виведення найбільш ефективної (графічної) форми даних було розглянуто у пп. 3.1, 3.3. Але для повноцінної роботи користувача з засобами обробки інформації цього не достатньо, оскільки введення зображень не надає користувачу зручного каналу для керування обробкою даних. Тобто потрібні інші засоби, яки з самого початку розвитку обчислювальної техніки сформувались як обов’язкові периферійні пристрої.

Історично першим засобом вводу даних до комп'ютера стала клавіатура. На початку розвитку комп'ютерних технологій це була клавіатура телетайпу або приєднаної до комп'ютера електричної друкарської машинки. Трохи пізніше до цього додалась клавіатура пристроїв для підготовки перфокарт та клавіатура перших електронних дисплеїв (спеціальних комбінованих термінальних пристроїв, які об'єднали електронну клавіатуру з телевізійним алфавітно-цифровим монітором). При цьому структура клавіатури мало змінилась, залишившись спадкоємцем вже відпрацьованої друкарської машинки, оскільки основний відсоток потоку вводу зазвичай припадає саме на літери. Відповідно використовується розкладка літер по клавішам, адаптована до кожного з алфавітів (наприклад, латиниці) таким чином, що літери, які використовуються більш часто розташовані ближче до середини клавіатури. Для зручності до літерної клавіатури додалась калькуляторна та курсорна клавіатури, сформувався стандарт додаткових службових та функціональних клавіш. Доволі часто для зручності додаються спеціалізовані клавіші, наприклад, для керування настройками мультимедійної підсистеми комп'ютера.

Сучасна клавіатура фактично не є прив'язаною до конкретних літер, замість цього використовуються коди клавіш, які формує спеціальний контролер, що опитує клавіатуру. Це дозволяє простим перекодуванням (заміною так званої кодової сторінки, що задає таблицю кодування) забезпечити введення літер довільної мови.

Не зупиняючись на технологічних особливостях реалізації механіки клавіш, відмітимо найбільш цікаві тенденції розвитку клавіатури:

- перехід до короткого ходу клавіш;
- ергономічність для зменшення навантаження на кисті рук:
- підсвічування клавіш;
- комп'ютерне керування зображенням символів на клавішах;
- віртуальна на сенсорному екрані;
- віртуальна на довільній поверхні (проеціюються світлові зображення клавіш).

Останній час розробники активно працюють на альтернативою клавіатурного вводу, вдосконалюючи розпізнавання мови та рукописних літер, що в перспективі значною мірою зменшить потребу у клавішному наборі.

Іншим пристроєм вводу інформації від користувача є комп'ютерна мишка. За час свого існування цей пристрій змінився досить сильно як ззовні, так і в середині. Мінімально мишка призначена для вказування координат, тобто повинна включати

відстеження двокоординатного руху та натискання (однієї кнопки). Саме це і забезпечувала перша мишка (1964 р.). Далі для розширення функціональності збільшилась кількість кнопок спочатку до двох, потім до трьох (іноді використовується і значно більша кількість), додався ролик (іноді віртуальний) для завдання прокрутки контента. Сучасні мишки як третю кнопку використовують натискання ролика прокрутки. Є й більш екзотичні мишки, але на їх особливостях зупинятись не будемо через їх малу поширеність. З точки зору технологій, більше за все приділяється уваги технології точного вказувань зміни координат. Перша мишка для цього мала два ролика, ортогональних один до одного. Потім до цих роликів додалась кулька, яка більш точно розподіляла по них двокоординатний рух. Обертання роликів реєструвалось оптопарою з зубчастим коліщатком, яке під час обертання періодично переривало світловий промінь.

Сучасні мишки для відстеження руху використовується порівняння кадрів від мініатюрної відеокамери, яка реєструє зміну зображення підкладки безпосередньо під мишкою. Такі мишки використовують або некогерентну підсвітку (оптичні мишки) або когерентну (лазерні). Лазерні мишки відрізняються більшою точністю і меншою вибагливістю до підкладки.

Альтернативою використання мишки є сенсорна панель (англійською мовою touchpad), яка є обов'язковим пристроєм для ноутбуків, та сенсорний екран.

Сенсорна панель є сіткою з металевих провідників, розділених тонкою ізолюючою прокладкою з лавсановій плівки, що створює матрицю конденсаторів. Ємність конденсатора змінюється при торканні пальцем поверхні панелі, за рахунок великої провідності тіла людини. Сила натискання через зміну плями торкання також впливає на ємність. Реєстрація зміни ємностей дозволяє реалізувати повноцінне двокоординатне цілевказування, хоча й дещо менш точно, ніж це забезпечую мишка.

Альтернативою матриці конденсаторів є розташування лінійних сенсорів вздовж вертикальної та горизонтальної осей з вимірюванням ємностей між пальцем та сенсором.

Перенесення подібної технологій з окремого непрозорого пристрою, призначеного виключно для вказування координат, на прозорий шар екрану дозволило значно розширити можливості використання такого пристрою. Нова технологія отримала назву сенсорного екрану і стала основним засобом вводу даних на компактних пристроях – смартфонах та планшетах, дозволила відмовитись від механічної клавіатури за рахунок програмної віртуальної. Навіть на пристроях значно більшого форм-фактору

сенсорний екран почав завойовувати все більш значні позиції. Перевага подібного вводу даних пов'язана з суттєвим покращенням інтерактивності за рахунок накладання полів вводу і виводу інформації. Сучасні сенсорні екрани використовують виключно ємнісну технологію, яка спирається на вимірювання реактивного опору між пальцем та електродами у кутках екрану, або ємність між пальцем та сіткою електродів, яку, наприклад, можна нанести між субпікселями екрану.

Серед перспективних досліджень засобів цілевказування можна відмітити відстеження за допомогою відеокамери та розпізнавання використанням розвинених програмних засобів з цього зображення рухів користувача, його окремих жестів і, навіть, напрямку погляду. Є також системи трикоординатного відстеження у просторі спеціальних маркерів.

4.2.2. Засоби введення звуку

Сучасні системи цифрової обробки інформації активно використовуються такі надважливі для людини типи інформації, як звукова. Таким чином задача ефективного та високоточного введення подібної інформації є дуже важливою.

Технології введення звуку (фактично – залежності від часу миттєвого значення тиску повітря) до цифрових систем передбачають послідовне використання двох перетворювачів:

- тиск/електричний сигнал;
- АЦП.

Частотний діапазон звуку (до 20 КГц) не викликає проблем з реалізацією АЦП потрібної швидкодії при достатній розрядності (до 12-16 бітів). Таким чином уся технологія введення звуку може бути успадкована від класичних аналогових технологій створення високоякісних мікрофонів. До основних робочих характеристик мікрофону відносяться:

- чутливість (відношення напруги на виході мікрофона до звукового тиску);
- амплітудно-частотна характеристика;
- рівень шуму (відношення ефективної напруги на виході мікрофона за відсутності звукового поля до напруги при наявності звукового поля з ефективним тиском у 0,1 н/м²);
- направленість.

Найбільш поширеними на поточний момент типами мікрофонів, що використовуються в парі з цифровими системами є:

- електродинамічний;
- електромагнітний;
- стрічковий;
- конденсаторний;
- п’єзоелектричний.

Кожен з ним має свої переваги і недоліки і отримав розповсюдження для окремого класу задач. Наприклад, для високоякісного введення звуку найкращим є електродинамічний (мембрана мікрофону змінює положення котушки в полі постійного магніту) та стрічковий (фактично – різновид електродинамічного). П’єзокерамічний на відміну від зазначеного відрізняється найменшими розмірами та ціною.

4.2.3. Засоби виведення звуку.

Виведення звуку зводиться до формування електричного сигналу та подальшого його перетворення у зміну в часі тиску повітря, що і є звуком. Першу з цих задач виконує ЦАП. Як і при введенні звуку в цифрову систему, особливих проблем зі швидкодією перетворення не виникає. Відповідно при виборі перетворювача вибирається достатня для забезпечення потрібної якості розрядність ЦАП та використовується придушення вищих за частоту дискретизації частот низькочастотним фільтром.

За відсутності необхідності реалізувати надзвичайно високу якість звуку, яка визначається:

- малим рівнем нелінійних спотворень;
- високим співвідношенням сигнал/шум;
- високою лінійністю амплітудно-частотної та фазо-частотної характеристик,

особливих проблем з подальшим перетворенням також немає. Основними типами акустичних перетворювачів є:

- електродинамічні;
- електростатичні;
- п’єзоелектричні.

Їх конструкція та технологічні особливості добре відпрацьовані ще до активного використання звуку у комп’ютерних технологіях. Дещо складніше вирішується подальше перетворення електричного сигналу в акустичний при потребі надвисокої якості. В цьому випадку використовуються додаткові ускладнення конструкції перетворювачів (головок), високоякісні аналогові підсилювачі з малими фазовими викривленнями, багатосмугові (тобто з декількома голівками)

акустичні системи, в тому числі з акустичним фазоінвертором. У найскладніших системах може використовуватись навіть акустичний зворотний зв'язок. Більш детально це питання розглядати не будемо, оскільки прямого відношення до інформатики ці особливості реалізації акустики не мають. Зупинимось тільки на тих особливостях, які співпали з розвитком цифрових технологій обробки сигналу.

Одним з факторів правдоподібності відтворення звук є його «об'ємність», тобто надання людині можливості визначити напрямок надходження звукового сигналу, точніше напрямок кожного зі складових загального звукового сигналу. Природним апаратом для визначення напрямку звуку є наявність двох рознесених в просторі приймачів сигналу (вух), які реєструють відмінність фази. Відповідно ця особливість була використана при реалізації технології відтворення двома акустичними системами, звуку записаного двома незалежними мікрофонами. Сучасні можливості цифрового запису та обробки сигналів дозволили трохи розвити ці технології, розклавши сигнал по більшу кількості каналів. Це дозволяє крім забезпечення визначення напрямку при статичному положенні голови слухача (джерело зліва–справа) реалізувати динамічне розрізнення при зміні положення голови (джерело попереду–позаду). Зауважимо також, що у подібних системах відокремлюється низькочастотний канал з єдиним джерелом (сабвуфером), оскільки при довжині звукової хвилі значно більшій за відстань між вухами інформація про фазу втрачається. Іноді також використовується додаткова цифрова фільтрація каналів для поглиблення стереоефекта, або його імітації при відсутності реального розділення каналів.

Комп'ютерні технології призвели до значно більших змін в технологіях відтворення звуку у задачі синтезу, що значно змінило характеристики відповідних музичних інструментів. Цифровий синтез музичних інструментів охоплює дві основні технології:

частотний адитивний синтез;

хвильовий синтез (wavetable).

Частотний синтез визначає формування тембру інструменту додаванням до основного тону (ноти) субгармонік з необхідним співвідношенням, що є досить зручною задачею для цифрових технологій. Саме тому такий варіант синтезу досить довго використовувався у комп'ютерних звукових картах на початку ери персональних комп'ютерів. Хвильовий синтез вимагає більших ресурсів. Метод полягає у тому, що для кожного інструменту оцифровуються і записуються семпли на деяких опорних нотах. Проміжні за частотою семпли розраховуються інтерполюються з найближчих семплів у записаній таблиці. З врахуванням високої

точності розрахунку інтерпольованого семплу та використання для запису зразків високоякісних інструментів, цей метод забезпечує значно кращі результати і активно використовується як у комп'ютерах, так і у цифрових музичних синтезаторах.

4.3. Питання для самоконтролю

1. Чому потрібно автоматизувати повільні вимірювання?
2. Чи є сенс виконувати автоматизацію одноразового дослідження?
3. В чому полягає автоматизація вибору діапазону вимірювання?
4. Чи є коректним казати про «метод вимірювання опору зважуванням»?
5. Чи може використовуватись міра іншої природи, ніж досліджуваний сигнал ?
6. В чому перевага використання багатоканальної міри?
7. Чи є класичні аналітичні ваги компаратором?
8. Який АЦП складніший для реалізації – послідовний, чи паралельний?
9. Чому для ЦАП проблема швидкодії не є такою складною, як для АЦП?
10. Де використовується широтно-імульсна модуляція?
11. Чи можна кроковим двигуном реалізувати безступінчастий рух?
12. Запропонуйте спосіб керування зсувом без крокового двигуна.
13. Для чого в конструкції мишки використовують лазер?
14. Чому для відтворення низьких частот звуку в комплекті акустичних систем часто використовують один перетворювач (сабвуфер)?

5. Апаратні засоби обробки даних

5.1. Забезпечення ефективності обробки даних

5.1.1. Архітектура сучасних процесорів

У попередніх главах ми розглянули основні принципи обробки інформації та її програмної реалізації. Проте виконання будь-якої програми вимагає наявності відповідного програмованого пристрою, на якому вона може бути виконана. Крім того накопичення та передача інформації відбувається через певні фізичні сигнали, що також обов'язковим чином використовує спеціальні апаратні пристрої. Загалом сукупність усіх подібних засобів складає апаратне забезпечення обчислювальної техніки. Значною мірою усі складові апаратного забезпечення підлаштовуються під засоби обробки, тобто процесори. Саме тому першим серед апаратних пристроїв розглянемо саме процесор, як правило найскладнішу мікросхему будь-якого сучасного програмно-керованого пристрою.

Процесором називається апаратний (в сучасних системах практично завжди – електронний) блок для виконання послідовності команд деякої програми.

Відповідно саме пристрій виконання зазвичай є найбільш критичним для досягнення тих характеристик, які необхідно отримати від усієї системи обробки даних. При проектуванні або виборі процесора необхідно обрати ті характеристики, які є для системи найбільш важливими. Серед цих характеристик слід виділити:

- мінімальній час виконання програми;
- мінімальна ціна;
- мінімальне енергоспоживання;
- максимальна надійність в певних експлуатаційних умовах;
- прийнятні розміри.

Зрозуміло, що ідеальним був би процесор, найкращий за усіма зазначеними ознаками, але на жаль практично усі вони вступають у протиріччя з іншими. Відтак доводиться для кожної конкретної задачі шукати певний компроміс, в тому числі у виборі архітектури процесора, яка у свою чергу значною мірою пов'язана з архітектурою усього обчислювального пристрою або системи.

Архітектура обчислювального пристрою представляє концептуальну структуру цього пристрою (п. 1.2.2). Вона

визначається архітектурою процесора та доповнюється потребою реалізації функціонального навантаження, яке є обов'язковим для пристрою. Але не є реалізованим безпосередньо у процесорі. Основною з таких додаткових функцій є узгодження обміну даними з іншими пристроями (підсистемою пам'яті, пристроями вводу-виводу, мережею, тощо). У сучасних пристроях реалізація цих функцій виконується у вигляді деякого ***набору мікросхем*** (англійською мовою – ***chipset***).

При реалізації обчислювального пристрою архітектурні рішення тісно пов'язані з потребою компонування, яке залежить від задач пристрою, наприклад, сервера, десктопа, ноутбука або смартфона.

Архітектура сервера обирається з потреб великої обчислювальної потужності і збереження значних обсягів даних. Але навіть при таких вимогах доводиться шукати компроміс між характеристиками та ціною. Зазвичай серверна архітектура передбачає можливість об'єднання декількох блоків (корпусів) у стійку, використання зовнішнього по відношенню до корпусу блоку дискових накопичувачів, наприклад RAID-масиву (п. 5.2.3). У серверному блоці використовуються найбільш потужні процесори, зазвичай саме орієнтовані на роботу з багатьма завданнями одночасно, активно використовується багатопроцесорність, тобто декілька процесорів в блоці. Особлива увага приділяється максимальній надійності і можливості ремонту з мінімальним простоєм або взагалі без зупинки (вилученням з роботи окремого блоку з передачею його задач на інші аналогічні).

Десктоп зазвичай проектується на основі материнської плати (англ. - motherboard), з активним використанням модульності, тобто можливості роз'ємного приєднання до цієї плати інших блоків – процесора, модулів пам'яті, відеоадаптера, дискових пристроїв, тощо. В залежності від реалізації материнської плати деякі з подібних модулів можуть бути виконані інтегрованими, тобто бути невід'ємною частиною цієї плати. Модульна конфігурація дозволяє адаптувати комп'ютер під задачі конкретного користувача (обрати процесор, об'єм пам'яті, графічної систему тощо).

Ноутбук є компактнішим пристроєм, що ускладнює свободу розташування модулів. Це вимагає додаткових компромісів при проектуванні розміщення модулів та використання інтегрованості. Вимога довшої роботи від акумулятора накладає певні обмеження і на вибір процесора та графічної системи, віддаючи перевагу менш вибагливому до потреб живлення складовим, тобто жертвуючи загальною швидкодією. Відповідно ноутбук має значно менші можливості апаратної адаптації зо певних задач та модернізації.

Розбирання та збірка такого комп'ютера з метою профілактичного обслуговування або ремонту є значно складнішою за десктоп чи сервер.

Максимально компактні програмовані пристрої (планшети, смартфони) є ще більш критичними до живлення та вимагають щільної компоновки, що зводить модульність до мінімуму. Фактично компоновка при проектуванні враховує конкретні обрані для пристрою модулі, в тому числі спеціально для цього виробу розроблені і не передбачає заміни на аналоги. Відповідно вони мають ще меншу ремонтопридатність.

Детальніше особливості архітектури програмованого пристрою загалом розглядати не будемо, і перейдемо безпосередньо до архітектури самого процесора. На рис. 5.1 як приклад наведена спрощена схеми універсального процесору одного з старих поколінь, сучасні процесори є значно складнішими.

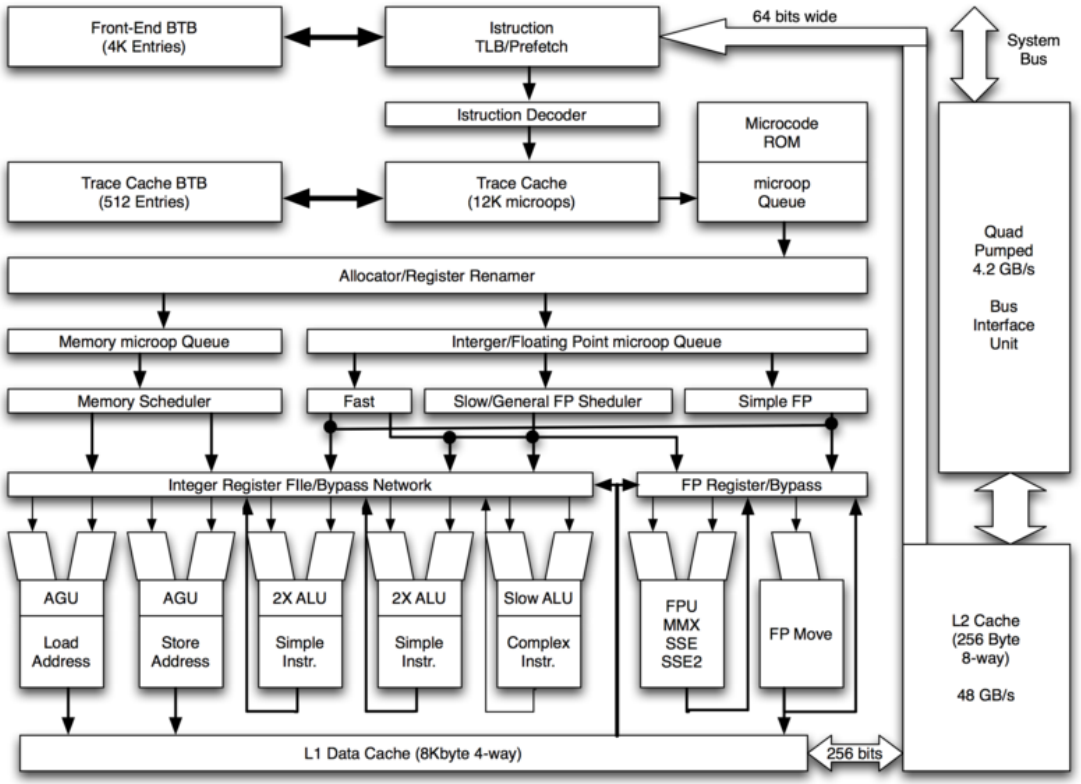


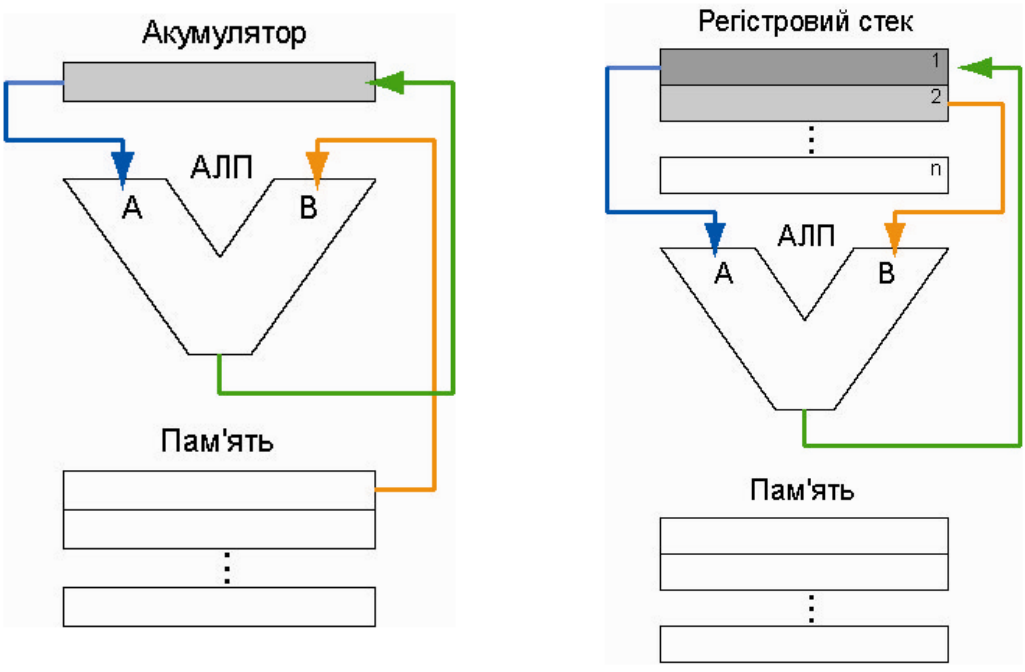
Рис. 5.1. Приклад архітектури процесору (Pentium 4 на ядрі Willamette)

Виходячи з основ обробки інформації (розділ 1), процесор повинен виконувати команди з деякого набору (множини) команд. Для цього код кожної наступної команди має бути отриманий ззовні (по відношенню до процесора) та розпізнаний (дешифрований). Після цього для виконання вже конкретизованої команди повинні бути

отримані операнди (п. 1.1.4), наприкінці в певне зазначене місце треба записати результат.

Пристрій процесора, який безпосередньо виконує команду, оскільки значний відсоток команд пов'язаний з арифметичними і логічними діями, зазвичай називають арифметико-логічним пристроєм (АЛП, англійською мовою – ALU, arithmetic and logic unit). Зауважимо, що не зважаючи на те, довільна арифметична операція з дійсними числами може бути виконана на тому ж пристрої, що і обробка аналогічної цілочисельної (для цього використовується певна програмна реалізація дії), у сучасних системах, для яких критичною є швидкодія, дійсну арифметику виконує окремий блок.

Для тимчасового збереження даних над якими виконуються дії, з метою забезпечити максимально швидкий доступ до них використовуються регістри, тобто спеціальні блоки які є частиною самого процесора. Оскільки регістри виконуються на елементній базі, яка є основою АЛП, вони мають аналогічну швидкодію. Слід зауважити, що структура системи регістрів процесора та їх взаємодія з АЛП також впливає на швидкодію обчислень, причому для різних класів задач різним чином. Таму структура системи регістрів також є частиною архітектури процесора. Як приклад можна навести базову архітектуру з акумулятором (рис. 5.2.а).



а б

Рис. 5.2. Варіанти базової архітектури процесора

Акумулятором називається регістр, до якого повертається результат операції. Така архітектура цікава тим, що дуже часто в

обчисленнях результат попередньої дії одразу є одним з наступних операндів. Цікавою альтернативою є архітектура регістрового стека (рис. 5.2.б). Є й інші варіанти подібних архітектур.

Інша частина архітектури процесора включає у себе оптимізацію буферизації потоків команд і даних та внутрішньої передачі інформації між пристроями процесора.

Загалом немає універсального рецепту до вибору архітектури процесора, яка вдовольняє усім можливим потребам. Крім того постійний прогрес обчислювальної техніки та швидке зростання складності програмного забезпечення та самих задач, які воно розв'язує, вимагає розвитку цієї архітектури. Одразу зауважимо, що детальний розгляд такої архітектури виходить за рамки даної дисципліни і вимагає окремого навчального курсу. Тому розглянемо тільки деякі найбільш важливі шляхи забезпечення швидкодії.

Історично першим методом підвищення швидкодії було збільшення тактової частоти, яка саме і задає час виконання однієї операції у потоку команд. Зрозуміло, що ця частота обмежується граничною частотою базового елемента, тобто транзистора у ключовому режимі (ключа), з сукупності яких складається схема процесора. Для потреб якісної обробки імпульсу, тобто чіткого виділення його фронтів, гранична частота транзистора повинна майже на порядок перевищувати тактову. Відповідно збільшення тактової частоти вимагає збільшення граничної і розв'язується в першу чергу технологічним шляхом. Зараз такий шлях збільшення швидкодії вже значною мірою вичерпаний і не може вдовольнити потреби розробників.

Наступним шляхом є збільшення розрядності, тобто кількості бінарних розрядів, з якими одночасно виконується операція. Якщо процесор має обробити число з розрядністю, що перевищує розрядність АЛП, дія виконується з частиною операнду, а потім обробляються інші розряди з врахуванням результату попередньої часткової дії. Зрозуміло, що в цьому випадку виконання операції буде значно повільнішим. Саме тому, якщо пригадати історію розвитку архітектури процесорів, кожне подвоєння (так історично склалось) їх розрядності приблизно вдвічі збільшувало загальну швидкодію обробки даних. Однак слід зауважити, що у цьому є вигода тільки в тому випадку, коли серед даних є значна частина, що не вкладається у попередню розрядність процесора. Саме тому перехід від 32-розрядної архітектури до 64-розрядної вже не дав помітного виграшу у швидкодії виконання арифметичних дій. Тобто цей шлях також є вичерпаним. Зауважимо, що 64-розрдна архітектура має переваги у можливості звернення до більшого за розміром адресного простору,

що критично для особливо складних задач, тобто не можна сказати, що останній крок у збільшенні розрядності не був некорисним.

Протягом історії розвитку процесорів розвивали і внутрішню структуру АЛП з метою зменшити кількість тактів на виконання однієї операції. Для цього ускладнювався безпосередньо обробник операції та було застосовано конвеєризацію. Ідея ***конвеєризації*** полягає у тому, що команда процесора виконується як послідовність більш елементарних операцій – як вже зазначалось, команду треба отримати, розпізнати, отримати операнди, виконати, записати результат. Реально таких підкроків ще більше. Таким чином можна для кожного з підкроків реалізувати свій пристрій виконання, який результат своєї дії передає наступному пристрою. І в той час як наступник продовжує виконання цієї команди, можна виконувати свій підкрок вже для наступної команди. Приблизно так, наприклад, робітники збирають, автомобілі на конвеєрі. При цьому збільшення кількості підкроків повинно збільшувати і загальну швидкодію, але до певної межі. Додатковими особливостями збільшення ефективності конвеєрної архітектури є система передбачення галужень, перейменування регістрів, перемикання блоків регістрів тощо.

Поки що ми не згадали ще одну особливість архітектури процесора, яка також значною мірою впливає на швидкодію – будову самої системи команд, тобто визначення множини елементарних дій, які може виконувати пристрій та способу їх кодування. Зрозуміло, що множина команд повинна бути самодостатньою. Бажано залишити можливість розширення цієї множини для потреб подальшого розвитку (наступних поколінь процесора). Звідси випливають певні обмеження на спосіб кодування (формат) команди, тобто які з розрядів за яку частину коду відповідають. Загалом система команд традиційно розділяється на дві конкуруючі гілки:

- ***CISC*** (***C***omplex ***I***nstruction ***S***et ***C***omputer - комп'ютер зі складним набором команд)
- ***RISC*** (***R***educed ***I***nstruction ***S***et ***C***omputer - комп'ютер зі спрощеним набором команд)

CISC передбачає можливість варіювання довжини поля команди, що спрощує подальше розширення системи команд і ускладнення формату. Але за ці переваги доводиться платити більш складною і повільною процедурою дешифрування.

RISC за основну особливість має обмеження внутрішньої структури команди та довжини її коду, що пришвидшує її дешифрування. Але це призводить до потреби в багатьох випадках зводити команду, яка реалізуються у CISC як одна елементарна, до декількох послідовних RISC команд. До речі, через це порівняння

швидкодії для цих двох типів архітектури команд тільки за кількістю виконуваних команд за одиницю часу є дещо некоректним.

Кожна з таких гілок розвитку архітектури процесорів має свої переваги і недоліки, які проявляються різним чином у різних задачах. Відповідно обидві активно розвиваються. Прикладом CISC є системи команд x86 та x64 від Intel та AMD. Найбільш розповсюдженим зараз прикладом RISC процесорів є ARM, що використовується у планшетах, смартфонах та інших компактних пристроях.

Наявність багатьох різних систем команд є великою проблемою при розробці програмного забезпечення, оскільки програму у командах одного процесора не розуміє інший, тобто має місце ***програмна несумісність***. Зазвичай програмні засоби розробляються на мовах високого рівня (п. 1.3.1), які декларуються як універсальні. Для роботи на конкретному процесорі цей універсальний код транслюється у відповідну систему команд. Таким чином для забезпечення програмування конкретного процесору з своєю системою команд є потреба реалізувати транслятор під цю систему команд. Реальна ситуація дещо складніша, оскільки текст програми пишеться і транслюється не тільки під систему команд, а й під операційну систему а відповідні засоби розробки мають специфічні бібліотеки для узгодження з компонентами певної ОС. До цього можна додати проблему розширення раніше розробленої системи команд. Найбільш яскравим прикладом порушення сумісності при цьому є різні розширення для пришвидшення розрахунку мультимедійних потоків фірмами Intel та AMD.

Окремо стоїть проблема збереження сумісності старого програмного забезпечення на новому поколінні процесору, який має більшу розрядність. Для забезпечення такої сумісності нова архітектура розробляється таким чином, щоб архітектура попереднього покоління була підмножиною поточного, тобто стара система команд є незмінною і повністю виконується новим поколінням, а структура регістрів та інших пристроїв будується як надбудова над попередньою.

Зазначені вище шляхи збільшення швидкодії вже майже вичерпані в не можуть забезпечити подальшого прогресу. Тому основним сучасним шляхом збільшення швидкодії є використання паралельності виконання операцій. Суть цього підходу полягає у тому, що у більшості розрахункових задач, до яких зводиться обробка інформації, окремі фрагменти послідовного програмного коду можуть бути виконані одночасно на різних апаратних пристроях. Це стосується як декількох незв’язаних задач, що виконуються одночасно, так і однієї окремої задачі. Навіть наявність окремих

конвеєрів для цілих та дійсних операндів за рахунок суміші у виразі може прискорювати роботу. При виконанні на одному конвеєрі певної дійсної дії одночасно на іншому виконується ціла операція з цього ж виразу. Ще більш ефективним є використання декількох дійсних і декількох цілих конвеєрів.

Сучасна мікросхемотехніка дозволяє в одному кристалі сформувати декілька «майже окремих процесорів» – ядер (рис. 5.3), що дозволяє у декілька разів збільшити швидкодію процесора, особливо, коли виконуються незалежні задачі. Для довільної задачі одні фрагменти коду виконуються практично незалежно від сусідніх, тобто такий код ефективно розпаралелюється, а інші частини коду є практично повністю послідовними і спроба їх розпаралелення призводить до простоїв. Чим меншою є відсоток «поганого» коду, тим ефективнішим є використання паралельних обчислень і, відповідно, багатоядерної архітектури процесорів.

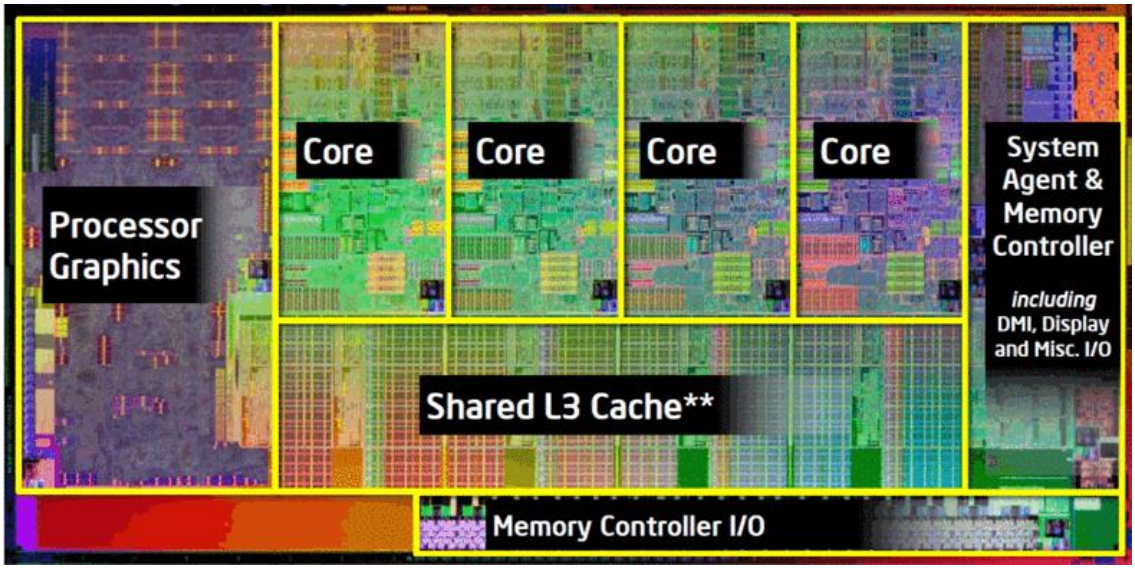


Рис. 5.3. Приклад структури кристалу багатоядерного процесору (Intel Core i5 Sandy Bridge)

Фірма Intel у напрямку розпаралелення задачі пішла ще далі впровадженням технології виділення незадіяних ресурсів апаратного ядра у віртуальне. Цю технологію було названо ***Gyper-threading***. У задачах із значними простоями (коли одному ядру доводиться певний час очікувати результати іншого), ця технологія забезпечує досить помітний виграш. В інших задачах виграш може бути майже непомітним. Загалом можна вважати, що одне віртуальне ядро за середньою швидкодією дорівнює приблизно половині справжнього

Найбільш ефективним є розпаралелення обрахунків у графічному процесорі, тобто спеціалізованому процесорі, який виконує основні обрахунки графічного адаптера. Це пов'язано з тим, що розрахунок пікселів, які складають кадр для виведення на екран, є практично незалежними один від одного. Відповідно ця особливість активно використовується при розробці архітектури засобів обробки графіки. Зауважимо, що останні покоління універсальних процесорів часто мають інтегрований у кристал додатковий графічний процесор (рис. 5.3). Це дозволяє при невибагливості до синтезу графіки відмовитись від додаткового дискретного відеоадаптера.

5.1.2. Особливості мікроконтролерів

Серед програмованих пристроїв крім процесорів (більш точно – універсальних процесорів) слід виділити мікроконтролери. Назва «мікроконтролер» пов'язана з мікросхемним виконанням цифрового контролера (керуючого програмованого пристрою). Потребу у таких пристроях має більшість сучасного обладнання – агрегати верстатів, літаків, автомобілів, побутові прилади, навіть найпростіші пристрої дистанційного керування. Контролери на відміну від комп'ютерів програмуються на одну конкретну задачу, яку виконують протягом усього свого періоду використання. Така задача у більшості випадків значно менш ресурсоємна, ніж обчислювальні задачі, які виконують навіть домашні комп'ютери, але дуже критичною є можливість взаємодії з додатковими апаратними засобами, якими керує мікроконтролер.

Загалом роль контролера може виконувати довільний універсальний процесор з відповідним до потреб конкретної здачі оточенням. Але така реалізація у багатьох випадках не буде оптимальною через більшу ціну, розміри та зайву потребу у живленні (що у випадку компактного мобільного пристрою призводить до скорочення часу роботи від акумулятору). В той же час такий процесор часто буде мати надлишкову швидкодію. Саме тому активно розробляються та виготовляються мікроконтролери специфічної архітектури (рис. 5.4), ознаками якої є:

- менша (в порівняння з універсальними процесорами) розрядність та простіша архітектура обчислювального пристрою;
- інтегрований тактовий генератор та таймери;
- інтегровані запам'ятовуючі пристрої – оперативний (ОЗП) та постійний ПЗП (п. 5.2);
- інтегровані стандартні інтерфейси (п. 5.3);

- інтегровані АЦП (п. 4.1.3), ЦАП (п. 4.1.4), контролери крокових двигунів (п. 4.1.5), інші засоби зв'язку з апаратними засобами.

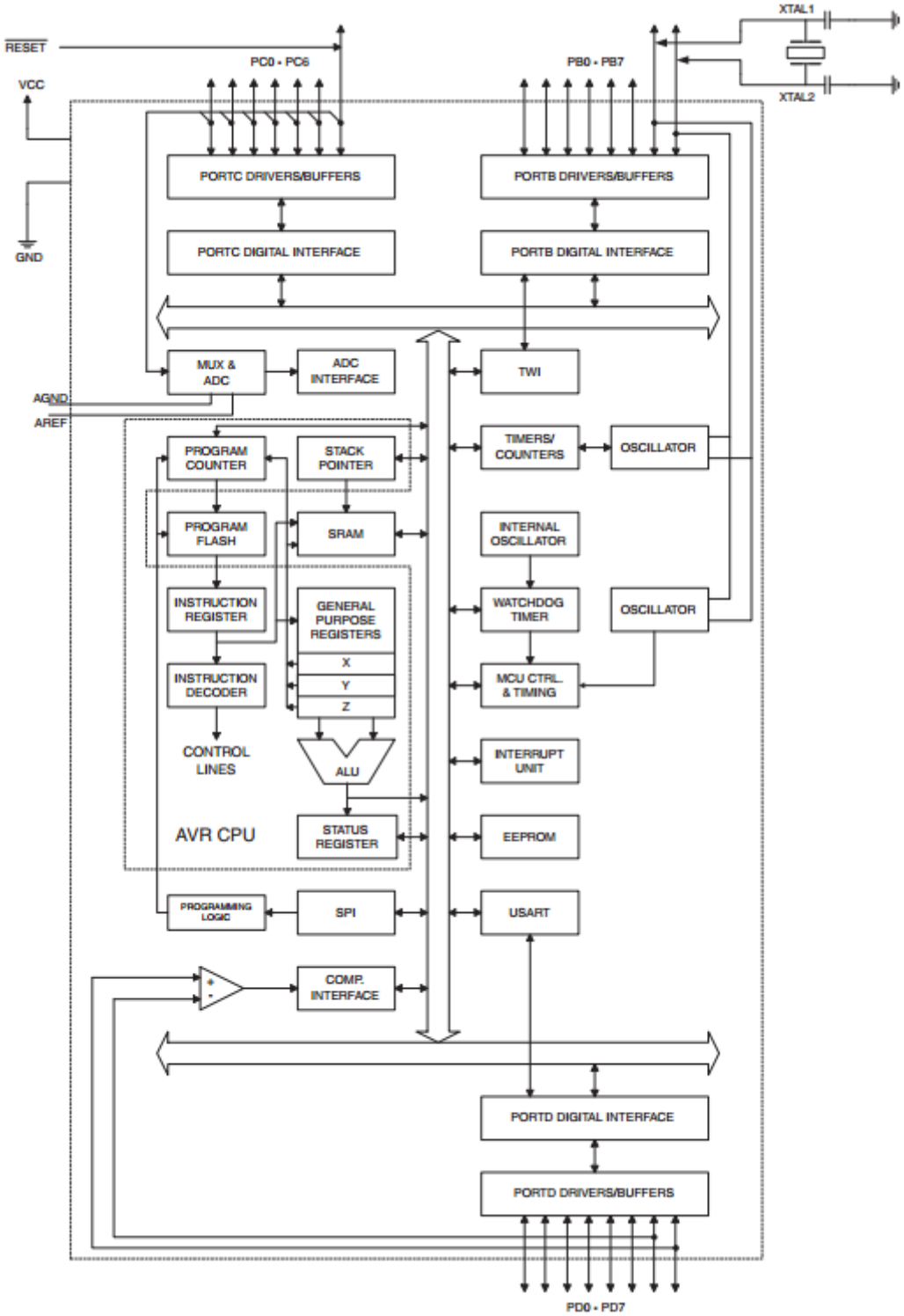


Рис. 5.4. Приклад архітектури мікроконтролера (Atmel ATmega8)

Спрощена архітектура здешевлює пристрій, тактовий генератор і таймер спрощують реалізацію оточення при конструюванні та виготовленні керованого пристрою, роблять систему більш компактною. Інтегрована ОЗП призначена для тимчасового збереження даних, а ПЗП зберігає керуючу програму. Зазвичай ПЗП у мікроконтролерах має об'єм, більший, ніж ОЗП. ПЗП сучасних пристроїв передбачає можливість перепрограмування, що дозволяє за потреби оновити версію керуючого програмного забезпечення.

На відміну від сучасних універсальних процесорів мікроконтролери є однопоточними, тобто не передбачають використання розпаралелення задачі. Це пов'язано з підходом, який спирається на максимальне наближення керуючого пристрою до виконуючого і за потреби використання у складній системі багатьох мікроконтролерів, кожен з яких відповідає за окрему дію (чи декілька дій, але з можливістю їх послідовного виконання).

Перший патент на однористальну обчислювальну машину, фактично мікроконтролер, отримали співробітники Texas Instruments М. Кочрен та Г. Бун. Саме тоді була запропонована ідея додати на кристал до процесору пристрої запам'ятовування і вводу-виводу. У найближчі роки до виробництва мікроконтролерів долучились також Intel та Motorola. Серед найбільш відомих сучасних виробників можна додати такі відомі фірми, як Atmel, Toshiba, Microchip Technology, Altera та інші.

Обмеженість обчислювальних ресурсів мікроконтролера дещо ускладнює розробку програмного забезпечення для нього. У даному випадку значною мірою є критичним мінімізація розміру коду та потреб в оперативній пам'яті. Подекуди можуть виникати потребу у якісній оптимізації швидкості виконання. З іншого боку обмеженість ресурсів зазвичай робить неможливим реалізацію програмних засобів розробки (редактору, транслятору та відгадчика) безпосередньо на мікроконтролері. Відповідно або доведеться програмувати безпосередньо у кодах системи команд, що надто незручно, або використовувати додаткові програмні засоби на іншій, більш потужній та універсальній, обчислювальній системі. Таким чином, розробка тексту програми та його компіляція зазвичай виконуються на універсальному комп'ютері, після чого можна виконати запис кодів в ПЗП мікроконтролера. Компіляція на пристрої з одною системою команд для пристрою з іншою системою команд має назву ***крос-компіляція***. Найбільш популярною мовою для програмування мікроконтролерів зараз є С++. Крім цього використовується також асемблер. Для зручності відладки використовується ***емулятор***

системи команд мікроконтролера на комп'ютері, на якому ведеться розробка та трансляція.

5.2. Система пам'яті сучасного комп'ютера

5.2.1. Енергозалежна пам'ять

Довільний програмований пристрій обробки інформації (комп'ютер або мікроконтролер) вимагає тимчасового збереження проміжних результатів розрахунків. Крім того для комп'ютера потрібно:

- завантаження програми і даних після вмикання живлення пристрою;
- запис оброблених даних перед вимиканням живлення;

Тимчасове збереження даних забезпечує оперативний запам'ятовуючий пристрій, для роботи якого може використовуватись живлення (***енергозалежна пам'ять***). Інші потреби вимагають використання пристрою, який протягом достатньо довгого часу може зберігати стан після вимикання живлення. Відповідно цей вид пам'яті отримав назву ***енергонезалежної***. Зазвичай енерго незалежна пам'ять є значно повільнішою за енергозалежну.

Один з пристроїв для збереження проміжні результатів був розглянутий у п. 5.1.1 – регістр. Регістри виконуються на кристалі процесора по технології, що відповідає іншим пристроям процесора. Відповідно, вони мають аналогічну швидкодію. Але така пам'ять, ідеальна для взаємодії з АЛП, придатна для збереження надзвичайно обмеженої кількості даних. Таким чином для реалізації пам'яті системи обробки даних потрібно реалізувати значно більший об'єм з прийнятними характеристиками, до яких можна віднести:

- швидкодію;
- надійність;
- ціну;
- мініатюрність;
- мінімальну потужність енергоживлення.

Зрозуміло, що ідеальним можна вважати пристрій пам'яті, що максимально вдовольняє усім перерахованим параметрам одразу, проте у реальній ситуації доводиться шукати деякі компромісні рішення. Наприклад пам'ять з прийнятною швидкодією має гірші показники за ціною та мініатюрністю, що вимагає обмежити її об'єм для наявної технології виготовлення мікросхеми. Рішенням такого

протиріччя є ***ієрархічна побудова підсистеми пам'яті, тобто кожен наступний рівень пам'яті є повільнішим, але більшого об'єму***.

Ефективність ієрархічної будови пам'яті пов'язана з тим, як саме використовуються пам'ять загалом. На сучасних системах обробки даних як правило виконуються одночасно декілька задач. В даному випадку під одночасністю маємо на увазі виконання наступної задачі без завершення раніше запущених. Задачі можуть виконуватись дійсно одночасно, при умові наявності декількох виконуючих пристроїв (процесорів або ядер), але зазвичай використовується і послідовне перемикання задач у часі, оскільки зазвичай кількість задач як правило значно перевищує кількість ядер.

Сучасні операційні системи дозволяють розв'язати таку проблеми за рахунок виділення ресурсів для даної задачі протягом певного невеликого кванту часу. Тобто усі задачі, які повинні виконуватись у даний час, по черзі отримують можливість бути виконаними. Оскільки квант часу є досить коротким, таке почергове виконання декількох задач навіть при умові тільки одного ядра сприймається користувачем як одночасне (паралельне).

При виконанні багатьох задач кожна з них вимагає своєї області пам'яті для тимчасового використання. Крім того перемикання з однієї задачі на іншу повинне супроводжуватись резервуванням стану задачі, що призупиняється, та відновленням стану задачі, що відновлюється. Деякі задачі самі по собі вимагають дуже великих обсягів пам'яті. Таким чином об'єму швидкої пам'яті в системі обробки інформації може не вистачити. В цьому випадку саме ієрархічна організація пам'яті допомагає розв'язати проблему шляхом витіснення частини даних з більш швидкої пам'яті у менш швидку. Зрозуміло, що повинні бути витіснені ті дані, які є найменш потрібними у даний момент часу.

Відповідно, ієрархічна побудова пам'яті сучасного комп'ютера включає такі рівні:

- регістри;
- кеш пам'ять (зазвичай дво-трирівнева);
- оперативний запам'ятовуючий пристрій (ОЗП);
- зовнішня (енергонезалежна) пам'ять.

Розглянемо детальніше задачу створення великого масиву пам'яті, що має прийнятні для експлуатації характеристики. Зазвичай при цьому треба забезпечити:

- технологію збереження одного біту;
- створення масиву однобітових елементів з ефективним доступом.

Одразу зауважимо, що це стосується усіх перерахованих рівнів пам'яті.

Для збереження біта в енергозалежній пам'яті на поточний момент використовують дві основні технології:

- статична пам'ять (англійською мовою SRAM);
- динамічна пам'ять (англійською – DRAM).

Носієм в першому випадку є ***тригер – електронна логічна схема, яка має два стійкі стани, в яких вона може перебувати доки не зміняться відповідним чином сигнали керування.*** Зазвичай реалізація одного тригера, що зберігає один біт, вимагає використання 4-6 ключів (транзисторів), рис. 5.5.

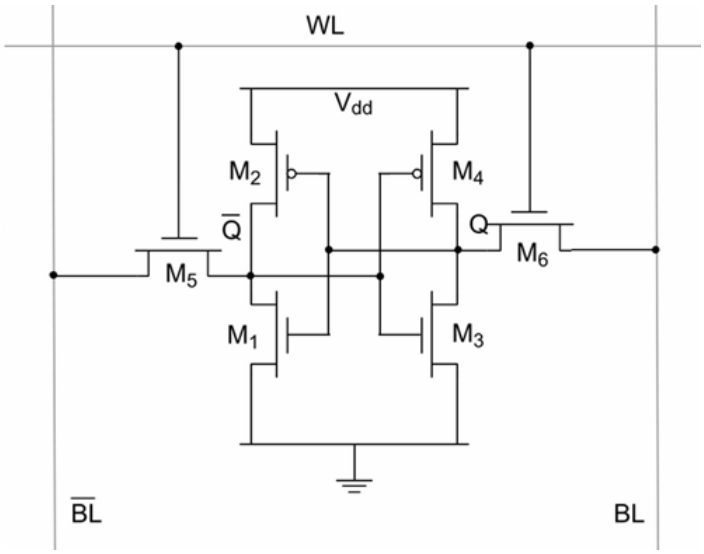


Рис. 5.5. Схема тригера

Носієм біта динамічної пам'яті є конденсатор (рис. 5.6), у якому наявність або відсутність заряду і кодує значення біта. DRAM має потребу 1-2 ключів на біт, що робить її дешевшою, компактнішою (при збереженні технології виготовлення напівпровідниковий кристал вміщує в 2–3 рази більший об'єм пам'яті). Але при зазначених перевагах така пам'ять має і певні недоліки, основними з яких є :

- конденсатор має саморозряд;
- зчитування стану комірки знищує її зміст;
- менша швидкодія.

Для зчитування стану конденсатора виконується його розряд, тобто реєстрація струму за наявності заряду (відповідного стану біта). Оскільки при цьому зміст комірки знищується, після зчитування її треба перезаписати. Вже через подібну особливість комірки динамічної пам'яті стають зрозумілими причини її меншої швидкодії.

До цього також треба додати додаткові частотні обмеження сталої часу RC-комірки. Крім того саморозряд конденсатора призводить до потреби регенерації, тобто потреби перезапису стану через деякий час.

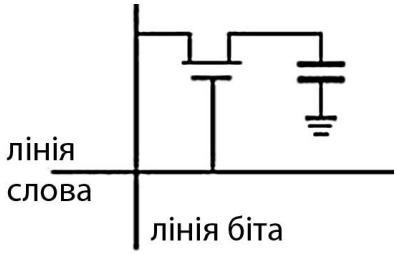


Рис. 5.6. Схема бітової комірки DRAM

Певні переваги DRAM визначають те, що ОЗП зазвичай виконується саме за такою технологією, але для компенсації її недостатньої швидкодії додатково використовується кеш за технологією SRAM. Для збільшення ефективності роботи кеш робиться багаторівневим, тобто більша частина об’єм кешу другого рівня (L2) робиться дещо повільнішим за перший (L1), але за рахунок цього загальний об’єм кешу може бути збільшеним.

Масив комірок робиться у вигляді двовимірної структури рядків та колонок (рис. 5.7). Завдання адреси рядка та колонки визначають адресу вибірки даних (рис. 5.8). Зрозуміло, що така вибірка з однієї матриці буде відповідати одному біту. Вибіркам потрібного розміру (за кількістю бітів) відповідає набір відповідної кількості таких двовимірних матриць.

Загалом швидкодія ядра пам’яті визначається сумою часу обробки адреси рядка (його передача супроводжується синхросигналом RAS), колонки (супроводжується CAS) та часом отримання даних, що також призводить до деякої затримки. Зазначимо, що у сучасних пристроях, оскільки програми використовують зазвичай не одне слово, а блок даних, реалізується пакетний режим вибірки із буферизацією при отриманні першого слова пакету декількох наступних слів в статичному буфері, що робить доступ до наступних слів пакету значно швидшим.

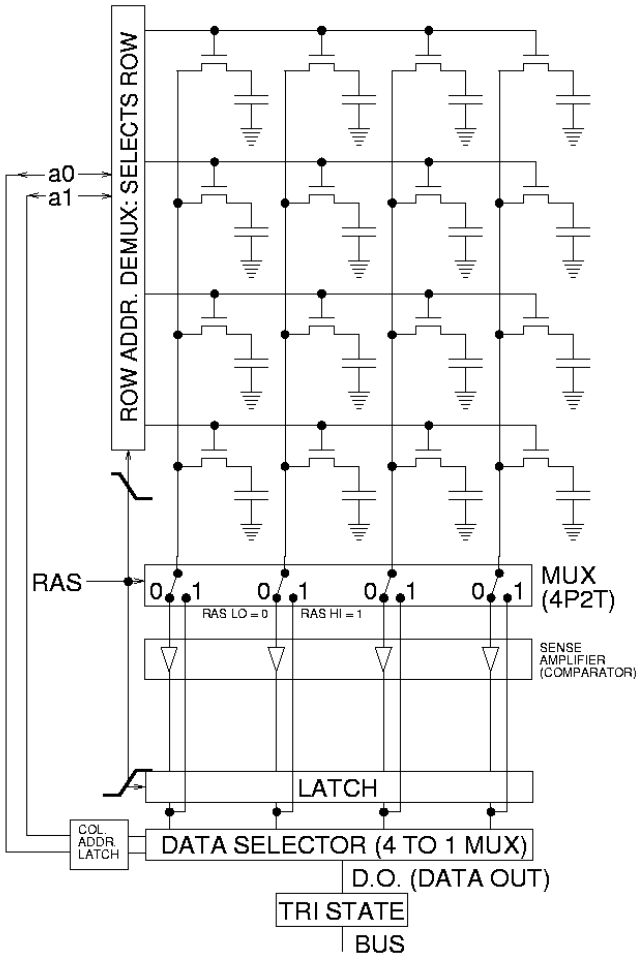


Рис. 5.7 Структура масиву бітових комірок DRAM

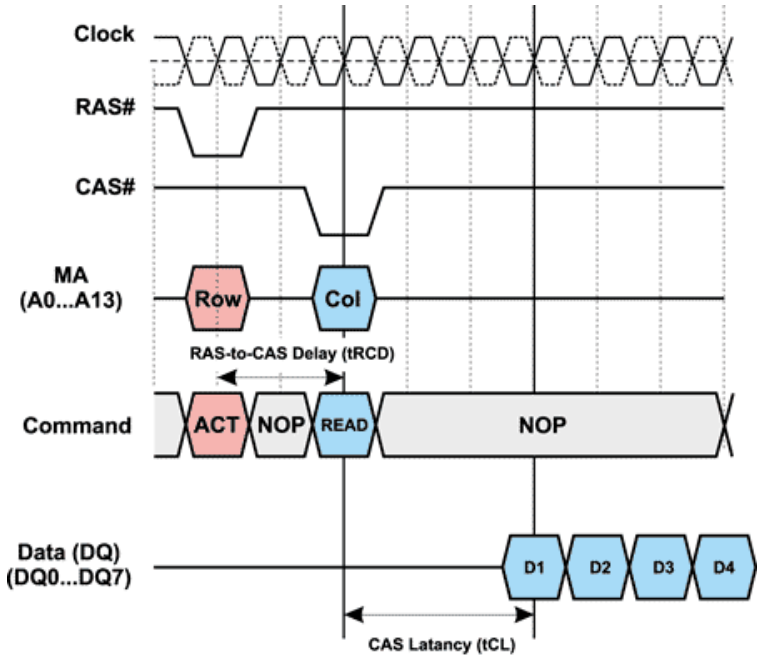


Рис. 5.8. Діаграма доступу до пам’яті

Блоки пам’яті, виконані за різними технологіями, мають деякі відмінності у зазначених часах, що вимагає певної обережності при виборі модулів пам’яті при комплектуванні комп’ютера. Слід також зауважити, що часто у комп’ютерах використовують подвійну вибірку, що вимагає використання узгодженої пари модулів ОЗП.

За рахунок динамічної реалізації ОЗП робота комп’ютера може значною мірою пригальмовуватись. Загальний принцип вирішення проблеми узгодження швидкого та повільного пристрою полягає у буферизації (кешування) частини даних у швидкому проміжному пристрої. Розглянемо роботу цієї ідеї на прикладі однорівневої кеш-пам’яті. Відповідно маємо великий за об’ємом ОЗП на основі повільної DRAM та кеш меншого розміру, але на основі більш швидкої SRAM.

При виконанні довільної програми з ОЗП завантажуються інструкції (команди) та дані для обробки (розрахунків). Певна частина потоку команд виконується одноразово. Це стосується і частини даних. Навіть у цьому випадку буферизація в кеш (при використанні блочного режиму передачі даних) може допомогти за рахунок попередньої підготовки наступних команд і даних тоді, коли немає звернень з боку процесора до повільного пристрою (ОЗП). Зазвичай така буферизація у сучасних системах реалізується на рівні самої DRAM (вона розглядалась раніше) і кеш-пам’ять практично нічого не покращить. Але у довільній програмі існує значна частина команд, які виконуються багато разів, наприклад деяка сукупність однакових дій з набором даних. Такі фрагменти програм реалізуються галуженнями, циклами, багаторазовим викликом функцій. Відповідно буферизація такого фрагмента програми у швидкий кеш дозволяє повторне звернення реалізувати до кешу замість більш повільного ОЗП. Буферизація (дублювання) повинна відбуватись кожного разу, коли дані читаються з ОЗП, або записуються до нього.

Як приклад, можна навести розрахунок пікселів кадру, при розрахунку яких значною мірою використовуються ті самі дані. Іншим прикладом може бути ітераційний розрахунковий процес, коли масив результатів розрахунку попередньої ітерації використовується як вхідні дані наступної.

Той факт, що найчастіше використовується відносно невеликий відсоток команд і даних саме і призводить до того, що найкращим для більшості задач є використання у системі обробки даних декількох рівнів кешу (наступній рівень повільніший за попередній, але значно більшого розміру). Відповідно у сучасних процесорах реалізуються до трьох рівнів. Зазначимо, що ефективність кешу з врахуванням співвідношення характеристик його рівнів може проявлятись різним

чином на різних класах задач, тобто ідеального універсального розподілу по рівнях не існує.

При проектуванні кешу доводиться вирішувати дві проблеми, надважливі для забезпечення високої ефективності:

- які дані витісняти з кешу при потребі запису у нього;
- коли дублювати до ОЗП дані, оброблені процесором, що заносяться до кешу.

Проблема потреби витіснення витікає з того, що розмір кешу у багато разів менший за розмір ОЗП, яку він буферує, тобто доводиться деяким чином проеціювати деякий адресний простір (кількість комірок) великої ОЗП на значно менший об'єм кешу. Саме спосіб відображення адрес буферованого ОЗП у адресу в кеші визначає архітектурні особливості реалізації кешу.

За аналогією з пакетним доступом до пам'яті кеш зазвичай реалізується також у вигляді деякої сукупності блоків-рядків. Одним з простих способів витіснення з кешу найменш потрібного блоку даних є пошук блоку, до якого найдовше не було звернення. Це не означає, що до нього не буде звернення найближчим часом, але більш якісного і одночасно простого у реалізації передбачення наступної корисності не існує. Відповідно до цієї ідеології слід забезпечити можливість запису довільного блоку з ОЗП у довільний рядок кешу. В цьому випадку молодші (три) біти адреси ОЗП складають так званий ***зсув*** – номер стовпця у рядку кешу. Інші розряди цієї адреси утворюють значення додаткової комірки рядка – ***тегу***. Кеш з такою внутрішньою будовою називається ***повністю асоціативним*** (англійською – ***fully associative cache***).

Основною проблемою повністю асоціативного кешу є необхідність реалізації апаратної перевірки відповідності потрібного значення адреси значенню тегу у всіх рядках. Альтернативним рішенням, яке не має подібної проблеми, є ***кеш прямого відображення*** (англійською ***direct mapped cache***). Така реалізація кешу відрізняється тим, що кількість бінарних розрядів тегу скорочується за рахунок включення на їх місце номера рядка. Це означає, що кожному з рядків кешу відповідає певний набір конкретних блоків ОЗП, які можуть бути буферовані саме у цьому рядку. Ефективність заміщення при цьому значно зменшується, але спрощується апаратна система керування кеш-пам'яттю.

Компромісом між описаними архітектурами є n-входовий (або n-канальний) частково-асоціативний (або наборно-асоціативний) кеш. В цьому випадку рядки ділять на набори по n рядків. Номер набору в адресі ОЗП використовується замість рядка. При цьому блок даних з

ОЗП може витіснити не конкретний рядок а один з декількох (у певному наборі).

Вибір одного з зазначених варіантів розв'язує задачу – що саме витісняти з кешу при записі до нього. Залишилось визначитись коли саме дублювати дані до ОЗП. Є декілька варіантів визначити політику витіснення:

- наскрізний запис (write through) – запис до ОЗП одночасно з кеш;
- буферований наскрізний запис (buffered write through) – запис в ОЗП протягом циклів (тактового генератора) коли немає звернення процессору до ОЗП;
- Зворотний запис (write back) запис до ОЗП тільки при потребі звільнення рядка кешу, або цього вимагає узгодження стану пам'яті для паралельних обчислень.

5.2.2. Енергонезалежна пам'ять

Енергонезалежна пам'ять зазвичай реалізується як самостійний пристрій, у якому зберігаються певний об'єм даних, або як пристрій для роботи зі змінними носіями – дисками, стрічками, картами. Для реалізації енергонезалежної пам'яті потрібно розв'язати дві основні задачі:

- спосіб довгострокового збереження біту;
- фізичну та логічну організація бітових комірок у носій або пристрій.

При цьому необхідно оптимізувати носій або пристрій по таким характеристикам, які значною мірою вступають у протиріччя одна з одною:

- інформаційний об'єм;
- накладні витрати (ціна збереження певного об'єму даних);
- розмір носія або пристрою;
- надійність збереження;
- зручність користування.

Почнемо розгляд з **способів збереження біту**, які використовуються у сучасних комп'ютерних технологіях. Для збереження даних використовується стабільний просторовий сигнал, тобто залежна від просторових координат фізична величина певної природи. Обирається така природа фізичної величини, для якої можна забезпечити зручну для технологічної реалізації зміну стану та його зчитування. При цьому при прийнятних експлуатаційних умовах стан ділянки запису біта не повинен змінюватись самостійно. Принаймні

ймовірність такої зміни не повинна перевищувати значення, для якого викривлені дані можна виправити прийнятними методами обробки інформації.

Одним з традиційних способів такого просторового сигналу є намагніченість тонкого феромагнітного шару, ділянки якого, що відповідають одному біту, намагнічуються незалежно. Для запису (намагнічування) та зчитування (при цьому стан комірки не змінюється) використовується таж сама магнітна голівка. Виключенням є магнітооптичний спосіб запису, коли магнітна голівка використовується тільки для запису. Такий спосіб запису даних добре себе зарекомендував у аналогових стрічковий пристроях – магнітофонах ще задовго до широкого застосування у комп'ютерних технологіях. Його перевагами є відносна простота та технологічність реалізації носія, компактність, прийнятна надійність. Саме через це основними засобами для реалізації енергонезалежної пам'яті сучасних комп'ютерів є ***жорсткі диски*** (англійською hard disk drive, HDD). Досить довгий час в комп'ютерах також використовувались змінні носії невеликого об'єму – ***дискети*** (floppy disk). Такий спосіб запису використовується і в стрічкових пристроях резервного копіювання (п. 5.2.3) інформаціях – ***стримерах*** (streamer).

Магнітний спосіб збереження даних має деякі недоліки, а саме:

- можливість ушкодження даних зовнішнім потужним джерелом магнітного поля;
- потребу для реалізації пристрою в механічних елементах, що рухаються з великою швидкістю.

Серед усіх технологій реалізації енергонезалежної пам'яті першу з цих проблем було вирішено у такому носії як ***магнітооптичний диск***. Друга проблема відсутня тільки в одному з сучасних накопичувачів – ***твердотільному накопичувачі*** (англійською solid-state drive, ***SSD***) та його попереднику – флеш-пам'яті, яка частіше за все виконує функції змінного носія та постійної пам'яті у сучасних цифрових пристроях.

Легкість перемагнічування феромагнітного шару, що є обов'язковою умовою реалізації традиційного магнітного запису, передбачає використання феромагнетика з температурою Кюрі ненабагато більшу за температуру експлуатації носія (кімнатну). Нагадаємо, що при нагріві до температури Кюрі феромагнетик переходить у парамагнітний стан і втрачає можливість «запам'ятовування». При зменшенні температури чим далі робоча температура від точки Кюрі, тим сильніше магнітне поле потрібне для перемагнічування. Цю особливість використовує магнітооптичний запис, при якому перемагнічування відбувається магнітною голівкою,

але перед цим лазер нагріває цю ділянку до достатньо високої для легкого перемагнічування температури. Після зменшення температури зони запису до кімнатної комірка переходить у стан, який майже неможливо змінити магнітним полем без нагріву. Для зчитування даних у такому носії використовується ефект обертання площини поляризація світла магнітним полем (ефект Фарадея). Для цього використовується той же лазер, але на значно меншому рівні потужності.

Наступним способом запису біта є зміна коефіцієнта відбиття поверхні або механічним порушенням дзеркального шару, що відбиває світло, або за рахунок зміни стану з кристалічного на аморфний робочого шару, а саме плівки з спеціального сплаву халькогенідів, що супроводжується зміною оптичних властивостей. Ці технології дозволили реалізувати такий клас носіїв, як оптичні диски різних поколінь.

У SSD та флеш-пам’яті для запису біту використовується стабільна зміна стану каналу МДП транзистора (рис. 5.9). Спочатку згадаємо принцип дії звичайного МДП транзистора з індукованим каналом. Між сильно легованими ділянками – істоком та стоком існує проміжок напівпровідника з надмалою кількістю власних носіїв. Якщо напруга на затворі, ізольованому від іншої частини транзистора тонким шаром окислу, перевищує деякий поріг, у цьому збідненому носіями шарі індукуються носії, тобто між стоком та істоком створюється провідний канал. У комірці збереження біту є два затвори – перший так званий плаваючий, тобто електрично відірваний від інших електродів приладу, а другий – керуючий, призначенням якого є зміна заряду плаваючого затвора.

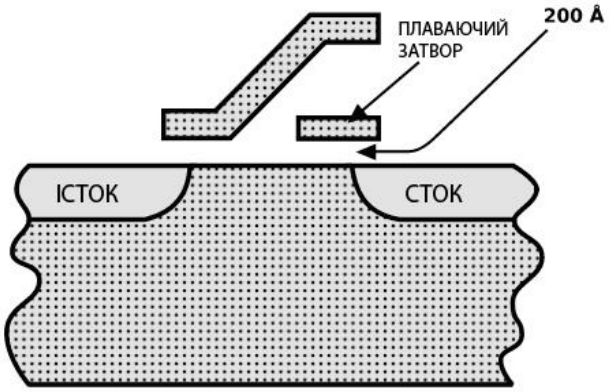


Рис. 5.9. Бітова комірка флеш-пам'яті

Для запису інформації потрібно зарядити плаваючий затвор, для чого на керуючий затвор транзистора подається достатня напруга та пропускається струм через канал. При цьому виникають електрони з

енергією, достатньою для тунелювання у пастку на ізольований затвор. Для стирання інформації між керуючим затвором та стоком прикладають значну напругу протилежної, ніж при запису, полярності, яка створює сильне поле, яке висмоктує носії з плаваючого затвору (знов тунелюванням). Зчитування інформації відбувається реєстрацією наявності каналу у комірці. Є комірки, у яких записується більша кількість бітів за рахунок варіювання заряду на плаваючому затворі та вимірювання струму через канал при зчитуванні.

Збільшення ємності носія з розвитком технології зазвичай визначається можливістю зменшити ділянку, яка відповідає за збереження одного біта при збереженні достатньої надійності вірного розпізнавання записаного сигналу.

Фізична організація накопичувача повинна забезпечити доступ до достатньо великої кількості бітових комірок. Для забезпечення зручності реалізації пристрою та надійності його роботи потік даних реалізується як послідовність блоків наперед заданого обсягу – ***секторів***, певна кількість яких і є носієм. Відповідно, якщо блок даних є більшим за сектор, його дані будуть займати декілька повністю зайнятих секторів та один частково вільний. Зауважимо, що на рівні файлової системи комп'ютера, за яку відповідає операційна система, логіка «неподільних ділянок» потоку даних продовжує свій розвиток за рахунок кластеризації, тобто об'єднання (з настроюванням розміру кластеру) декількох секторів.

При реалізації пристрою зазвичай реалізується:

- послідовний доступ (доступ до елементів пам'яті накопичувача тільки у заданому порядку);
- довільний доступ (швидкий доступ до довільного за адресою елементу пам'яті накопичувача).

Стрічкові та дискові пристрої вимагають механічного суміщення ділянки носія, де у поточній момент відбувається запис або читання з пристроєм (голівкою) запису/читання. Найпростіше таке суміщення забезпечується одновимірним рухом при використанні стрічкового носія (магнітофон, стример), але це відповідає досить повільному послідовному методу доступу до даних. Дискові накопичувачі більш гнучкі до вибору «точки входу» у необхідний блок даних, але і в цьому випадку все одно існує помітний час очікування, пов'язаний з механічним рухом. На диску реалізується або спіральна доріжка (оптичні диски), або набір концентричних кругових доріжок (магнітні диски). Адреса доступу при цьому визначається радіусом позиціонування голівки відносно шпинделя диску, кутом (від умовного нуля до даного сектору), поверхнею, якщо

таких робочих поверхонь декілька (у HDD можуть використовуються обидві поверхні диску або взагалі декілька дисків на спільному шпинделі). Для вибору поверхні використовується:

- для кожної з них окрема голівка (HDD);
- перевертання диску (двобічний DVD);
- перефокусування з одного шару на інший для (двошаровий DVD, який крім основного непрозорого шару над ним має додатковий прозорий).

Для HDD з метою забезпечення максимальної швидкості запису та читання в першу чергу варіюється кут в межах доріжки (змінюється номер сектору), при завершенні одної доріжки на поверхні змінюється робоча поверхня (перемикання голівки є доволі швидкою електронною комутацією без механічного зсуву), долі змінюється (механічним рухом блоку головок) доріжка.

У випадку двошарового DVD зміна поверхні відбувається після запису або читання повної спіральної доріжки, оскільки в даному випадку перемикання є доволі повільним.

Пристрій може бути реалізований або з незмінним носієм (HDD), або з можливістю заміни диску. Це дозволяє збільшити формальний обсяг даного накопичувача при недостатній для робочих потреб ємності одного диска та використовувати такий пристрій накопичення для дублювання даних та їх перенесення між комп'ютерами.

Розглянемо детальніше конструкцію магнітного накопичувача HDD, рис. 5.10. Як вже зазначено, загальна структура пристрою спирається на незмінний носій, який складається з одного або декількох дисків на спільному шпинделі. Диски є алюмінієвими або скляними. Кожен має одну чи дів робочі поверхні за рахунок напиленого тонкого феромагнітного шару (наприклад з окису хрому). Кожна робоча поверхня (як запис, так і читання) обслуговується окремою голівкою, які зібрані у єдиний блок, поворот якого перепозиціонує усі голівки блоку на іншу радіальну доріжку (циліндр). Голівки не торкаються поверхні диску (відстань – декілька нм), що забезпечує довгий час життя накопичувача та дозволяє використовувати велику швидкість обертання (до 10 тис. обертів на хвилину і навіть більше для серверних дисків). Оскільки безпечний зазор забезпечується аеродинамічним шляхом, при зупинці обертання зазвичай забезпечується виведення головок у безпечну зону (паркування). Велика швидкість обертання та малий зазор вимагають високої чистоти робочої зони, що забезпечується спеціальною гермозоною, саме через яку і виникає обмеження реалізації цього пристрою без можливості заміни носія.

Рис. 5.10. Накопичувач на магнітному диску (HDD)

Для реалізації логічної організації пристрою та узгодження з іншими пристроями комп'ютера у пристрої використовується контролер. За рахунок використання секторної організації носія контролер узгоджує власну внутрішню адресацію диску з адресацією за стандартом протоколу зв'язку з диском. За потреби контролер може замінити ушкоджений сектор переадресацією на сектор з резервної зони.

Оскільки загалом пристрій є повільним, для узгодження з більш швидкими пристроями комп'ютера, використовується кешування потоку даних в електронній пам'яті. Зрозуміло, що чим більший розмір кешу (буферу) дискового пристрою, тим менше впливає недостатня швидкість даного носія.

Прикладом пристрою зі змінним носієм, який активно (поки-що) використовується є оптичний диск. Збереження даних забезпечує спіральна доріжка на поверхні прозорого полікабонатного диску. Значення біту забезпечується зміною на деякій ділянці (англійською Pit, тобто яма) коефіцієнту відбиття світла за рахунок порушення цілісності металевого шару (рис. 5.11), або переходом полімерного шару від кристалічного до аморфного (носії з перезаписом).

Зміна поколінь (CD, DVD, Blu-Ray Disc) за основну мету ставила збільшення об'єму носія. Для цього було використано зменшення довжини хвилі від інфрачервоного (CD), до червоного (DVD), а пізніше до синього (Blu-Ray Disc). Для DVD було також запропонована опція двобічного запису (але такий диск вимагає ручного перевстановлення на інший бік, що зводить нанівець його переваги), двошарового запису (з додатковим напівпрозорим робочим шаром). Відповідно, в стандарті наступного покоління залишили двошаровий запис, але відмовились від двобічного.

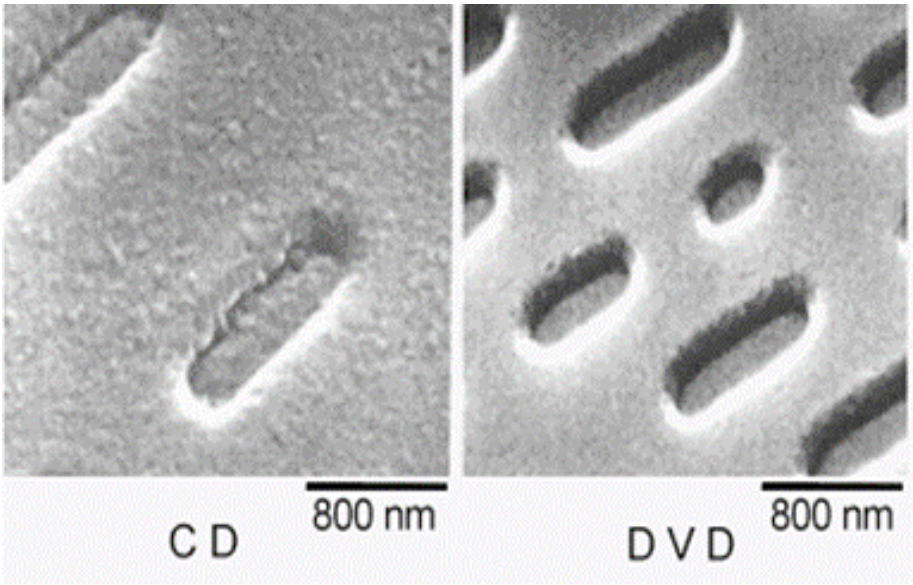


Рис. 5.11. Запис даних на різних поколіннях оптичного диску

Зауважимо, що фізична надійність збереження даних на оптичному диску є недостатньою, що компенсується надлишковістю запису інформації з досить складним кодуванням. Сама це забезпечує безпроблемну експлуатацію таких дисків навіть після помітних ушкоджень поверхні.

У випадку електронної бітової комірки (SSD, flash) для реалізації довільного доступу може бути реалізована матриця елементів пам'яті з адресацією по рядку та стовпцю (аналогічно організації RAM). Ця особливість забезпечує ще одну перевагу у порівнянні з стрічковими та дисковими накопичувачами крім відсутності механічного руху.

5.2.3. Забезпечення надійності збереження даних

У більшості систем обробки інформації основним пристроєм реалізації потреб у енергонезалежній пам'яті є HDD або SSD. Інші носії використовуються у випадках, коли виникає потреба перенести дані з одного пристрою на інший при відсутності мережевого сполучення, або для створення резервної копії. Вибір, які саме з розглянутих носіїв використовувати, є достатньо неоднозначним. До цього слід додати тенденцію до вилучення з використання дискових пристроїв зі змінними носіями, замість яких використовується флеш-пам'ять у формі карти або окремого пристрою, який приєднується через стандартний порт (USB). Як змінній носій великого об'єму

часто використовується також зовнішній HDD, тобто окремий пристрій, що складається з самого HDD та інтерфейсного модулю для приєднання до комп'ютера через USB-порт.

Якщо порівняти загальні характеристики HDD та SSD, останній є більш швидким, не має рухомих деталей і через це значно менш чутливий до ударів, як і інші електронні складові комп'ютера, легший і компактніший за конкурента, має значно менше енергоспоживання. Основним недоліком даного пристрою є обмежена кількість циклів (близько 10000) перезапису значення у бітовую комірку. Але сучасні системи підтримують «ковзний запис», тобто постійно змінюють фізичну адресу комірки для запису так, що кількість записів по комірках усього масиву пам'яті розподіляється приблизно рівномірно, що забезпечує прийнятний час життя носія в реальних умовах експлуатації. Таким чином, заміна у комп'ютері HDD на SSD забезпечує значно більш швидке завантаження програм та даних до них (досить помітно у випадку, коли програма обробляє великі об'єми інформації), прискорює розрахунки при значній потребі витіснення даних з ОЗП на диск (відчутно в деяких задачах моделювання). Розплата за такий приріст швидкодії визначається ціною мегабайту, яка на поточний момент для SSD в декілька разів більша. Але є підстава сподіватись, що з часом цей розрив буде скорочуватись.

Боротьба за швидкість доступу до даних та прийнятний рівень накладних витрат збереження даних не вичерпують задачу реалізації енергонезалежної пам'яті. Більш детальний розгляд надійності збереження інформації почнемо з тези, що досить часто ***ціна самої інформації значною мірою перевищує ціну пристроїв для її збереження*** і тому забезпечення надійності збереження даних є більш доцільним, ніж спроба зекономити на ціні накопичувачів або носіїв.

Першим кроком із запобігання втратам інформації є якомога повне дублювання важливих даних, які отримуються в процесі роботи інформаційної системи, в енергонезалежну пам'ять, навіть у випадках, коли використовується система резервного живлення, що запобігає втратам у випадку аварійних ситуацій безпосередньо у системі обробки. Але збереження одного екземпляру блоку даних також не забезпечує достатню надійність їх збереження. Єдиним способом забезпечення надійності є надлишковість даних, тобто створення таких додаткових даних, використанням яких можна відновити втрачену в результаті аварії інформацію. Зауважимо, що абсолютно надійного варіанту і в такому варіанті не може існувати, оскільки усі додаткові засоби також мають певну ймовірність виходу з ладу. Тому зазвичай іде мова про забезпечення певного, достатньо низького рівня ймовірності втрати інформації.

Найпростішим способом реалізації надлишковості деякого блоку даних для запобігання втратам інформації є їх дублюванн, яке може відбуватись:

- на тому ж розділі того ж носія (з метою випадкового видалення або помилкової зміни даних)
- на інший розділ того ж носія (запобігає аваріям файлової системи)
- на інший носій або пристрій того ж комп'ютера (запобігає аварії пристрою накопичення даних)
- на інший комп'ютер або носій, що від'єднується від комп'ютера (більш надійний спосіб боротьбі з аваріями комп'ютера, в тому числі з втратами через вірусне зараження).

Подібне дублювання може виконуватись неавтоматичним чином безпосередньо автором даних (наприклад офісного документу). При роботі з великим проектом при цьому зручно робити копії версій файлу (або архіву з файлами), що в разі потреби дає змогу повернутись до розгляду одного з попередніх варіантів (наприклад, для пошуку помилки у розробці). Зрозуміло, що для зручності подальшої роботи створені версії робляться у тому ж розділі. Але навіть у цьому випадку є бажаним дублювати їх на інший комп'ютер або носій, що від'єднується.

Зауважимо, що при використанні SSD замість HDD слід пам'ятати про такі особливості:

- при помилковому видаленні файлу з HDD часто його можна відновити спеціальними утилітами, SSD через особливості логіки запису не дозволяє цього зробити;
- в разі аварійного виходу з ладу HDD є висока ймовірність повного або майже повного відновлення даних, SSD не має такої можливості.

При роботі серверів, і навіть окремих персональних систем ручний режим дублювання даних та створення версій є незручним. Цю функцію можна автоматизувати, поклавши її на програмний засіб, наприклад, інформаційну систему. Цей варіант забезпечення надійності виконується розробниками цього програмного засобу і зазвичай передбачає додаткові умови на апаратне забезпечення комп'ютера, на якому буде використовуватись даний засіб.

Є більш універсальний спосіб спеціалізованої сукупності апаратних та програмних засобів, які забезпечують аналогічний захист для всієї інформації, що накопичується, зберігається та генерується на даній системі обробки даних. Засоби, які реалізують

такий спосіб отримали назву ***засобів резервного копіювання*** (англійською мовою – ***backup***).

Засоби резервного копіювання відстежують зміну даних у системі та забезпечують один з можливих режимів дублювання:

- повне (full backup), тобто дублювання усіх даних;
- диференціальне (differential backup), копіюються файли, що були змінені після останнього повного копіювання;
- інкрементне (incremental backup), копіюються файли, що були змінені після будь-якого останнього копіювання.

Зрозуміло, що повне резервне копіювання є дуже ресурсоємним. Воно є обов'язковим, оскільки інші способи резервного копіювання так чи інакше спираються на його результат. Для зменшення загальної ресурсоємності процедури резервного копіювання повне виконується відносно рідко і планується до виконання на той час, коли система обробки даних має найменше робоче навантаження. В інші фази копіювання використовується диференціальне або інкрементне. Диференціальне має перевагу більш швидкого відновлення системи після аварії, оскільки передбачає використання останньої повної копії і однієї диференціальної, але дещо більш ресурсоємне, ніж інкрементне (для відновлення використовується остання повна копія і увесь ланцюжок інкрементних копій після неї).

Резервне копіювання може обслуговувати окремий комп'ютер, або декілька комп'ютерів через мережу. Для цього може використовуватись один з робочих комп'ютерів системи, або спеціалізований.

В якості носіїв для резервного копіювання зазвичай використовуються змінні або відокремлені носії:

- стрічка (касета) стримера;
- оптичний диск (наприклад DVD-R або DVD-RW):
- магнітний диск (з'ємний);
- дискова система окремого сервера або хмарки (для копіювання з персональної системи).

При використання стримера або оптичних дисків для збільшення надійності може бути застосована політика одноразового використання носія (запис тільки один раз), в більшості випадків реалізується один з варіантів ротації носіїв.

Резервне копіювання даних забезпечує майже повне відновлення інформації. Неповнота відновлення пов'язана з тим, що може бути втрачена інформація, яка змінювалась у період часу від останнього копіювання. Оскільки найчастіше система резервного копіювання працює вночі (раз на добу), то може бути втрата інформації останньої доби. Більш часте резервування надважливих

даних можна реалізувати окремим чином, наприклад за рахунок вище названого дублювання самим програмним засобом, однак таке рішення не є універсальним. Більш зручним і надійним є застосування окремого засобу, який значною мірою збільшує надійність цілісності інформації майже без робочих втрат часу. Відповідно цей спосіб активно використовується на серверах та інформаційних системах великої надійності. Він спирається на використання надлишкового дискового простору і використання декількох пристроїв (HDD). Засіб, що його реалізує отримав назву ***RAID***-масив (redundant array of independent disks — надлишковий масив незалежних дисків).

Загалом RAID-масив можна визначити як сукупність декількох дисків, їх контролер та швидкі канали зв'язку, що сприймається системою як ціле. Цей масив конфігурується відповідно до потреб забезпечення надійності та підвищення швидкості доступу до даних. RAID-масив можу бути сформований із звичайних HDD у корпусі настільного персонального комп'ютера, але більш правильним чином реалізується як окремий пристрій із спеціалізованим контролером з можливістю гарячої (тобто без відключення живлення) заміни дисків. Крім того у таких пристроях використовуються більш надійні серверні HDD.

В залежності від особливостей роботи RAID-масив ділиться на різні варіанти, які називаються рівнями:

RAID 0 – (striping, чергування), блоки даних з потоку записуються на два чи більше дисків по черзі без використання надлишковості, призначений для збільшення швидкодії дискової системи, але збільшує ймовірність втрати інформації;

RAID 1 – (mirroring, дзеркалювання), два диска, які є повними копіями один одного, при виході з ладу одного з них аварійний диск замінюється новим та відновлюється дублюванням даних з працездатного;

RAID 2 – за рахунок використання кодів Хемінга (розряди якого розподіляються по дисках, мінімально 7 накопичувачів) мають менші накладні витрати (кількості дисків), ніж RAID 1

RAID 3 – за рахунок використання замість самовідновлюваних кодів біту парності має ще менші накладні витрати і може бути реалізований з меншою кількістю накопичувачів, ніж RAID 2;

RAID 4 – схожий на RAID 3, але дані розбиваються на блоки більшого розміру, що збільшило швидкість доступу;

RAID 5 – подальша модифікація RAID 5 з розподілом контрольної функції від окремого диску на усі;

RAID 6 – є більш надійним за рахунок збільшення кількості контрольних даних.

Може також використовуватись комбінація рівнів, наприклад RAID 10 (1+0) – дзеркалювання двох масивів з чергуванням з метою забезпечити високу надійність та швидкість.

Надлишковість даних, що зберігаються у RAID-масиві, значно зменшує ймовірність аварійної втрати інформації, оскільки до втрати призводить тільки вихід з ладу ще одного накопичувача за час відновлення даних. Відповідно, ймовірність незворотної втрати інформації визначається добутком ймовірностей виходу з ладу окремих дисків та множенням на співвідношення часу відновлення та часу життя диску (останній множник додатково зменшує цю ймовірність на декілька порядків).

5.3. Засоби передачі даних

5.3.1. Загальні принципи побудови інтерфейса

Забезпечення високої швидкості пристрою обробки даних (процесора) та достатньої швидкодії та надійності пристроїв збереження даних (оперативної та зовнішньої пам'яті) є необхідною, але не достатньою умовою для створення високоефективної системи обробки інформації. Це пов'язано з тим, що така система є багатокомпонентною і загальна її ефективність залежить не тільки від складових, а й від ефективності (швидкості та надійності) передачі даних між компонентами. За цю задачу відповідають інтерфейсні засоби.

Загалом ***інтерфейс*** (англійською мовою ***interface*** — поверхня розділу, перегородка) — сукупність засобів і методів взаємодії між елементами системи. Загалом ці засоби можуть бути як апаратними, так і програмними. Відповідно:

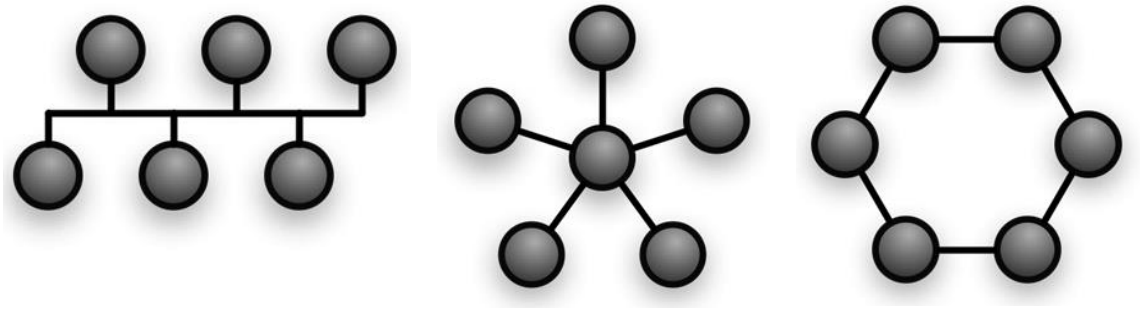
- ***Апаратний*** (фізичний) інтерфейс — сукупність апаратних і конструктивних засобів, необхідних для реалізації взаємодії;
- ***Програмний*** (логічний) інтерфейс — сукупність програмних засобів, необхідних для взаємодії різних елементів інформаційних систем, що на потребують спеціального апаратного забезпечення.

Одразу зауважимо, що навіть інтерфейс, який реалізується в основному апаратними засобами, може включати у себе програмні складові, призначені для реалізації ***протоколу***, тобто порядку дій, необхідних для виконання інформаційного обміну. Часто до поняття

протоколу додають ще ***формат*** (внутрішню структуру) даних. Але більш логічним є це поняття виділяти як окреме, оскільки є протоколи, які працюють з декількома форматами даних, а є формати, які використовуються багатьма протоколами. Розглянемо основні принципи створення інтерфейсів. Оскільки у цьому розділі йде мова про апаратні засоби, основна увага буде приділена саме апаратній частині,

У загальному випадку інтерфейс повинен забезпечити передачу даних між двома чи більшою кількістю пристроїв. Відповідно при обміні даними між двома пристроями кажуть про інтерфейс точка–точка (англійською – point to point), що є найпростішим випадком. Багатоточкове з'єднання є складнішим. Воно повинно передбачити топологію фізичного з'єднання, основними варіантами якого є:

- шина (паралельне приєднання усіх пристроїв до інтерфейсу), рис. 5.12.а;
- зірка (зв'язок окремих пристроїв з виділеним центральним (точка–точка), рис. 5.12.б;
- кільце (кожен з пристроїв має з'єднання точка–точка з двома найближчими сосудами таким чином, що загальна структура зв'язків створює кільце, зазвичай з передачею даних тільки в один бік) , рис. 5.12.в.



а б в

Рис. 5.12. Основні варіанти топології з'єднання: магістраль (а), зірка (б), кільце (в)

Є й інші варіанти, такі як:

- повнозв'язне з'єднання (кожен з кожним);
- гратка (зв'язки рядками і стовпцями);
- змішана.

Фізичною основою запису та передачі інформації або даних, які є кодовим (логічним) її носієм, є сигнал (п. 1.1.1). Таким чином апаратний інтерфейс повинен спиратись на передачу від одного пристрою до іншого сигналу певної фізичної природи. Найбільш

надійним та зручним для реалізації інтерфейсів є електричні (струм, напруга) та електромагнітні хвилі радіо- та оптичного діапазонів.

Процес передачі даних передбачає розповсюдження сигналу у середовищі (лінії) передачі та внесення сигналу у це середовище з боку джерела даних та реєстрацію з боку приймача. Таким чином інтерфейс включає у себе середовище передачі, наприклад лінію передачі або їх сукупність та ***термінальне обладнання***, призначене саме для внесення та зчитування сигналу з середовища передачі. При цьому використання певного протоколу взаємодії пристроїв, що з'єднуються, вимагає певного алгоритму роботи термінальних пристроїв, який може бути реалізований апаратним або програмно-апаратним чином. Зауважимо, що тільки відносно простий протокол, або частина протоколу обміну може бути реалізована апаратно, без використання програмної складової.

При роботі інтерфейсу кажуть про джерело даних та їх приймач. При цьому є необхідним, щоб обміном даних керував в довільний момент тільки один з пристроїв, яким може бути як передавач, так і приймач. Такий пристрій називають ***керуючим***, або активним. Спроба одночасного керування інтерфейсом називається колізією (від англійського collision – зіткнення), яка призводить до втрати даних.

Оскільки передача даних, як вже було зазначено, передбачає джерело та приймач, кажуть про три основні режими взаємодії двох зв'язаних пристроїв між собою:

- ***симплексний*** (передача тільки в один бік);
- ***напівдуплексний*** (в кожен момент часу передача в один бік, але з часом напрямок може змінюватись);
- ***дуплексний*** (одночасна передача в обидва напрямки).

Основою є симплексний режим. Напівдуплекс реалізується як перемикання симплексної передачі з відповідним ускладненням термінальних пристроїв, дуплекс реалізується як пара симплексних засобів протилежної направленості.

Сумісна робота двох чи більше пристроїв в рамках єдиного протоколу передбачає потребу узгодження в часі для цих пристроїв усіх фаз протоколу, тобто ***синхронізації***, якій приділяється значна увага при розробці та використанні інтерфейсів. Порушення синхронізації, як і колізія, призводить до втрати даних.

При розробці або використанні інтерфейсу крім узгодження фаз протоколу є потреба узгодити саму апаратну частину складових інтерфейсу, тобто відповідних елементів пристроїв, що з'єднуються. Таку сумісність зазвичай розділяють на:

- логічну;

- електричну;
- механічну.

Логічна сумісність визначає призначення і структуру складових інтерфейсу. З врахуванням того, що зазвичай інерфейс зводиться до сукупності ліній передачі, то до такої сумісності відноситься кількість ліній передачі для кожної з підзадач, потрібних для передачі даних.

Електрична сумісність визначає узгодження статичних і динамічних параметрів сигналів в лініях (навантажувальна спроможність, частота, логічні рівні сигналу)

Механічна або конструктивна сумісність включає у себе узгодження конструктивних елементів в тому числі розмірів роз'ємів.

Зрозуміло, що порушення одного з зазначених типів сумісності призведе до неможливості коректної взаємодії пристроїв що узгоджуються. Порушення логічної та електричної сумісності призводять зазвичай до втрати даних (в деяких випадках можуть призводити навіть і до виходу пристроїв з ладу), невиконання умов механічної сумісності не дозволяє взагалі виконати з'єднання.

Велика різноманітність складових системи обробки інформації вимагає великої кількості інтерфейсів, кожен з яких є пристосованим для конкретної задачі та певних умов використання. Наприклад принципово різними за вимогами інтерфейсами є з'єднання материнської плати з чіпом процесора або модулем ОЗП, приєднання до дискових запам'ятовуючих пристроїв або зовнішньої периферії, зв'язування комп'ютерів у мережу.

Узгодження окремих складових загальної системи використовує принцип ***уніфікації***, тобто приведення до одноманітності. Саме при розробці інтерфейсів, особливо з врахуванням їх певної складності, інженери вперше надзвичайно гостро зіткнулись з потребою уніфікації. Юридичне оформлення уніфікації є ***стандартизацією***. Саме у стандартах різного рівня (галузевих, національних, міжнародних) викладаються принципи уніфікації інтерфейсів.

Загалом комп'ютерні інтерфейси передачі даних традиційно діляться на дві великі групи:

- послідовні;
- паралельні.

Послідовний комп'ютерний інтерфейс спирається на передачу потоку даних біт за бітом, за рахунок чого вдається мінімізувати кількість ліній передачі, які складають інтерфейс. ***Паралельний*** інтерфейс використовує значно більшу кількість ліній.

Лінія передачі загалом призначена для передачі одного аналогового сигналу. Проте за рахунок, використання частотної модуляції та ділення загального діапазону частот, у якому можливе

розповсюдження сигналу, на під діапазони, у лінії можна реалізувати декілька незалежних каналів. В описаному випадку сумарний сигнал може бути створений з декількох їх апаратним додаванням і потім розділений на окремі складові смуговими фільтрами. Такі канали називаються частотними, що є прийнятним як для аналогової, так і цифрової техніки. Іншим способом розділу на канали є використання певної логіки кодування таким чином, що деяким алгоритмом обробки даних можна розділити їх на декілька незалежних потоків (наприклад послідовною передачею фрагментів-кадрів різних потоків). В цьому випадку можна говорити про логічні канали, що відповідає випадку цифрових технологій.

Кожна лінія передачі має певні обмеження за кількістю інформації, яку можна передати за одиницю часу. Загальна пропускна здатність визначається граничною частотою (на яку впливає дисперсія, затухання в лінії та її довжина) та динамічний діапазон, який обмежується досяжним на приймачі даних співвідношенням сигнал/шум. Розбиття на частотні канали не дозволяє збільшити цю пропускну здатність, оскільки вона стосується сумарного широкосмужного сигналу, тобто загальна пропускна здатність ділиться між каналами. Більше того, за рахунок неідеальності смугових фільтрів загальна пропускна здатність навіть зменшується із зростанням кількості каналів.

Зазначене природне обмеження стосується аналогового випадку. Для цифрових сигналів швидкість передачі є навіть меншою. Це пов’язано з тим, що фактично цифровим сигналом є деякий аналоговий сигнал спеціальної форми, на яку накладаються обмеження з метою забезпечення надійної інтерпретації даних приймачем. Цифровий режим використання ліній передачі найчастіше використовує імпульсне дворівневе кодування (низький та високий рівні, які за попередньою домовленістю відповідають значенням біта), тобто один біт кодується одним – двома прямокутними імпульсами. Таким чином динамічний діапазон частіше за все зводиться до двох рівнів, що на 2–3 порядки менше за досяжний, а потреба забезпечення якості фронту прямокутного імпульсу зменшує граничну частоту майже на порядок.

Приклади різних видів кодування бітового потоку наведені на рис. 5.13.

В деяких випадках з метою забезпечення більшої пропускної здатності використовується більш ємне багаторівневе кодування, яке повніше використовує динамічний діапазон. Наприклад, може використовуватись квадратурна модуляція, при якій для кодування використовується двопараметрична (амплітуда–фаза) сітка

дискретних значень, що й дозволяє за один відлік передати одночасно декілька бітів.

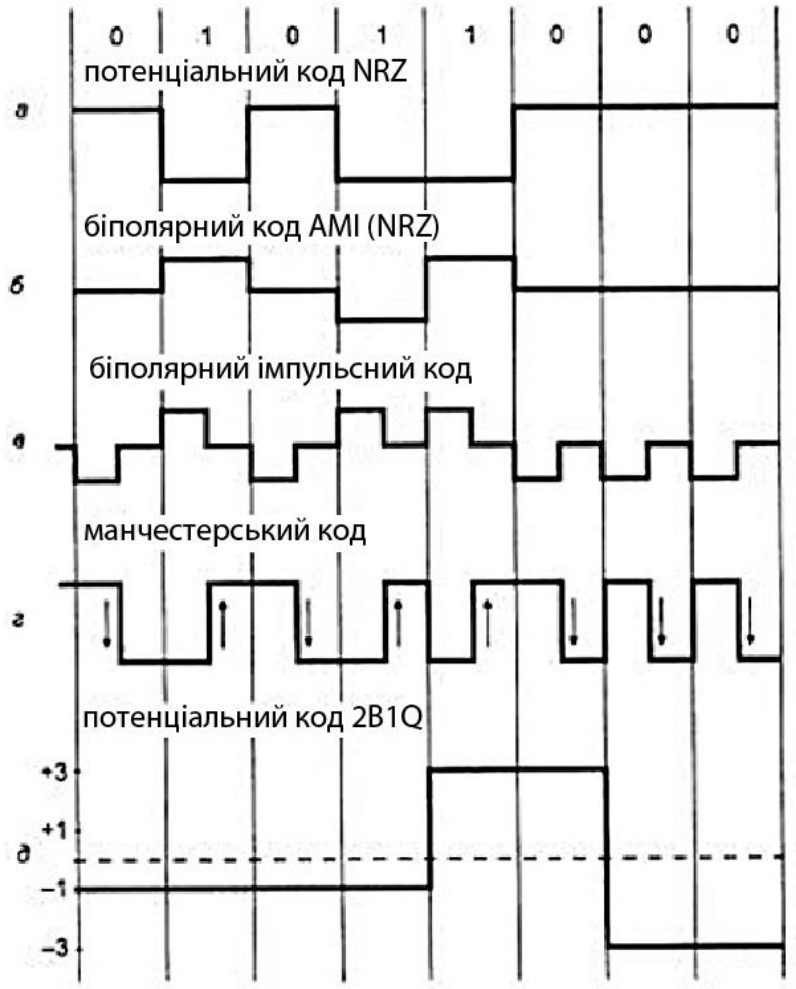


Рис. 5.13. Кодування бітового потоку

Зауважимо, що традиційно інтерфейси для зв'язування складових комп'ютерних систем обмежуються дворівневим кодуванням, тобто збільшення пропускної спроможності лінії потребує покращення частотних характеристик лінії та термінального обладнання, або розпаралелення. Таким чином у випадках, коли пропускна спроможність лінії є недостатньою, інтерфейс проектується як паралельний, з використанням декількох ліній, через які паралельно передаються дані (звідси і назва).

5.3.2. Послідовний інтерфейс.

Розглянемо особливості реалізації послідовного інтерфейсу передачі даних. В цьому випадку основою є деяка лінія передачі

призначена саме для процесу передачі даних з одного його боку в інший. Інші стандартизовані лінії інтерфейсу такого типу використовуються як допоміжні. Наприклад, популярний інтерфейс для приєднання периферійних пристроїв USB використовує лінію з двох дротів для передачі даних D- та D+ (біт кодується різницею потенціалів між ними) та два допоміжні дроти – заземлення та живлення пристрою, що приєднується.

Мінімальна кількість ліній, тобто дротів, в інтерфейсі (**далі один дріт будемо називати лінією** на відміну від лінії передачі, яка, наприклад, може бути двома дротами або, наприклад, коаксіальною) вимагає синхронізації приймача з джерелом даних за рахунок використання самого сигналу даних. З радіоелектроніки відомо, що в імпульсному сигналі як джерело синхронізації зручно використовувати його фронт, який відносно легко та надійно виділяється із загального сигналу (рис. 5.14).

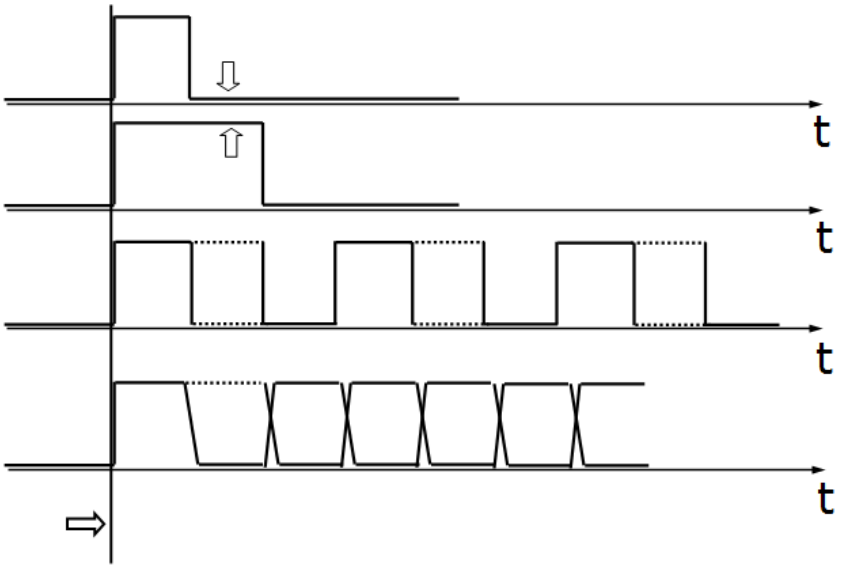


Рис. 5.14. Принцип синхронізації даних для послідовного інтерфейсу (пояснення у тексті)

Наперед обумовлена затримка від обов'язкового фронту визначає початок допустимого часового проміжку інтерпретації значення біта. При цьому гарантується незмінність миттєвого значення сигналу протягом цього проміжку часу, що забезпечує надійність отриманого коректного значення. При такому кодуванні на один біт припадає два фронти, але з врахуванням того, що значення нуля та одиниці відповідають двом різним положенням заднього фронту обмеження швидкості передачі даних визначається трьома тактами на біт. Під тактом розуміється один прямокутній імпульс з періодом, обмеженим граничною частотою складових інтерфейсу.

Швидкість передачі даних можна значною мірою збільшити за рахунок використання фронту синхронізації не на рівні одного біта, а для цілого слова, тобто деякої сукупності бітів наперед заданої довжини. Для цього використовується не одна затримка від фронту синхронізації, а «лінійка» послідовних затримок, по одній на наступний біт. Це вимагає більш якісного відтворення тактової частоти, але дозволяє суттєво збільшити швидкодію, оскільки на N інформаційних бітів слова даних використовується N+2 тактів. При цьому перший фронт обов'язковим чином є додатним (синхронізація), далі йде послідовність даних, у якій фронти наявні тільки у випадку, коли сусідні біти є різними (але все одно один такт на біт), наприкінці обов'язковий від'ємний фронт. Зазвичай формування початкового та завершального фронтів називають старт-бітом та стоп-бітом.

При реалізації послідовного способу передачі даних може бути реалізована перевірка правильності передачі слова додаванням перед стоп-бітом ще одного біта – біта парності. Цей біт доповнює до парного кількість одиниць слова і при наявності імпульсної завади, що перевертає один з бітів, загальна парність змінюється, що і визначається перевіркою на приймачі даних.

Описаний спосіб передачі даних є прийнятним для з'єднання точка–точка. При цьому досить довго протягом історії розвитку цифрової техніки послідовний інтерфейс позиціонували як відносно повільний інтерфейс для випадків, коли треба забезпечити відносно велику відстань передачі. Останній час за рахунок значного прогресу в частотах ключів інтегральних мікросхем прийшли до ситуації, коли мінімальна кількість ліній є вигіднішою за фазовий набіг за рахунок великої кількості ліній (докладніше розглянемо це у наступному параграфі), що призвело до широкого застосування послідовного інтерфейсу навіть для швидких периферійних пристроїв дисків, в тому числі SSD та HDD, принтерів, сканерів.

Розглянемо приклади деяких послідовних інтерфейсів.

На поточний момент основним зовнішнім портом сучасних комп'ютерних пристроїв (від смартфона до десктопа) є USB (англійською мовою Universal Serial Bus), розроблений групою компаній «Compaq», «Digital Equipment», IBM, Intel, «Microsoft», NEC і «Northern Telecom». Перші реалізації з'явились у 1996 р. Як вже зазначалось інтерфейс має чотири дроти, точніше дві витих пари – одна для передачі даних в диференціальним включенням), друга для подачі живлення (5 В, до 500 мА). Перша реалізація інтерфейсу (USB 1.0) мала досить малу швидкість з'єднання – 12 Мбіт/с при максимальній довжині кабелю 3 м та 1,5 Мбіт/с при довжині кабелю до 5 м. Пізніша реалізація (USB 2.0), залишивши повну сумісність з

попередньою, додала опцію високошвидкісного з'єднання – до 480 Мбіт/с. Стандарт передбачає декілька варіантів роз'ємів, в тому числі різного розміру для використання у компактних пристроях.

В листопаді 2008 року було завершено розробку нового розширення стандарту USB 3.0 зі збільшенням граничної швидкості до 4,8 Гбіт/с. Надалі, у 2013 р. – 10 Гбіт/с (USB 3.1). В третій версії інтерфейсу було збільшено кількість ліній, але з збереженням електричної та механічної сумісності з старою версією за рахунок включення додаткових контактів. Загалом такий інтерфейс використовує 10 ліній. При цьому додаткові дві виті пари обслуговують надвисокошвідкісний прийом та передачу в повному дуплексі.

Важливою особливістю інтерфейсу є можливість приєднання до 127 пристроїв (включаючи розгалужувачі) за топологією ієрархічної зірки (дерева). На кожному рівні ієрархії приєднується чотири пристрої (функціональні, або розгалужувачі). Керує загальним обміном кореневий хост.

Іншим ефективним послідовним інтерфейсом є послідовне високошвидкісне з'єднання FireWire, розроблене Sony та Apple і стандартизоване як IEEE-1394. Різні компанії використовують власні назви FireWire (Apple), i.LINK (Sony), mLAN (Yamaha), Lynx (Texas Instruments), SB1394 (Creative). Цей інтерфейс має дві перехрещені в кабелі витих пари дуплексної швидкісної передачі даних. Стандартом допускається додаткова лінія живлення. В одній топології (як і у USB деревоподібній) може бути з'єднано до 64 пристроїв. Швидкість передачі даних — 100, 200 і 400 Мбіт/с (800, 1600 Мбіт/с). На відміну від конкурента в даному інтерфейсі пристрої є рівноправними, тобто не потрібний виділений хост. FireWire також допускає декілька варіантів роз'ємів.

Наступним надзвичайно важливим послідовним інтерфейсом є SATA. (англійською мовою Serial ATA), призначений для обміну даними з дисковими накопичувачами. Цей інтерфейс є правонаступником паралельного інтерфейсу ATA (IDE), який з появою SATA було перейменовано у PATA (Parallel ATA). SATA використовує 7-контактний кабель, дві пари якого використовуються для прийому та передачі даних. Інші лінії – заземлення. Для живлення (+12 В, +5 У и +3,3 В) дискового пристрою використовується окремий кабель. Цей інтерфейс, як і USB, передбачає гарячу заміну пристроїв.

Загалом можна побачити, що простота послідовного інтерфейсу при приєднанні багатьох пристроїв вимагає перекладення реалізації протоколу на програмне забезпечення та відповідної структуризації (формату) потоку даних.

5.3.3. Паралельний інтерфейс

Паралельний інтерфейс, як вже було зазначено (п. 5.3.1), використовує декілька ліній замість однієї для забезпечення збільшення пропускної здатності. Сукупність усіх ліній, які разом використовують певну функцію, називають ***шиною***. Наприклад, на лініях шини даних пристрій-передавач виставляє дані (рис. 5.15), на кожній лінії один біт слова, що передаються паралельно.

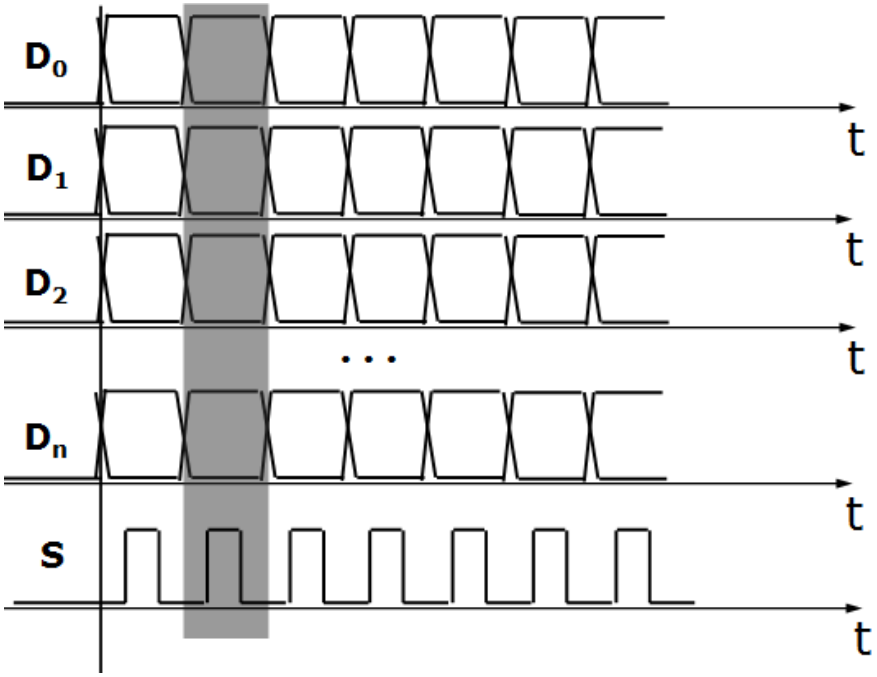


Рис. 5.15. Використання стробування для синхронізації даних паралельного інтерфейсу (пояснення у тексті)

Пристрій-приймач повинен, як і у випадку послідовного інтерфейсу мати ознаку початку часового проміжку, протягом якого можуть зчитуватись дані. Оскільки в даному випадку слово передається не як послідовність бітів, на початку якої було зручно сформувати фронт синхронізації – старт-біт (п. 5.3.2). в даному випадку для синхронізації використовуються окремі лінії. Використовуються два режими синхронізації:

стробування (рис. 5.15);

квітування (рис. 5.16).

Стробування використовується тоді, коли усі пристрої, приєднані до інтерфейсу, мають однакову швидкодію і можуть виконати елементарну операцію за часовий проміжок, який задається спеціальним тактовим генератором. Сигнал тактового генератора виступає в ролі імпульсів синхронізації – стробів.

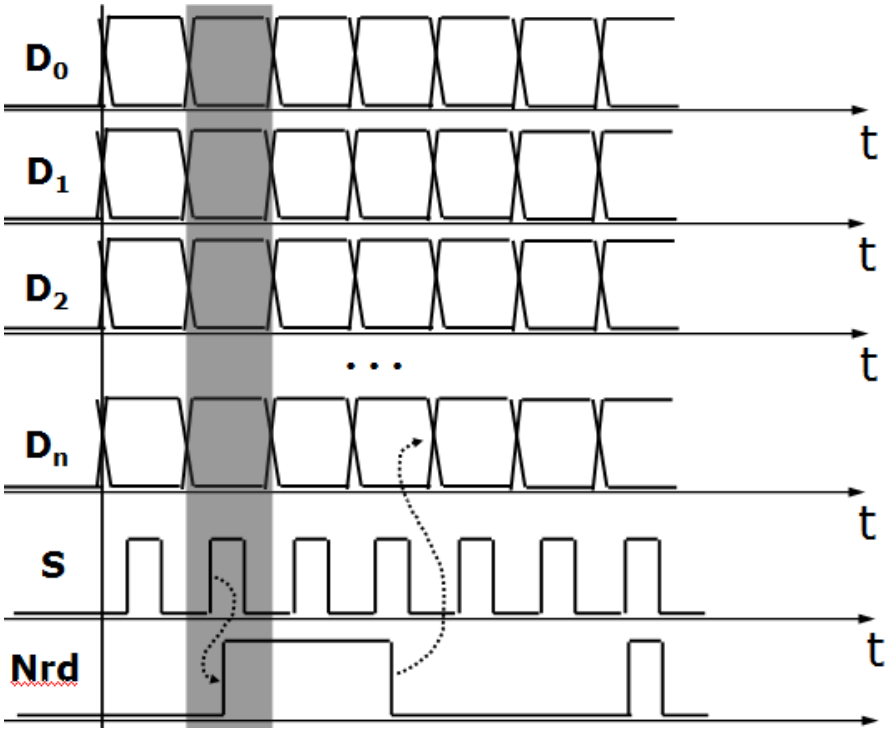


Рис. 5.16. Використання квітування для синхронізації даних паралельного інтерфейсу (пояснення у тексті)

У даному випадку для передачі одного слова є потреба у двох фронтах (перемиканнях), тобто швидкість передачі даних обмежується частотою строба, інші лінії працюють фактично на половинній частоті. З метою збільшення пропускної здатності може використовуватись синхронізація не високим рівнем стробу, тобто фактично тільки одним з його фронтів, а за обома, що вдвічі збільшує потік даних. Цей режим отримав назву ***DDR***.

Альтернативною стробування є квітування, яке дозволяє узгодити пристрої, що мають різну швидкодію. Для цього «повільний» пристрій по фронту строба виставляє сигнал неготовності, до зняття якого діє заборона на зміну даних. У більш старій реалізації квітування взагалі не використовувався строб. При цьому пристрій-передавач після встановлення даних виставляв сигнал готовності даних, приймач за цим сигналом виставляв неготовність, який знімався по завершенні зчитування і дозволяв зняти готовність даних та перейти до наступного циклу передачі слова.

Паралельний інтерфейс часто передбачає паралельне приєднання декількох пристроїв. При цьому тільки один є джерелом даних і як правило один є їх приймачем. Для того, щоб пристрої-приймачі могли визначитись, до якого з них їде звернення, використовується передача адреси, тобто умовного номера потрібного пристрою-приймача (адресата). Відповідно пристрій-приймач при

передачі через інтерфейс кожного слова виконує ***селекцію адрес***, тобто порівняння коду, виставленого на шині адрес, з власним номером.

У послідовному інтерфейсі адресація може бути також використана за рахунок виділення поля адреси у форматі повідомлення (пакету). У випадку паралельного інтерфейсу для цього часто використовують окрему шину, яку за аналогією з шиною даних називають шиною адреси. Можуть виникати потреби передачі деяких слів (кодів) й іншого змісту – команда для приймача, стан інтерфейсу тощо. Ця можливість може бути реалізована додатковими шинами. За такою архітектурою сукупність усіх шин, які призначені для передачі інформації різного призначення (безпосередньо даних, адреси, тощо) є ***інформаційною магістраллю***. Сукупність шин, призначених для потреб інтерфейсу, тобто фактично для забезпечення керуванням самим інтерфейсом, є ***магістраллю керування***.

Керувати обміном інформації у довільний момент часу для запобігання конфліктам, що супроводжуються втратою даних, повинен тільки один з пристроїв. Процес розв'язання конфліктів має назву ***арбітраж***. Арбітраж магістралі ділиться на

- централізований;
- децентралізований.

Централізований арбітраж передбачає наявність спеціального пристрою – ***арбітру***, який відповідає за надання керування обміном даних в певний момент тільки одному з пристроїв. Цей спосіб арбітражу спирається на передачу арбітру запиту від пристрою, що має захопити керування обміном даними. Арбітр передає поточному активному пристрою повідомлення на потребу передачі керування, відстежує коректне завершення поточного обміну та передачу керування іншому пристрою. При цьому може використовуватись однорівневий алгоритм (без пріоритетів), або з наданням права на керування за вищим пріоритетом. Альтернативою використання запитів є арбітраж з наданням права на захоплення керування усім пристроям по черзі (по колу).

Децентралізований (розподілений) арбітраж не передбачає наявності виділеного арбітра, тобто усі пристрої взаємодіють один з одним, розділяючи між собою відповідальність за доступ до керування магістраллю.

5.4. Засоби та технології мереженого сполучення

5.4.1. Мережа як засіб інформаційного обміну

Важко уявити сучасний інформаційний світ без комунікаційних мереж – комп'ютерних, телефонних, телебачення, тощо. Загалом ***комунікаційна мережа — система фізичних каналів зв'язку і комутаційного устаткування, що реалізовує передачу даних.*** Мережі охоплюють практично усі напрями діяльності людини – спілкування, керування обладнанням та діяльністю підприємств, доступ до сховищ даних, розваги, тощо. Велика кількість різних задач вимагає відповідних рішень, що призводить до різноманітності технологій, які повинні бути узгоджені між собою, що вимагає прийняття та імплементації певних стандартів.

Загалом мережа може спиратись як на аналогові технології обробки сигналу, так і на цифрові. Одразу зауважимо, що на сучасному рівні є стала тенденція до максимального заміщення аналогових технологій цифровими, оскільки при використанні цифрового способу кодування можна забезпечити безвтратність передачі даних на довільну відстань, ущільнення каналу використанням високоефективних математичних методів стиснення даних, комутацію у часовому домені, тощо.

Розглянемо базові принципи реалізації комунікаційної мережі на прикладі комп'ютерної мережі. Пристрої, які входять до мережі називають вузлами, з'єднання між суміжними вузлами – ланками. Надалі будемо використовувати саме ці терміни.

Загалом довільна комп'ютерна мережа – система передачі даних між комп'ютерами (та іншими пристроями), включає у себе середовище передачі сигналу в просторі (лінія або середовище передачі) та додаткові пристрої. Додаткові пристрої забезпечують:

- зв'язок комп'ютера з середовищем передачі;
- цілісність мережі;
- перенапрямлення потоків даних між ланками мережі.

Зазвичай комп'ютерні мережі класифікують за

- територіальним охопленням;
- топологією;
- середовищем передачі;
- архітектурою.

За територіальним охопленням класичним розподілом є:

- ***Локальна мережа*** (LAN, Local Area Network), мережа закритого типу, з обмеженим доступом ззовні, яка охоплює невелику територію, не має дублювання маршрутів між вузлами.
- ***Корпоративна мережа***, або мережа кампусу (CAN, Campus Area Network) – цільове об'єднання декількох локальних мереж більш широкої території:
- ***Глобальна мережа*** (WAN, Wide Area Network), охоплює великі території на основі єдиної політики адміністрування та утримання. Раніше до ознак відмінності локальних та глобальних мереж відносили використання швидких каналів у локальних мережах і більш повільних у глобальних. Але зараз такий розподіл втратив актуальність.
- ***Інтернет*** – узгоджене об’єднання необмеженої кількості глобальних мереж.

Іноді до такого розподілу додають Персональну мережу (PAN, Personal Area Network), тобто пристрої одного власника, але за сучасним підходом макі мережі є фактично компактними локальними мережами. Також можна зустріти термін «мережа міста» (MAN, Metropolitan Area Network), який відповідає охопленню міста швидкими ланками.

Архітектурно найбільш поширеними є магістраль, зірка, в тому числі ієрархічна (дерево), кільце (рис. 5.12). Окремо можна виділити характерний для глобальних мереж довільний граф ланок. Є багато інших топологій, які є або неефективними, або зводяться до комбінації перерахованих.

За середовищем передачі мережі можна поділити на

- дротові, тобто такі, що використовують для створення ланки певну лінію передачі;
- бездротові, у яких розділення каналів забезпечується способом кодуваня (наприклад, виділенням вузького частотного діапазону) та територіально (за рахунок послаблення сигналу зі збільшенням відстані від джерела).

За архітектурою мережі діляться на:

- однорангові (усі вузли мережі є рівноправними, мережа не має виділеного центру);
- клієнт-серверні (мережа має виділений центр, який забезпечує її функції).

5.4.2. Мережа як основа розподілених систем

Досить часто засоби обробки інформації утворюють єдину систему, яка охоплює велику кількість елементів, розміщених на значній території. В такому випадку кажуть про ***розподілену систему, для якої розташування її елементів (або груп елементів) відіграє значну роль у функціонуванні.*** Відповідно, обсяги системи доводиться враховувати при її розробці та використанні. Вже зараз більшість видів діяльності людини нерозривно пов'язані з такими системами, а з часом їх значимість буде ще зростати.

Причини популярності розподілених систем криються в тому, що сучасне суспільство є саме по собі глибоко інтегрованим. Така інтегрованість, як економічна, так і соціальна, вимагає швидких комунікаційних можливостей та можливостей швидкої обробки великих об'ємів інформації. При цьому часовий розподіл подібних вимог є вкрай нерівномірним. Тільки включення окремих локальних ресурсів у спільне об'єднання дозволяє вирішити ці проблеми.

Розподілена система може бути «одноранговою», коли усі вузли мають приблизно однакову функціональність, а може мати розподіл функцій за вузлами або їх групами. Для адміністрування такої системи може використовуватись (не обов'язково) єдиний керуючий центр або єдина політика. Значна частина дуже великих систем не має жорсткої централізації, що забезпечує її стійкість навіть при втраті працездатності помітної кількості елементів.

Кожна розподілена система створюється для виконання певного функціонального навантаження. Як приклади найбільш часто використовуваних розподілених систем різного призначення можна виділити:

- комунікаційні мережі;
- системи управління підприємств або керування технологічними процесами;
- мережеві системи накопичення інформації;
- системи розподілених обчислень;
- система доменних імен (DNS).

Ефективність розподіленої системи значною мірою залежить від ефективності та надійності доставки даних для забезпечення взаємодії окремих елементів цієї системи. За потреби забезпечення особливих умов безпеки, гарантованої швидкості доставки або через потребу великого трафіку можуть використовуватись і власні закриті канали даної системи, але частіше, особливо для систем, що охоплюють велику територію, використовуються засоби загально використовуваних інформаційних мереж.

Яскравим прикладом надвеликого розподіленого утворення є Інтернет (як зазначено в п.5.4.1, узгоджене об'єднання необмеженої кількості мереж). Якщо взяти довільну обмежену (закриту) мережу, то її комунікаційна система, що включає певну кількість вузлів та ланок між ними, надає можливість абонентам мережі (але тільки цієї мережі) налагодити потрібний інформаційний обмін. Завдяки тому, що в рамках Інтернету до сукупності таких мереж додані міжмережеві з'єднання, та використовуються сумісні методи (протоколи та формати) передачі даних, а також діє формальна домовленість про взаємне використання ланок різних мереж, можливе забезпечення шляху проходження між довільними вузлами всієї території, яку охоплюють мережі Інтернету.

5.4.3. Основні технології комп'ютерних мереж

У зв'язку з особливою важливістю для розподілених систем забезпечення ефективної і надійної доставки даних детальніше розглянемо основні принципи мережевого сполучення.

Під терміном ***«Мережа передачі даних» зазвичай розуміють довільну сукупність термінальних пристроїв, об'єднаних каналами передачі даних***. Цей термін може застосовуватись для мереж довільного призначення, які працюють як в аналоговому, так і в цифровому режимі. Далі будемо використовувати більш вузький клас мереж, яким є ***комп'ютерна мережа – система зв'язку між двома чи більше комп'ютерами***.

Довільна комп'ютерна мережа повинна забезпечити передачу даних між її вузлами, для чого повинен бути вирішений ряд загальних задач, серед яких найважливішими є :

- адресність доставки;
- забезпечення надійності доставки;
- розділення середовища між багатьма інформаційними потоками.

Передача довільного потоку інформації передбачає використання деяких способів інтерпретації бінарного потоку (форматів даних) та сукупності правил, за якими виконується необхідна послідовність дій (протоколів). Тобто зазначені задачі вирішуються саме створенням відповідних протоколів, які використовують певні формати. При цьому потік інформації розкладається на послідовність блоків даних, які додаванням деякої службової інформації перетворюються в інформаційні **пакети**.

Адресація у мережах. Для довільного пакету, який передається через мережу, існує відправник (джерело даних) та адресат (отримувач даних). Для коректної доставки пакету є обов'язковою адреса адресата та бажаною адреса відправника. Оскільки комп'ютер зазвичай використовується у багатозадачному режимі і може існувати багато процесів, які вимагають мережевого обміну даними з іншими комп'ютерами, є потреба забезпечити унікальність ідентифікації:

- вузлів мережі (будь-якого рівня);
- процесів на вузлі.

Для ідентифікації комп'ютерів (хостів), які є вузлами мережі, в залежності від рівня реалізації (про це трохи згодом) використовуються два типи адрес:

- MAC-адреса мережевого адаптера;
- умовний код хоста (для найбільш розповсюджених зараз IP-мереж це IP-адреса).

MAC-адреса (Media Access Control) – унікальна адреса, що "зашивається" виробником у мережевий адаптер, через який виконується доступ даного комп'ютера до локальної мережі. Довжина MAC-адреси складає шість байтів, з яких три ідентифікують виробника пристрою, а інші три – унікальний номер, що назначається виробником для даного конкретного пристрою. Формально, в світі не повинно існувати двох мережевих адаптерів з однаковими MAC-адресами, проте це не завжди виконується. Тому загальним обмеженням на рівні локальної мережі можна визначити вимогу унікальності MAC в межах даної мережі. Зауважимо, що ця адреса може замінюватись адміністратором використанням спеціальної утіліти.

IP-адреса (IP – Internet Protocol) дозоляє реалізувати деяку глобальну адресацію. Перше покоління реалізації такої адреси, **IPv4** (ця версія і зараз активно використовується), передбачало чотирибайтове поле адреси, зазвичай для зручності користувачів при відтворенні у текстовій формі записується чотирма числами від 0 до 255 через крапки, наприклад, 216.65.41.182.

Через потребу розбиття загального (спільного) простору на окремі мережі передбачено розділення адреси на дві частини:

- адреса мережі;
- адреса абонента,

що дозволяє виконувати перевірку належності пакету до даної мережі. Для цього використовується бітове OR значеня адреси з ***маскою мережі*** – чотирибайтовим числом, у якому одиниці визначають саме ту частину адреси, яка визначає саме мережу (інші біти є нульовими). Границя поля адреси мережі може варіюватись:

- мережа класу A – використовується в великих мережах зі значною кількістю вузлів (адреса мережі займає 1 байт – 0xxxxxxx);
- мережа класу B – середні мережі (адреса мережі займає 2 байти – 10xxxxxx xxxxxxxx);
- мережа класу C – малі мережі (адреса мережі займає 3 байти – 110xxxxx xxxxxxxx xxxxxxxx);
- мережа класу D – групи станцій (адреса мережі обмежується 4 –бітами 1110);
- мережа класу E – зарезервовано (адреса мережі виділено 5 бітів –11110).

Чотирибайтова глобальна адреса стала своєрідною «міною сповільненої дії». Загалом, така довжина поля адрес може забезпечити трохи більше 4 млрд. абонентів. Але за рахунок наведеної вище структуризації адреси до такого обмеження додається ще і нерівномірний розподіл пулу вільних адрес. Як результат, вже на початку 90-х почали проявлятись проблеми з нестачею адрес. 14 вересня 2012 року організація RIPE NCC повідомила про розподіл останнього вільного блоку.

На зміну IPv4 приходить **IPv6** (англ. Internet Protocol version 6). В даному стандарті довжина адресного поля була збільшена до 128 біт, тобто в чотири рази. Цей протокол вже використовується багатьма тисячами мереж, хоч поки що не отримав загального розповсюдження.

При використанні текстового запису (для зручності людини) такої подовженої адреси використовується трохи інший синтаксис – адреса розбивається символом «двокрапка» на двобайтові блоки, які записуються шістнадцятковими цифрами, наприклад, fe80:0:0:0:200:f8ff:fe21:67cf. Одна чи декілька нульових груп підряд (тільки один пропуск в адресі) може бути пропущена з заміною на подвійну двокрапку (fe80 :: 200: f8ff: fe21: 67cf).

Саме IP адреса зазвичай є адресою прив’язки інформаційних ресурсів і супроводжує обмін даними. Але для користувачів навіть у текстовій формі вона є надто незручною. Саме з орієнтацією на людей паралельно до IP-адреси була введена система доменних імен – зручної текстової форми адреси, яка має змістовне навантаження. Доменна адреса є ієрархічним об’єднанням (через крапку як роздільник) декількох доменних імен (корньовий домен позаду), кожне з яких відповідає за свій домен (множину) імен, наприклад, user_host.rpd.univ.kiev.ua. Домен є довільним об’єднанням абонентів або доменів нижчого рівня:

- може відповідати деякому фізичному утворенню:

    - локальній або глобальній мережі;
    - державі (наприклад, .ru, .ua, .de);
- охоплює логічні (змістові) зв'язки (наприклад, .com, .gov, .net).

Оскільки доменні імена призначені для використання людьми і безпосередньо при адресації даних, що пересилаються, не використовуються, потрібно забезпечити трансляцію доменної адреси з запиту користувача у цифрову IP-адресу. За це відповідає DNS – Domen Name System (Service), тобто мережа серверів (сервіс).

DNS є прикладом ієрархічної розподіленої бази даних, призначеної для обробки надзвичайно великої кількості запитів. При потребі звернення до певного хоста з комп'ютера користувача відправляється запит на сервер, який відповідає за зону користувача (посилання на цей сервер вказується у настройках комп'ютера). Якщо цей сервер не може виконати запит, тобто у його таблиці немає відповіді, запит переадресується до іншого сервера за ієрархічними посиланнями. Результатом виконаного запиту є IP-адреса, яка надалі і використовується програмним засобом в процесі інформаційного обміну з зазначеним хостом. Слід зауважити, що одній доменній адресі можуть відповідати декілька IP, так само як одній IP можуть відповідати декілька доменних.

Оскільки для адресації до певного інформаційного ресурсу на хості зазвичай адреса хоста є недостатньо повною, до зазначеної адреси додається інша інформація, потрібна для виконання запиту. Такий доповнений рядок отримав назву URL – Uniform Resource Locator. Наприклад:

http://www.rpd.univ.kiev.ua/eng/index.php

Таким чином, структура URL включає у себе:

- метод доступу http, https (захищений протокол), ftp, file;
- ідентифікаційне ім'я користувача і пароль у форматі USER:PASSWORD@. (необов'язкові елементи);
- мережеве ім'я сервера, де знаходиться інформація;
- номер порту після адреси сервера (за потреби);
- шлях у файловій системі сервера та ім'я файлу через прямі слеші (/), якщо ім'я файлу не вказано, відображається файл, на який налаштовано сервер для відображення за домовленістю;
- параметри, які передаються на сервер для виконання даного запиту, починаються знаком питання, роздільником є знак &

(?параметр_1=значення_1&параметр_2=значення_2&параметр3=значення_3);

- при необхідності відображення документа не з початку далі через символ # ставиться якір (anchor), який вказує саме той рядок документа, що повинен відображатись у верхньому рядку браузера.

Як вже було зазначено вище, на даному комп'ютері можуть одночасно існувати декілька процесів, які потребують мережевої роботи. Відповідно, до мережевої адреси слід додати ознаку процесу-адресата (або його частини). Така ознака отримала назву порту. Таким чином, пакети, що надходять на даний комп'ютер з мережі, фільтруються за ознакою мережевої адреси, і надалі організуються у вигляді багатьох черг, кожна з яких відповідає своєму порту. Для завдання порту використовується двобайтове беззнакове ціле число, тобто є доступним діапазон значень 1-65535:

- порт 0 не використовується. Якщо 0 передати з прикладного процесу як номер порту, буде виконано автоматичний вибір в діапазоні 1024-4999;
- порти 1-255 зарезервовані для стандартних мережевих служб, таких як Telnet, FTP тощо;
- порти 256-1023 зарезервовані для інших служб загального призначення, наприклад, маршрутизації;
- порти 1024-4999 вважаються клієнтськими;
- порти 5000-65535 рекомендовані для використання серверами.

Сукупність адреси хоста і порту визначають «сокет» (Socket), тобто «гніздо».

Забезпечення надійності доставки. В залежності від потреб використання обмін інформації може вимагати:

- точної доставки усього потоку даних зі співпадінням до останнього біту;
- максимально швидкої доставки, жертвуючи якістю доставки.

Часто ці два режими називають:

- потоковим;
- дейтаграмним.

Ці два варіанти інформаційного обміну реалізуються різними протоколами. Надійність доставки пакетів забезпечується відправкою у зворотний бік спеціального пакету-квитанції, яка або підтверджує коректну доставку, або є вимогою на повторну відправку

(ретрансляцію). Зрозуміло, що такий алгоритм доставки саме і призводить до деякого сповільнення.

Терміном сокет також позначають і програмну реалізацію обслуговування мережевого з'єднання. При утворенні потокового каналу його кінці розділяють на серверний і клієнтський. Спочатку для формування каналу на сервері утворюється деякий сонет в режимі прослуховування (умовно назвемо його InSocket) із портом, номер (Port_Serv) якого за домовленістю відомий клієнту. Клієнт з деякого сокету (ClientSocket) звертається на даний порт по з'єднання, сервер створює інший сокет (ServerSocket), який і є логічним терміналом новоствореного каналу. Протягом сеансу (до його завершення) пара сокетів ClientSocket–ServerSocket можуть передавати дані як в один, так і в інший бік, до цього обміну InSocket вже відношення не має.

Для реалізації дейтаграмного режиму на кожній станції, що входять до комунікаційної системи, утворюється сокет, тут вже не йде мова про умовний серверний бік. Кожний з абонентів через свій сокет може звернутись до будь-якого іншого вказуванням мережевої адреси і порту адресата. Зрозуміло, що ці реквізити повинні бути відомі відправнику повідомлення.

Розділення середовища. При забезпеченні інформаційного обміну часто виникає задача суміщення в той самий проміжок часу потоків даних від різних інформаційних процесів. Якщо віддати усі потрібні для з'єднання двох вузлів ланки (їх сукупність створить наскрізне з'єднання) на час усього сеансу, інші потоки, які потребують цих ланок, будуть блокуватись. Саме для запобігання цьому потік даних і розбивається на послідовність блоків. Для організації мережі, яка використовує таке розбиття, треба реалізувати деякий досить складний набір протоколів, які при роботі з кожним блоком даних потребують додаткової інформації. Для цього блок даних доповнюють заголовком зі службовою інформацією, що і утворює **пакет**.

Мережа, яка використовує таку ідеологію, називається мережею з комутацією пакетів. З метою узгодження програмного і апаратного забезпечення, яке розв'язує різні задачі при наскрізній передачі абонент–абонент, використовується розділення задач за деякими рівнями таким чином, що довільний рівень є користувачем сукупності послуг, які постачає нижчий сусідній рівень, і постачальником послуг для верхнього сусіда. На основі такого розподілу у 1983 році було вироблено стандарт на базову еталонну модель взаємозв'язку відкритих систем (Open System Interconnectivity – OSI), яка

складається з 7 послідовних рівнів, для кожного з яких виконується окрема стандартизація логічних інтерфейсів і протоколів (табл. 5.1).

Таблиця 5.1. Основні задачі рівнів OSI

№	Назва рівня	Задача рівня
7	Прикладний	Змістове навантаження (семантика) всіх процесів. Саме можливості цього рівня безпосередньо цікавлять користувача
6	Представлення даних	Перекодування синтаксису і формату даних, забезпечує незалежність для прикладних процесів від розбіжностей у формі представлення даних. Може виконуватись шифрування даних (SSL)
5	Сеансовий	Забезпечує керування взаємодією, визначає початок і кінець завдання, поновлення зв'язку після помилок під час сеансу
4	Транспортний	Забезпечує надійність проходження повідомлення між кінцевими абонентами (адресування, встановлення відповідності між адресами і мережевими іменами абонентів, розборка і зборка повідомлень сеансового рівня
3	Мережевий	Логічне об'єднання ланок у мережу (маршрутизація, підтримка віртуальних з'єднань, формування-розформування пакетів, їх адресування, керування потоками пакетів, пріоритетністю їх передачі). Забезпечує незалежність вищих рівнів від різноманітності фізичних засобів зв'язку
2	Канальний	Забезпечує передачу інформації між довільними вузлами мережі (утримує функціональні і процедурні засоби передачі даних між компонентами мережевого рівня – функції встановлення, підтримки, роз'єднання ланки даних і керування ланкою даних)
1	Фізичний	Виконує передачу бітів по фізичних каналах (забезпечує механічні, електричні, функціональні і процедурні засоби встановлення, підтримки і роз'єднання з'єднань)

На кожному з рівнів працюють свої протоколи. Сукупність протоколів, яка повністю забезпечує інформаційний обмін, називається стеком. Реальні стеки протоколів можуть відрізнятись за кількістю рівнів від OSI, але в основному відтворюють його ієрархію. На рис. 5.17. наведена відповідність моделі OSI найбільш популярному зараз стеку TCP/IP.

Рис. 5.17. Співвідношення рівнів OSI та TCP/IP

На рис. 5.17 на відповідних рівнях позначені також найбільш застосовувані протоколи стеку: HTTP (hipertexp transfere protocol) – головний протокол обслуговування web технологій, Telnet – віддалений доступ до командного рядка, FTP (file transfer protocol) – протокол обміну файлами, SNMP (simple network management protocol) – простий протокол керування мережею, SMTP (simple mail transfer protocol) – підтримка поштових повідомлень, ICMT (internet control message protocol) – сумісно з IP утворює IP-модуль, ARP (address resolution protocol) – забезпечує переведення IP адрес в Ethernet адреси, RARP (reverse address resolution protocol) – зворотній до ARP, RIP (Routing Internet Protocol) і OSPF (Open Shortest Path First) – відповідають за маршрутизацію, IEEE 802.3 – IEEE 802.12 – протоколи локальних мереж SLIP (serial line internet protocol), PPP (point to point protocol) – протоколи передачі кадрів по послідовних каналах.

Складна структура ланок глобальної мережі вимагає для проходження кожного повідомлення утворення відповідного маршруту проходження пакетів, виходячи з фізичного розташування відправника і адресата. Маршрутизація, як одна з головних функцій мережевого рівня, є дуже складною задачею і не має однозначного розв’язку, оскільки потребує координації усіх вузлів мережі. Вона повинна враховувати зміни навантаження і тимчасові виходи з ладу окремих ланок і вузлів. Задачею маршрутизації є визначення для даного маршрутизатора напрямку на один з найближчих маршрутизаторів, у який слід відправити пакет, що надійшов.

Алгоритм маршрутизації повинен мати цілком визначені властивості, до яких відносяться:

- надійність;
- коректність;
- стабільність;
- простота;
- оптимальність.

Найбільш ефективним способом маршрутизації є динамічна, для якої на маршрутизаторах утримуються спеціальні таблиці, за якими і визначається напрямок відправки отриманого пакету (за IP-адресою, яка є у пакеті). Динамічна маршрутизація передбачає періодичний аналіз зміни стану мережі і відповідне коректування цих таблиць. Для дослідження стану мережі використовуються спеціальні протоколи.

5.4.4. Особливості реалізації розподілених систем

При виконанні деякого завдання узгодженими діями двох чи більше пристроїв обробки інформації (комп'ютерами) виникає потреба:

- обмінюватись частиною даних між цими пристроями;
- узгоджувати виконання (синхронізуватись).

При розподілі частини загальної задачі по складових системи слід брати до уваги можливі затримки доставки, особливо коли використовуються загальні канали. В цьому випадку можлива ситуація, коли завдання одного вузла повинне очікувати для продовження виконання свого процесу результатів іншого вузла. Через це при проектуванні розподілених систем велику увагу приділяють оптимізації розподілу задач по окремим вузлам системи. При цьому слід брати до уваги і те, що далеко не завжди збільшення ресурсів, які виділяються для розв'язання певної задачі, призводять до бажаного приросту швидкості виконання (це питання розглянемо у наступному параграфі).

За рахунок того, що розподілена система охоплює велику територію, до її складу зазвичай включається велика кількість комп'ютерів, в тому числі часто й таких, для яких значна кількість задач не пов'язана з цією системою. В результаті склад системи може бути вкрай неоднорідним. Крім того, зростання розмірів системи часто супроводжується більш ніж лінійним зростанням ресурсоємності обробки даних. Відповідно, основні проблеми, які доводиться вирішувати у подібному випадку, можна згрупувати по двом категоріям:

- неоднорідність;
- масштабність.

Розглянемо докладніше ці особливості та найпопулярніші шляхи їх вирішення.

Неоднорідність. При створенні відносно невеликої системи не є проблемою забезпечити однорідність за платформою. Але на довільному робочому місці зазвичай складається потреба доступу до великої кількості досить різних задач. Відповідно, для корпоративного рівня розподілена система повинна передбачати можливість роботи на різних платформах. Найпростіше це реалізується Інтернет-інтранет технологією (перенесенням методів роботи в Інтернеті на внутрішню мережу, п. 6.1.1), але у цьому випадку розподіленість фактично відноситься до забезпечення доступу до серверу з довільного вузла системи, а основне обчислювальне навантаження припадає на сервер (ядро системи).

Реалізація повноцінної багатоплатформовості для повноцінного використання клієнт-серверної архітектури (яка забезпечує розподіл обчислювального навантаження між момп'ютером користувача – клієнтом та ядром системи – сервером) є досить складною задачею, особливо якщо взяти до уваги необхідність реалізації однакової логіки роботи на різних платформах та забезпечення подібного інтерфейсу користувача. Засобом спрощення подібних розробок є інтегровані програмні середовища розробки (framework), які полегшують перенесення програм між різними середовищами. До них належать Windows Open Systems Architecture (WOSA); Win32, загальне відкрите програмне середовище UNIX COSE і App Ware Foundation та інші. Для вказаної задачі, а саме реалізації клієнтського програмного забезпечення розподіленої системи, також можуть використовуватись середовища виконання, які орієнтуються на використання проміжної компіляції, наприклад, Java. Альтернативним варіантом є використання MS.NET завдяки Mono – багатоплатформовій відкритій (Open source) реалізації середовища MS.NET. Ця система відповідає стандартам ECMA, включаючи серед іншого підтримку C# і Common Language Runtime, реалізацію ADO.NET і ASP.NET, графічної системи GTK+.

Масштабованість. ***Можливість збільшення розмірів системи зі збереженням її основних характеристик називається масштабованістю***. Ця властивість є особливо бажаною у галузі телекомунікацій та інформаційних технологій. Масштабованість, зазвичай, важко визначити однозначно і в кожному випадку необхідно

вказувати конкретні параметри, за якими в ній є потреба. Алгоритм, архітектура, мережевий протокол, програма або інша система називаються масштабованими, якщо вони ефективні в застосуванні до великих задач (наприклад, великий набір вхідних даних або велика кількість вузлів у випадку розподіленої системи).

При збільшенні обсягів системи може виникати ситуація, коли кількість звернень до деякого сховища інформації або спроба передати певний об'єм даних через даний комунікаційний сегмент перевищує його можливості. Зазвичай балансування швидкості запитів і швидкості обслуговування забезпечується використанням черги, що дозволяє вирішити проблему великого пікового (але не середнього) навантаження, проте в даному випадку черга буде постійно збільшуватись аж до повного заповнення виділених для резервування ресурсів. Відповідно, збільшення системи повинно передбачати можливість появи такої проблеми. Для цього можуть використовуватись такі шляхи:

- при забезпеченні доступу до даних:
    - кешування даних;
    - дзеркалювання систем доступу до інформації;
    - розподілення джерела інформації;
- в комунікаційній системі:
    - дублювання шляхів доставки;
    - оптимізація шляхів доставки;
    - оптимізація методів доставки.

Кеш (англ. cache) – проміжний буфер зі швидким доступом, що містить інформацію, яка може бути запрошена з найбільшою ймовірністю. Кешування може використовуватись як проміжний елемент навіть в рамках локальної системи для узгодження швидкодії складових з суттєво різною швидкодією, наприклад, кешування оперативної пам'яті або кешування жорсткого диску. В даному випадку може виконуватись буферизація даних, які вже надійшли до деякої ділянки загальної системи. За рахунок цього наступний запит тих саме даних буде задовольнятись не первинним джерелом даних, а системою кешування, що зменшує навантаження на джерело. Кешування (разом з іншими задачами) виконує **проксі-сервер**.

При великій кількості запитів на доступ до інформації завантажуються як сам сервер, так і ланки мережі, які забезпечують доступ до нього. Для зменшення навантаження можна зробити переадресацію на інший ресурс – ***дзеркало (англ. Mirror) – точну або часткову копію (зазвичай вважається не менше 80 %) даних одного сервера на іншому***.

Замість створення декількох копій даних для зменшення кількості запитів на кожне зі сховищ можна розмістити на окремих ресурсах окремі частини загального об'єму даних. З одного боку це дозволяє економити ресурси завдяки відсутності дублювання, але з іншого боку призводить до ускладнення системи та збільшення кількості переадресацій, що треба враховувати при виборі загальної архітектури під час проектування системи.

Проблема доставки даних є найбільш критичною при використанні розділення середовища. Це стосується застарілої архітектури дротових локальних мереж на основі концентраторів. Зараз така проблема втратила актуальність, оскільки комутатор (замість концентратора) дозволяє сегментувати мережу та передавати декілька пакетів одночасно. На рівні глобальних мереж (WAN) розв'язання такої проблеми реалізується для усієї сукупності задач створенням хребтової (backbone) мережі, додаванням додаткових ліній передачі в разі перевантаження існуючих. Але слід звернути увагу на те, що значною мірою на вирішення такої задачі впливає маршрутизація, оскільки саме вона визначає шляхи доставки і перерозподіл трафіку по мережі. Відповідно, саме оптимізації маршрутизації зазвичай приділяють велику увагу.

Оптимізація методів доставки може включати у себе використання методів стиснення даних та зменшення кількості даних, які передаються через комунікаційну систему при потребі дублювання для багатьох адресатів. Для зменшення кількості пакетів може використовуватись широкомовний (broadcast) та груповий (multicast) режими передачі. Широкомовний режим передбачає передачу пакету (але одного, а не багатьох дубльованих) на всю (певну) мережу. Груповий режим реалізується розмноженням пакетів на потрібні (за ознакою групи) напрямки.

5.4.5. Перспективи розвитку розподілених систем

Розподілені системи з часом будуть охоплювати нові і нові галузі застосування. Це пов'язано з постійним зростанням об'ємів інформації, які доводиться обробляти, та з певними проблемами забезпечення зростання швидкодії відокремленого пристрою. Найбільш яскраво цю ситуацію можна спостерігати у такій галузі, як виконання обчислень, до яких, наприклад, призводять задачі моделювання у багатьох галузях – від наукових досліджень до розрахунків синтезованого відео для кінематографії та сфери інтерактивних розваг.

Досить довго основними шляхами підвищення швидкодії обчислювального пристрою було підвищення тактової частоти та розрядності (п. 5.1.1). Але збільшення частоти зіткнулось із принциповими фізичними труднощами, а збільшення розрядності забезпечувало суттєвий приріст доти, доки не дійшли до охоплення в регістрі максимального потрібного діапазону значень. Відповідно, зараз основним шляхом боротьби за продуктивність обчислювальної системи є паралелізм.

Першими варіантами використання паралелізму стали конвеєризація, суперскалярні та векторні процесори. Конвеєризація дозволила розкладом операції на декілька складових застосувати одночасну роботу декількох пристроїв, кожен з яких одночасно з іншими виконує власну частину операції, передаючи по естафеті конвеєра результати виконання наступному блоку. Суперскалярність як розвиток потреби окремого блоку виконання для цілих та дійсних даних дозволила розкласти потік на декілька конвеєрів. Для задач, у яких часто застосовуються операції з векторами даних, реалізується оптимізована під такі задачі архітектура. Але і ці шляхи прийшли до певних обмежень масштабування, пов'язаних із накопиченням даних та передачею їх на відповідні блоки обчислень. Більш ефективною у багатьох випадках стала більш стала архітектура з масштабуванням на більшому рівні:

- ядро як майже повноцінний процесор (багатоядерний процесор);
- процесор (багатопроцесорна система);
- процесор з деяким оточенням (суперкомп'ютер);
- повноцінна обчислювальна система (суперкомп'ютер, кластер, грід).

Багатоядерні та багатопроцесорні системи використовують загальний пул пам'яті, хоча вже у цьому варіанті через потребу кешування з'являється проблема синхронізації даних, які використовують різні обчислювальні модулі. Масштабування на рівні повноцінних систем робить процес обміну даними між окремими вузлами ще критичнішим. Але такий шлях цікавий завдяки тому, що обмеження рівня масштабування визначається тільки особливостями самого обчислювального процесу.

Розглянемо детальніше цю проблему. Загальна ідея послідовного алгоритму спирається на те, що результат операції далі використовується як операнд, що визначає загальний порядок виконання операцій. У багатьох алгоритмах за рахунок повторення певних наборів дій із різними даними можна ці дії виконувати одночасно на різних пристроях, що і є паралельним виконанням. Але

так чи інакше виникають ситуації, коли один із пристроїв вимагає для продовження роботи результату від іншого. Зрозуміло, що чим більше буде таких ситуацій, тим до меншого виграшу буде призводити збільшення кількості обчислювальних пристроїв. Ще однією проблемою є загальна кількість пам'яті, якої вимагає дана задача. У цьому плані ієрархічна архітектура об'єднання повноцінних комп'ютерів збільшує ефективність використання надвеликих об'ємів пам'яті.

Ефективність розпаралелювання виконання та звернень до сховища даних значною мірою залежить від конкретної задачі, яка розв'язується. Є такі задачі, які взагалі немає сенсу розпаралелювати, є такі, для яких більш критичним є обмін інформації між вузлами (тобто критичним елементом є швидкодія комунікаційної системи), є задачі, які можна разпаралелити на дуже великій кількості вузлів, навіть нехтуючи швидкістю взаємодії вузлів.

Відповідно, зараз активно розвиваються усі зазначені напрямки, в тому числі використовуються і певні комбінації, наприклад, при створенні кластеру використовуються багатопроцесорні вузли. Особливу увагу приділяють розвитку кластерів та грід.

Кластер — це декілька незалежних обчислювальних машин, що використовуються спільно і працюють як одна система для вирішення тих чи інших задач. Зазвичай кластери проектуються як локалізована система зі швидкою комунікаційною складовою, бажано з однаковими характеристиками вузлів, що спрощує балансування їх навантаження.

Системою, менш прив'язаною до умови компактності, є грід. Загалом, ***грід (англ. grid – решітка, мережа) – це форма розподілених обчислень, в якій система будується на основі слабко зв'язаних вузлів.*** З точки зору мережевої організації грід являє собою узгоджене, відкрите і стандартизоване середовище, яке забезпечує гнучке, безпечне, скоординоване використання за рахунок розподілу ресурсів як обробки, так і зберігання інформації. Відповідно, грід може будуватись як на основі окремих комп'ютерів, так і на базі обчислювальних кластерів.

Сучасна концепція розпаралелювання обробки та накопичення даних охоплює досить швидко прогресуючий напрямок – хмарну обробку. Ця концепція базується на тому, що користувачу не принципово, на яких ресурсах будуть виконані потрібні йому обчислення (цікавить тільки час отримання результату), і де саме і у якому вигляді зберігається його інформація при умові надійності та захищеності сховища.

Хмарні обчислення (англ. Cloud Computing) — це модель забезпечення зручного доступу з довільного місця через мережу до спільного пулу обчислювальних ресурсів, що підлягають налаштуванню (наприклад, до комунікаційних мереж, серверів, засобів збереження даних, прикладних програм та сервісів), і які можуть бути оперативно надані та звільнені з мінімальними управлінськими затратами та зверненнями до провайдера. [35].

При використанні хмарних обчислень програмне забезпечення надається користувачеві як Інтернет-сервіс. Користувач має доступ до власних даних, але не може управляти і не повинен піклуватися про інфраструктуру, операційну систему і програмне забезпечення, з яким він працює.

Хмарне сховище даних (англ. cloud storage) – модель онлайн-сховища, в якому дані зберігаються на значній кількості серверів, розподілених в мережі.

Ресурси хмари надаються в користування клієнтам деяким представником системи, який бере на себе задачу узгодження. Дана модель збереження даних відрізняється від традиційного використання власного виділеного сервера або оренди частини дискового простору чужого сервера тим, що користувачу невідомі, внутрішня структура серверів, їх кількість та географічне розташування. Дані зберігаються на одному великому віртуальному сервері. При цьому користувач платить тільки за те місце в сховищі, яке фактично використовує, а не за оренду пулу ресурсів (сервера чи його певної частини), які він може повністю і не використовувати. Загалом, така ідея робить зберігання великих об’ємів даних більш економним. Крім того, усі процедури з резервування та збереження цілісності даних проводяться провайдером даного сервісу.

Слід зазначити, що хмарне збереження даних має деякі не до кінця вирішені проблеми:

- конфіденційність (безпека зберігання та пересилання даних);
- можливе зменшення загальної продуктивності при роботі з даними в хмарі, ніж з локальним сховищем.

5.5. Питання для самоконтролю

1. Чим відрізняється архітектура сервера в порівнянні з архітектурою користувацьких комп’ютерів?
2. Які функції виконують регістри процесора?
3. Чим особливий такий регістр, як акумулятор?

4. Чи є можливість необмежено збільшувати тактову частоту процесора?
5. Чому подальше збільшення розрядності процесора не є таким ефективним, як на початку розвитку мікропроцесорної техніки?
6. Які архітектурні особливості сучасних мікропроцесорів?
7. Чому не отримали розповсюдження багатоядерні мікроконтролери?
8. За рахунок чого багаторівнева КЕШ збільшує ефективність роботи процесору?
9. Чому ОЗП реалізується зазвичай на динамічних комірках?
10. За рахунок чого SSD є більш ефективним в порівнянні з HDD?
11. Чи забезпечує резервне копіювання достатньо високу надійність збереження даних в інформаційній системі банку?
12. Спробуйте оцінити зменшення ймовірності втрати даних у RAID 1 в порівнянні з поодиноким накопичувачем.
13. Чим визначається обмеження кількості приєднань до шини?
14. Чим відрізняється передача адреси в послідовному та паралельному інтерфейсах?
15. Наведіть приклади задач, що потребують застосування розподілених систем.
16. Чому в IP-пакетах не використовується доменна адреса?
17. Для чого використовується ретрансляція пакетів?
18. Чи для будь-яких обчислювальних задач паралельність обчислень дає певну перевагу у швидкодії?

6. Обробка складної інформації

6.1. Інформаційні системи та бази даних

6.1.1. Інформаційні системи

Обробка інформації досить часто супроводжується накопиченням дуже великих об'ємів даних та маніпулюванням ними. Масштабування задач, пов'язаних з обробкою інформації, не є лінійним. Те, що без проблем реалізується на відносно малих об'ємах даних, при певних масштабах взагалі може втратити працездатність. Відповідно, для рівня великих об'ємів інформації, як правило, використовується деякий системний підхід. Хоча технології, придатні для реалізації надвеликих систем, часто використовуються і у відносно простих випадках, для реалізації яких ці технології певною мірою є надлишковими, але ця надлишковість компенсується спрощенням розробки завдяки уніфікації рішень.

Системність роботи з інформацією призводить до терміну ***інформаційна система – сукупність організаційних і технічних засобів для збереження та обробки інформації з метою забезпечення інформаційних потреб користувачів***. Загалом, інформаційна система (далі – ІС) не є обов'язково комп'ютерною. Наприклад, інформаційною системою можна вважати навіть звичайну бібліотеку. Але сучасне життя вимагає досягнення певного рівня ефективності роботи, швидкодії та зручності доступу до інформації, що може забезпечити тільки використання автоматики та цифрових методів роботи з даними. Відповідно, ***автоматизована інформаційна система – це взаємозв'язана сукупність даних, обладнання, програмних засобів, персоналу, стандартних процедур, які призначені для збору, обробки, розподілу, зберігання, представлення інформації згідно з вимогами, які випливають з цілей організації***. Серед задач, які покладаються на інформаційну систему, слід виділити:

- введення інформації;
- накопичення та систематизацію даних;
- збереження інформації;
- аналіз та представлення інформації;
- прийняття рішень.

Кожна з цих задач при реалізації автоматизованої (комп'ютерної) ІС розв'язується відповідно до потреб використання такої системи.

Введення інформації може включати як автоматичні засоби, які забезпечують введення даних, що надходять з апаратних пристроїв (приладів), що оточують ІС, так і через взаємодію з користувачами. Часто є потреба використовувати це одночасно. Наприклад, ІС керування транспортними потоками може використовувати автоматику введення координат знаходження транспортних засобів на основі GPS трекінга, але одночасно з цим передбачає ручний режим роботи з диспетчерами та іншим персоналом. Відповідно, вже виходячи з цієї задачі, виникає проблема локальності ІС, що у свою чергу призводить до їх розподілу на:

- локальні (усі компоненти розміщуються на одному комп'ютері, за яким працює користувач ІС);
- розподілені (ІС, орієнтована на обслуговування певної кількості користувачів, що як мінімум вимагає реалізації розподіленого інтерфейсу користувача, який зазвичай реалізується із застосуванням мережі).

Локальні ІС в більшості випадків вирішують задачі однієї людини, але можуть відповідати частині більш загальної колективної задачі. Розподілена ІС може бути реалізована як деяка множина окремих приблизно рівноправних складових, розміщених на різних комп'ютерах, але частіше за все значну частину функцій ІС виносять на деяку відокремлену частину ІС, яку називають її ***ядром***, що забезпечує спрощення адміністрування системи, збільшує рівень надійності та інформаційної безпеки. Наявність ядра значною мірою спрощує також проблему синхронізації даних.

Відповідно, одночасно з локальністю за ознакою архітектури можна визначити ще і розподіл за призначенням:

- персональна ІС (саме для вирішення завдань однієї людини);
- групова ІС відповідає невеликій робочій групі або підрозділу;
- корпоративна ІС охоплює групу необмеженого розміру, зазвичай використовується на рівні великого підприємства, державної або міжнародної організації, може використовуватись умовним (віртуальним) співтовариством людей у загальному інформаційному просторі.

Взаємодія з користувачами може бути реалізована або встановленням додаткового програмного забезпечення, специфічного для даної ІС, на робочому комп'ютері користувача (що відповідає використанню **клієнт-серверної архітектури**), або орієнтується майже виключно на функціональність ядра системи (архітектура **мейнфрейм-термінал**). Особливості цих архітектур будуть

детальніше розглянуті у п. 6.1.2. З врахуванням сучасних принципів організації інтерфейсу користувача, а саме використання графічного інтерфейса користувача GUI (graphic user interface) навіть в останньому випадку на робочому місці користувача потрібна певна обробка даних, які відносяться до даної ІС, а саме обслуговування графічних об'єктів GUI. Ці функції покладаються на деяке універсальне програмне забезпечення, зазвичай один з web-браузерів, що однак не робить загальну архітектуру клієнт-серверною. В останній час, правда, навіть при реалізації робочого місця користувача на основі браузера, частково попередня обробка інформації при взаємодії з користувачем виноситься на бік користувача, тобто границі між клієнт-серверною архітектурою та системою типа термінал–мейнфрейм дещо розмиваються.

Персональна ІС не потребує розділення ресурсів, але за рахунок використання уніфікованих рішень і в даному випадку може бути застосована клієнт-серверна архітектура, яку користувач взагалі може не помічати. Розробка групової ІС, як правило, з самого початку визначає структуру групи і умови використання. Це дозволяє на основі клієнт-серверної архітектури оптимізувати розподіл навантаження по складових системи. Корпоративну ІС краще будувати з web-інтерфейсом, що з точки зору розподілу навантаження саме за задачею обробки даних відповідає архітектурі мейнфрейм-термінал.

Накопичення та систематизація даних є окремою задачею, яка зазвичай вирішується залученням деякої готової програмної (іноді програмно-апаратної) складової – **системи керування базами даних** (СКБД), детальніше цю задачу розглянемо у п. 6.1.4, 6.1.5.

Збереження інформації в першу чергу повинно забезпечити цілісність даних, що напряму знову переноситься на рівень СКБД, які у свою чергу використовують певні самостійні програмні та апаратні рішення, такі як засоби резервного копіювання, журнальовані файлові системи, RAID тощо.

Аналіз та представлення інформації відповідає основному інтелектуальному навантаженню ІС. Детально зупинятись в даному курсі на реалізації цієї складової ІС немає особливого сенсу через те, що для цього використовується дуже широкий спектр методів, алгоритмів, готових бібліотек, засобів розробки, взагалі – значна частина усіх можливих засобів реалізації програмних продуктів. Зауважимо тільки, що представлення інформації в даному випадку зазвичай орієнтується на користувача та, відповідно, спирається на принципи реалізації GUI та графічне, іноді близьке до реалістичного представлення на основі математичного моделювання. В цьому

випадку є сенс як мінімум цю складову виконувати на комп'ютері користувача.

Прийняття рішень є дуже важливою складовою ІС, яка використовується для автоматизації керування складними апаратними засобами та управління великими виробничими та комерційними структурами. Ці задачі охоплюють дуже широке коло питань, в тому числі, наприклад, експертні системи та штучний інтелект. Відповідно, в даному курсі вони докладно розглядатись не будуть.

Дуже важливим моментом при розробці ІС, більших за персональну, є питання адміністрування, а саме задача сформувати програмний інструментарій для забезпечення інформаційної безпеки, в тому числі формування груп користувачів та визначення їх прав доступу, відстеження та блокування можливих атак з метою несанкціонованого доступу до даних та їх ушкодження. Особливо слід зазначити потребу у засобах протоколювання та аудиту, що є потужним засобом визначення причин збоїв системи, в тому числі через спробу зламу. Відповідно, експлуатація ІС вимагає залучення кваліфікованих адміністраторів та ретельного дотримання вимог безпеки.

6.1.2. Розподіл навантаження між складовими системи

Розглянемо довільну ІС. Для спрощення будемо розглядати ситуацію з коли довільна дія з даними ІС (контентом) відбувається за ініціативи користувачів. Зрозуміло, що користувачі мають різну активність (кількість запитів в одиницю часу), ця активність змінюється з часом. Але можна ввести деякий середній рівень запитів від користувача за одиницю часу. Виходячи з такої середньої активності та характерної ресурсоємності обробки одного запиту, можна визначити середню ресурсоємність усієї ІС. При цьому потужності одного обчислювального пристрою може не вистачати, через те що його ресурси в перерахунку на одного активного у даний момент користувача обернено пропорційні кількості користувачів. Якщо порівняти ресурси ядра при кількості активних користувачів, які можуть сягати сотень і тисяч, потужність клієнтської машини може бути навіть більшою, ніж потужність сервера в перерахунку на одного користувача. Таким чином, може бути доречно частину обчислювального навантаження виконувати на боці користувача.

Таке винесення частини навантаження сервера на клієнтську машину отримало назву клієнт-серверної архітектури. Ця архітектура прийшла на зміну раніше використовуваній ідеології мейнфрейм-термінал, яка відповідає в основному початковому рівню

комп'ютерних технологій. Оскільки в той час обчислювальна машина займала занадто багато місця, щоб наблизитись до користувача, і виникла потреба у віддаленому терміналі. Це зробило робоче місце зручнішим для користувача, відірвавши його від комп'ютера (мейнфрейму), дозволило збільшити ефективність пристосування комп'ютерних технологій до інших видів діяльності – наукової або інженерної роботи, статистичного економічного аналізу тощо.

Термінал в традиційному розумінні не має взагалі інтелектуальних функцій і фактично тільки об'єднує засоби введення інформації від користувача (з точки зору обчислювального процесу – стандартний потік вводу) і виведення отриманих даних (відповідно, стандартний потік виводу). Свого часу ці функції відігравали навіть телеграфні апарати та друкарські машинки.

Зараз ця архітектура застосовується для віддаленого доступу, зазвичай для адміністрування операційних систем, або в інших випадках, коли можна відмовитись від використання GUI. Зрозуміло, що зараз замість окремого пристрою з обмеженими термінальними функціями використовується довільний інший комп'ютер, на якому запускається спеціальна програма – емулятор термінала.

При розробці системи клієнт-серверної архітектури виникає потреба визначити границю розподілу дій між сервером та клієнтом. Така задача зазвичай не має однозначного рішення, а основними параметрами, за якими може бути виконана оптимізація, є:

- розрахункове навантаження на сервер;
- розрахункове навантаження на клієнта;
- мережевий трафік;
- інформаційна безпека системи.

Оптимізація за ознакою мінімізації навантаження на боці сервера або клієнта особливих питань не викликає і визначається завданнями системи та апаратним забезпеченням, на якому вона буде реалізована. В даному випадку архітектуру мейнфрейм-термінал можна розглядати як граничний випадок «полегшення» клієнта. Може використовуватись більш компромісний варіант, для якого застосовується термін «***тонкий клієнт***» (мінімалістична реалізація обладнання з боку користувача, але з виконанням певних нескладних функцій). Якщо цей термін використовується по відношенню до апаратної складової, мають на увазі недорогий максимально простий комп'ютер або спеціалізовану реалізацію (наприклад, касовий термінал). При цьому може навіть використовуватись комп'ютер без жорсткого диску з використанням завантаження ОС через мережу.

Зауважимо, що завдяки загальному зростанню швидкодії обчислювальних систем проблема недостатності ресурсів для

реалізації ІС може виникати тільки у випадку надресурсоємної обробки нових даних від користувачів, що буває досить рідко, або при умові надвеликої кількості користувачів, але це вже випадок корпоративної ІС, і його краще реалізовувати (відповідно до вимоги інформаційної безпеки) розподіленням ядра на декілька серверів.

Питання інформаційної безпеки призводять до менш однозначного розподілу функціональності між клієнтом та сервером. Воно повинне включати у себе безпеку доставки даних, що може призвести до часткового сповільнення трафіку та збільшенням навантаження як на клієнт, так і на сервер. Обмеження доступу до даних може вимагати відмови від буферизації на клієнтському боці, або, як мінімум, обмеження цієї буферизації. Слід зауважити, що використання широковживаного програмного забезпечення на боці клієнта спрощує несанкціонований доступ до усіх даних, що проходять через системи даного клієнта, особливо якщо взяти до уваги, що адміністрування усіх клієнтських комп'ютерів з тією ж ретельністю, яка зазвичай супроводжує адміністрування серверів, є надскладною задачею.

Щодо оптимізації мережевого трафіку, його зменшення може бути забезпечене максимально можливою обробкою даних перед відправкою, оскільки оброблені дані можуть біти зроблені значно компактнішими. Крім цього може бути застосована буферизація даних на пристроях, які обмінюються даними.

При розгляді розподілених ІС досить часто говорять про так звані Інтернет/інтранет-технології. Розглянемо цю технологію детальніше. Інтранет (буквальний переклад – внутрішня мережа) в даному випадку передбачає перенесення методів Інтернету на внутрішній рівень з метою спрощення розробки ІС за рахунок використання стандартних складових. В цьому випадку як основа забезпечення взаємодії компонентів ІС беруться широковживані протоколи – http(s), іноді разом із ftp(s), а для представлення інформації відповідні формати – HTML/XML. Така архітектура дозволяє як мінімум відмовитись від розробки комунікаційної складової, переклавши її функції на один із готових серверів. Більш того, при використанні розподіленого ядра комунікаційна система може бути перенесена на сервер, який виконує ці функції і для інших задач. Тобто замість розробки, відладки і встановлення спеціалізованого комунікаційного ресурсу можна використати вже працюючий в даній організації http- або ftp-сервер. Зазвичай така технологія орієнтується на «чистий» web-інтерфейс користувача, перевагами якого є:

- організація роботи користувача з довільного робочого місця без попередньої його підготовки;
- звичність робочого місця для непрофесійного користувача;
- легкість інтеграції послуг даної ІС в інші системи, при умові, що вони використовують аналогічну технологію;
- спрощення розробки за рахунок можливості використання наявного html-редактора для підготовки частини проекту, наприклад, підготовки шаблонів.

Але ці технології мають і певні недоліки:

- кодування інформації за допомогою мови розмітки, до якої відноситься HTML/XML, має великі накладні витрати і у багатьох випадках викликає ускладнення при потребі систематизації інформації;
- існує обмежений набір засобів прив'язки web-серверів до БД та інших допоміжних засобів;
- значна втрата швидкодії обробки даних на боці клієнта;
- загальне сповільнення роботи за рахунок надлишкової ресурсоємності відносно простих дій (табл. 6.1.).

Таблиця 6.1. Взаємодія клієнта і сервера за Інтранет/інтранет технологією

№	Операція	Виконує	Зауваження
1	Представлення обраної користувачем форми підготовки запиту	сервер	
2	Заповнення форми, відправлення CGI запиту	клієнт	невелике текстове повідомлення
3	Інтерпретація запиту	сервер	відносно повільна операція
4	Генерування динамічної HTML сторінки з даними для користувача	сервер	найбільш ресурсоємна операція
5	Відображення отриманої сторінки	клієнт	

Комунікаційну функцію ІС виконує web-сервер, на клієнтських місцях використовуються браузери. Уся обробка даних виконується на ядрі ІС. Формально, оскільки з обох боків працює деяке досить розвинене програмне забезпечення, можна віднести таку реалізацію до клієнт-серверної. Робоче місце користувача являє собою комплект web-сторінок (CGI-форм) із певним набором елементів керування. Браузер забезпечує відображення цього інтерфейсу у вигляді, який отримано з сервера, формує і передає запит користувача на сервер у

вигляді текстового рядка, тобто фактично виконує роль емулятора термінала. Таким чином, із точки зору використання обчислювальних ресурсів саме для потреб ІС, а не постійного перемальовування органів керування на клієнтському робочому місці, така ідеологія ближче до архітектури термінал-мейнфрейм. Деяку відмінність від цього граничного випадку може забезпечити використання для нескладної обробки інформації сценаріїв (скриптів), які виконує браузер клієнта.

У рамках платформи MS.Net можна виконати деяку компромісну реалізацію клієнта, функціонально більш глибоку, ніж викладена вище. Ця платформа, точніше сукупність засобів для її реалізації, включає у себе серед іншого набір розвинених бібліотечних класів для підтримки стандартних комунікаційних функцій, що дозволяє вбудовувати у довільну програму функції браузера або ftp-клієнта. Загалом, це допомагає створити спеціалізований інтерфейс клієнтського програмного забезпечення, комунікаційна частина якого буде відповідати Інтернет/інтранет-технології. При цьому, правда, в жертву приноситься можливість роботи з непідготовленого робочого місця.

Універсального варіанта визначення функціонального розподілу між клієнтом та сервером немає і, мабуть, не може бути. В кожному випадку є необхідність брати до уваги функціональні потреби ІС, умови, у яких вона буде використовуватися, передбачене навантаження на неї і навіть наявність ресурсів для розробки.

Потреба використання розподілення функцій ядра ІС знаходить своє відображення в ускладненні загальної архітектури клієнт-сервер. Часто кажуть про

- дволанкову;
- багатоланкову.

Дволанкова архітектура (англ. two-tier) ІС включає тільки два типи ланок – сервер, на якому повністю реалізоване ядро (back-end), і робочі станції, на яких знаходяться клієнтські програми (front-end). Багатоланкова архітектура (англ. multi-tier) ІС використовує ще проміжні ланки – сервери додатків (application servers), через які клієнтські програми взаємодіють з компонентами ядра – базами даних або загальними розрахунковими ресурсами. Наприклад, при реалізації ІС на основі web-технології роль сервера додатків може виконувати web-сервер.

6.1.3. Загальні принципи побудови баз даних

Дуже важливою складовою довільної ІС є підсистема забезпечення доступу до даних. При цьому для ІС є характерною велика кількість окремих операцій з відносно невеликими блоками, що загалом формує загальні ланцюжки дій з оброблюваною в системі інформацією. Це вимагає постійного пошуку необхідних для цих дій блоків даних (зазвичай за певним набором ознак). Для забезпечення високої ефективності такої роботи є потреба впорядкування загальної сукупності даних, тобто їх структуризації. Загалом, існують декілька моделей структуризації даних. До основних моделей належать:

- лінійна;
- ієрархічна;
- таблична (реляційна).

Лінійна структура даних відповідає «плоскому» невпорядкованому розміщенню даних (масив, список тощо). Така модель не вимагає особливих зусиль для своєї реалізації, але при масштабуванні досить швидко призводить до повільності пошуку. Як приклад такої моделі можна навести певну сукупність книжок, розкладених на одній дуже довгій поличці, одна поруч з іншою.

Ієрархічна модель передбачає певне багаторівневе вкладення (список списків, дерево тощо). На прикладі з книжками це набір шаф, розділених на полички, які в свою чергу впорядковані за певними секціями, у яких розташовані книжки (дані).

У багатьох випадках блоки даних зручно інкапсулювати у деяку умовну оболонку, розглядаючи кожне таке об’єднання як щось цілісне, тобто як об’єкт, а об’єкти можна зібрати в певні множини – класи, кожен з яких об’єднує усі об’єкти однакової структури. Аналогія з книжками в даному випадку є малопридатною. У випадку такої структуризації кажуть навіть про об’єктну модель.

Але для задач, у яких потрібна робота з великими обсягами даних, зазвичай на перше місце виходить не широка різноманітність класів та їх ієрархічна структурна взаємозв’язаність, а надзвичайно велика кількість елементів у класах, часто з появою тісних посилкових зв’язків між об’єктами різних класів. В цьому випадку клас може бути представлений як деяка таблиця, структура якої відповідає структурі полів класу, а об’єкти є рядками такої таблиці. Наприклад, дані телефонного довідника, який об’єднує сотні тисяч рядків, в кожному з яких внесена однакова за своєю структурою інформація про одного з абонентів. Це і є таблична або реляційна модель представлення даних.

При розробці технологій роботи з великою кількістю даних зазвичай використовують термін ***база даних (БД) – впорядкований набір логічно взаємопов'язаних даних, що використовуються спільно, та призначені для задоволення інформаційних потреб користувачів***. Відповідно до свого головного призначення БД повинна забезпечити як накопичення та гарантоване збереження певного обсягу даних, так і надання зручного доступу для маніпуляцій з ними. Зазначимо, наведене визначення не є однозначно пов’язаним з табличною формою, БД формально може спиратись і на іншу модель. Але зручність і ефективність табличного представлення забезпечили цій моделі надзвичайно широке розповсюдження у технологіях БД.

Термін БД часто супроводжується таким терміном як ***система керування базами даних (СКБД) — комп'ютерна програма чи комплекс програм, що забезпечує користувачам можливість створення, збереження, оновлення, пошуку інформації та контролю доступу до баз даних***. Використовують також скорочення СУБД, яке є точною копію російського варіанта (при цьому замість слова «керування» використовується «управління», яке в українській мові має трохи інше значення). СКБД надає сукупність інструментів для роботи з даними і для свого функціонування вимагає наявності однієї або декількох обчислювальних систем.

Локальними вважаються СКБД, розміщені на одному комп’ютері, як правило, разом з сукупністю даних. Розподілені СКБД охоплюють певну кількість систем, зв’язаних між собою інформаційними каналами, в цьому випадку дані можуть бути не локалізованими з відповідними підсистемами обробки, наприклад, за рахунок їх винесення на файл-сервери.

У файл-серверних СКБД файли даних розташовуються централізовано на файл-сервері. СКБД розташовується на кожному клієнтському комп'ютері (робочій станції). Доступ СКБД до даних здійснюється через локальну мережу. Нна даний момент ця архітектура вважається застарілою через велике навантаження на мережу.

Найбільш поширеною архітектурою вважається клієнт-серверна, при якій забезпечується розподіл функцій між окремими компонентами інформаційної системи за вимоги певної оптимізації, для БД частіше за все – зменшення навантаження на мережу, що є особливо актуальними при великих об’ємах даних та великій кількості операцій з ними.

Крім цього можуть використовуватись вбудовані СКБД, які входять як невід’ємна частина іншого (спеціалізованого) програмного

забезпечення, не вимагаючи самостійної інсталяції, і не розраховані на колективне використання.

6.1.4. Особливості реляційних баз даних

Теоретичним обґрунтуванням реляційної (від англ. relation – відношення) моделі БД є так звана реляційна алгебра, яка є реалізацією деякої алгебри на стику математичної логіки та теорії множин. Реляційна алгебра визначає певну структуризацію даних у вигляді «відношення», яке визначається як своєрідне зв'язування множин декартовим добутком, яке має просту геометричну інтерпретацію у вигляді таблиці (рис. 6.1), стовпці (поля, атрибути) якої відповідають входженням доменів у відношення, а рядки (записи, кортежі) – наборам з n значень, що взяті з початкових доменів. Кількість рядків n називають кардинальним числом відношення або потужністю відношення.

Рис. 6.1. Основні поняття реляційної моделі

Така таблиця має ряд базових властивостей:

- таблиця не може мати двох однакових рядків;
- стовпці відповідають атрибутам відношення;
- кожний атрибут має унікальне ім'я;
- порядок рядків є довільним.

Реляційна алгебра проголошує також основні властивості елементів цієї алгебри та операції над ними. Через деяку «надлишкову абстрактність» такої алгебри при практичному застосуванні БД використовують більш приземлені терміни. Замість більш загального терміна «кортеж» використовують ***«запис»*** як деякий умовно неподільний блок даних, за аналогією з ООП сукупність певної кількості даних (***«полів»***) різного типу. Сукупність таких записів, розміщених «один під одним», створюють регулярну ***«таблицю»*** за

рахунок однаковості структури усіх записів таблиці. Порядок слідування полів у записах такої таблиці є незмінним і визначає її вертикальну регулярність, створюючи однотипну ***«колонку»***, яка є сукупністю даних одного поля.

Відповідно, як і клас в ООП, перш ніж таблиця починає заповнюватись даними, вона повинна бути визначена саме за сукупністю майбутніх колонок, і кожна колонка повинна мати:

- спосіб виділення серед інших, що забезпечується унікальним (в межах даної таблиці) іменем;
- семантичне навантаження (тобто для чого саме використовуються її дані);
- тип (спосіб запису і відповідної інтерпретації даних).

Сукупність імен полів (неявним чином з їх типами) створюють ***заголовок таблиці*** (як і визначення класу в ООП), сукупність усіх даних є ***тілом таблиці***.

Дані можуть групуватись не в одну, а в декілька таблиць з можливістю накладання зв'язків на ці таблиці (що також трохи нагадує парадигму ООП). Але як один об'єкт повинен відрізнятись від іншого, так і запис також повинен бути унікальним, тобто повинен використовуватись деякий аналог імені (ідентифікатора) об'єкта. Роль такого ідентифікатора відіграє ***первинний ключ — атрибут (поле) або набір атрибутів, що однозначно ідентифікує кортеж (запис) даного відношення (таблиці). Первинний ключ повинен бути обов'язковим чином унікальним в межах даної таблиці***. У практичному використанні, зазвичай, обмежуються саме виділенням **одного** поля таблиці, як більш зручним і безпечним способом відстеження можливих помилок. Унікальність первинного ключа вимагається тільки в межах таблиці, оскільки повне посилання на запис може бути розширене додаванням посилання на таблицю (для чого використовується ***унікальне ім'я таблиці***).

Загалом вважається, що кожен запис відтворює інформацію про певну сутність навколишнього світу, що узгоджується з вимогою унікальності запису. В термінах реляційної алгебри це відповідає умові **цілісності сутності, відповідно до якої** сутність не може мати дублів, копія сутності фактично вже є новою сутністю. Таким чином, за такою логікою на перший погляд повинна визначатись вимога того, що одній сутності відповідає внесення інформації тільки в одну таблицю. Дійсно, формально можна звести задачу (мабуть, довільну) до набору таких незалежних таблиць. Але це є неефективним. Покращити ситуацію може використання декількох зв'язаних таблиць.

Для прикладу розглянемо реалізацію як запис в одній таблиці деякої банківської операції. В межах цієї операції переводиться певна грошова сума з рахунку одного клієнта на рахунок іншого. Звідси бачимо потребу прив'язки до інших сутностей – клієнтів. Крім цього, операція має відношення до оплати за якусь послугу чи товар, за правильність операції несе відповідальність певний працівник банку – ще прив'язки. В принципі, сукупність усієї інформації про кожну з перерахованих складових може бути реалізована як деякий набір полів у цій таблиці. При внесенні даної операції в базу (як запис) значення цих полів можуть братись з відповідних записів з інших таблиць (клієнти, працівники банку, послуги тощо), кожна з яких визначає свій (один) набір сутностей. Але той самий працівник відноситься до проведення багатьох операцій, один клієнт також може користуватись цим банком для багатьох дій. Відповідно, сукупність даних про одного працівника або про одного клієнта буда внесена у багато записів. В принципі, нічого страшного в цьому немає, але заміна будь-якого параметра (наприклад, адреси проживання клієнта) буде вимагати внесення виправлення у певній кількості записів таблиці. Крім того, дані про клієнта можуть мати потребу аналогічним чином вноситись і в інші таблиці (крім тієї, яку ми зараз розглядаємо). Більш ефективним замість копіювання даних в цьому випадку є внесення посилання.

Зв'язування таблиць відбувається таким чином, що в таблиці замість деякої сукупності полів утримується поле-посилання (зовнішній ключ), у якому в записі вноситься значення первинного ключа потрібних даних з іншої таблиці (рис. 6.2)

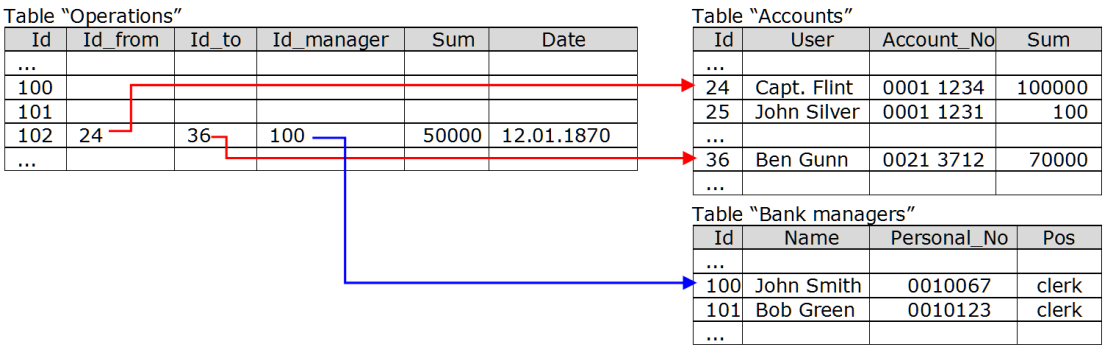


Рис. 6.2. Зв'язані таблиці

При цьому додатково до умови цілісності сутності з'являється умова **цілісності посилань –** обов'язковість того, щоб запису з певним значенням ключа-посилання обов'язково відповідав запис у

таблиці, на яку зроблене посилання, тобто є заборона порожнього посилання.

БД є динамічною структурою, її стан постійно змінюється у процесі роботи. Ситуація значною мірою ускладнюється тим, що з тими самими даними можуть працювати декілька користувачів одночасно, при цьому модифікуючи стан декількох таблиць в рамках однієї складної операції. При виникненні нештатних ситуацій це може викликати порушення цілісності. Щоб запобігти цьому, використовується підтримка транзакцій.

Транзакція (транзакція) – здійснення закінчених дій стосовно визначеного об'єкта, що переводить цей об'єкт з одного постійного стану в інший. Транзакція відповідає певній послідовності простих операцій, які розглядаються як єдине ціле Транзакція передбачає можливість перевірки правильності виконання при завершенні усієї послідовності. В разі негативного результату перевірки або переривання дій до її завершення вона вважається недійсною, а стан системи повертається до попереднього, що називається **відкатом транзакції.**

Потреба підтримки транзакцій є практично обов'язковою умовою для систем корпоративного рівня, але навіть для персональної СКБД вона є бажаною, оскільки завдяки такій підтримці в будь-яких умовах, навіть при наявності аварійних станів, **кожна транзакція починається при цілісному стані БД і залишає цей стан цілісним після свого завершення.**

Для СКБД є важливою можливість відтворити стан даних не тільки в разі аварійного завершення транзакції, а після довільного збою системи. Таку можливість відновлення стану даних дозволяє реалізувати деяка надмірність даних, що може бути реалізоване веденням журналу змін стану БД.

В інформаційних технологіях під терміном «журнал» (англ. «Log») розуміють особливу частину даних, яка є недоступною для користувачів. У СКБД журнал зберігається особливо ретельно, наприклад, двома копіями, на різних фізичних дисках. У журнал надходять записи про всі зміни основної частини БД. При різних реалізаціях докладність журналювання є різною. Наприклад, може записуватись рівень операцій типу видалення рядка, а в інших системах – модифікація сторінки пам'яті.

6.1.5. Забезпечення доступу до даних

Таблиця як спосіб забезпечення доступу до даних має значні переваги перед іншими форматами, оскільки через свою регулярність дозволяє виконувати:

- швидке позиціонування на початок довільного запису (зсув дорівнює різниці порядкових номерів рядків, помноженій на довжину запису);
- вибірку з таблиці за маскою по полях.

Для цього, звичайно, треба забезпечити регулярність довжини рядка протягом усієї таблиці. Оскільки з точки зору використання фізичного носія інформації таблиця не відрізняється від інших способів структуризації даних, зрозуміло, що і в пам'яті комп'ютера, і на зовнішньому носії (диску) таблиця записується як деякий файл (або декілька файлів), в якому окремі неподільні елементи даних (атоми) відповідають певному блоку бітів. Ці блоки можуть бути в неформатованому вигляді (тобто у тому виді, який використовує процесор для роботи з цими даними) або у наперед зазначеному форматі.

Для забезпечення регулярності доводиться при описі нової таблиці, тобто визначенні структури, яка відповідає рядку, жорстко задавати не тільки порядок слідування і типи простих змінних, які є елементами цієї структури, а й байтову довжину цих полів. Для числових полів останню вимогу задовольняє тип змінної при умові, що збереження у файлі виконується в неформатованому вигляді. В разі застосування текстового збереження у файлі необхідне додаткове обмеження розрядності. Для літерної інформації (тексту) потрібно задавати фіксовану довжину поля (кількість літер). Відповідно, це поле буде частково незаповненим, якщо довжина тексту є меншою, або частина інформації може бути втрачена (відрізана), якщо потрібно записати занадто довгий текст. Таким чином, при проголошенні таблиці доводиться орієнтуватись на максимально можливу для даного використання довжину рядка, що зазвичай призводить до деякого «розбухання» даних.

Типізація даних в межах технологій БД є схожою до тієї, яку використовують загальні мови програмування. Мінімальний набір, який задовольняє потребам обробки атомарних даних, включає:

- INTEGER - ціле значення,
- DECIMAL - дійсне значення (число із плаваючою комою),
- CHARACTER - текст обмеженої довжини,
- TEXT - текст довільної довжини (його називають також «коментар» або memo);

- LONG BINARY - бінарну послідовність довільної довжини.

Три перших типи відповідають знайомим із мов програмування. В різних СКБД дані аналогічних типів мають дещо різні назви, але це не принципово. Крім того, деякі СКБД на відміну від зазначених вище двох числових типів підтримують різну їх байтову довжину аналогічно до мов програмування.

Щодо поля типу TEXT, видно, що необмеженість довжини цього типу вступає у протиріччя з вимогою постійної в межах таблиці довжини рядка. Для забезпечення такої необмеженості це поле робиться посилковим, тобто у таблицю в поле фіксованої довжини заноситься посилання на дані, які вносяться у допоміжну інформаційну структуру (наприклад, як частина окремого файлу), для якої регулярність не є критичною. Аналогічним чином реалізується і тип LONG BINARY з тією різницею, що даними в цьому випадку є не текст, а довільна бінарна послідовність (зображення, звуковий потік, виконуваний файл тощо).

Багатьма СКБД підтримуються і деякі інші типи, призначенням яких є збільшення зручності роботи при розв'язанні задач, для яких зазвичай використовуються технології БД. З розширених типів, які в багатьох системах мають реалізацію, хоча не є стандартизованими, можна виділити:

- DATE - дата;
- TIME - час;
- BOOL - булеві дані;
- VARCHAR - текст невизначеної довжини, але що не перевищує обмежень даної системи керування (наприклад, 255 літер).

Розглянемо детальніше особливості типів DATE та TIME. Для запису дати достатньо використовувати дані цілого типу як деякий зсув від умовної дати, прийнятої за початок відліку. Квантом зсуву є доба. При використанні знакового цілого, що відповідає вже перерахованим раніше типам, навіть не є принциповим, яка саме дата береться за цей початок відліку. Так само і з часом, хоча якщо відокремити час від дати, найбільш логічним початком відліку часу є початок нової доби. За квант часу можна обрати проміжок, достатньо дрібний для забезпечення потрібної точності, наприклад, мілісекунду. Представлення користувачу дати та часу передбачає перерахунок у звичний нам календар та ієрархію годин, хвилин, секунд. Цей перерахунок з описаного способу представлення даних реалізувати досить просто з відповідною гарантією безпомилковості і однозначності. Але ситуація з введенням даних таких типів у звичній для людини формі може призвести до помилок через порушення

обмежень календаря або шкали годинника. Саме перевірку правильності введення даних та блокування спроби помилкового введення передбачають ці типи даних при реалізації засобами СКБД.

Основні операції з таблицею. До основних операцій, які зазвичай виконуються з таблицями БД, належать:

- створення таблиці;
- видалення;
- відкриття;
- закриття;
- додавання запису;
- видалення запису;
- зміна даних запису;
- індексація таблиці.

Створення таблиці. Зрозуміло, що не створивши таблицю, не можна ані додавати у неї дані, ані отримувати їх. Основною частиною операції створення таблиці є опис сукупності її полів, що нагадує опис класу в об'єктно-орієнтованому програмуванні. Загалом в БД можуть існувати таблиці однакової структури, але які відрізняються іменами.

Видалення таблиці, навпаки, є останньою операцію в життєвому циклі таблиці і використовується тільки у тому випадку, коли ані дані даної таблиці, ані структура таблиці як спосіб збереження даних стають непотрібними.

Відкриття таблиці. Таблиця одночасно є деякою умовною формою відтворення та регулярною структурою даних. Відповідно, вона має два принципово різні способи представлення:

- саме таблиця, як її бачить користувач;
- деякий файл, його частина або навіть декілька файлів, в яких зберігається у специфічному форматованому вигляді основна і допоміжна інформація таблиці.

Формування образу бази у вигляді, який є прийнятним для користувача на основі обробки файлового образу, зветься відкриттям таблиці. Відкриття таблиці супроводжується відкриттям її файлового образу. Зазвичай відкриття таблиці робиться з певною ***фільтрацією, що означає вибір тільки тих записів таблиці, які вдовольняють наперед заданій сукупності умов.*** Фільтрація є надзвичайно важливою при відкритті таблиці, оскільки загальна кількість записів може бути надзвичайно великою, але потрібними в даний момент зазвичай є тільки деякі з них.

Закриття таблиці. Відкриту таблицю (як, наприклад, і файл) після завершення необхідних операцій обов'язково треба закривати,

що завершує сеанс роботи з нею. Загалом, фактично закриття таблиці і супроводжується аналогічними файловими операціями.

Додавання запису. Створення нового запису у таблиці. Оскільки таблиця формально не є впорядкованою, про певне визначення місця нового запису мови не йде. Достатньо унікальності самого запису, що забезпечується відповідним значенням первинного ключа у доданому записі.

Видалення запису зазвичай реалізується без його фізичного видалення поміткою «видалено». Це дозволяє виконувати цю операцію максимально швидко, але з часом може призводити до значного накопичення «порожніх» рядків. Додавання ж нового рядка при наявності порожніх відбувається саме в один з таких «видалених» рядків.

Зміна даних запису. Є заміною даних довільних полів обраного рядка на інші значення.

Індексація таблиці. Для ефективного використання БД часто є потреба пересортування таблиці з впорядкуванням за окремим полем. Фізичне пересортування таблиці, особливо якщо взяти до уваги, що вона іноді має декілька сот тисяч і навіть мільйони записів, досить ресурсоємна задача. Тому замість цього використовується сортування даних тільки при відображенні (а також при пошуку даних перед відображенням системою керування). Для цього паралельно з таблицею формуються так звані індекси, що і є операцією індексації.

Індекс таблиці являє собою спеціальну інформаційну структуру типу бінарного дерева, у якому ліва гілка означає «менше», а права «більше». Елементи цього дерева (листя) є посилання на записи таблиці. При створенні Індексу створюється за кожним обраним полем таблиці окремо. При цьому по черзі обробляються усі записи таблиці, посилання на які вносяться як новий листок цього дерева. Таким чином, шлях по гілках виводить на даний листок за мінімальну кількість кроків. Завдяки цьому при відкритті таблиці є змога автоматичними засобами СКБД з мінімальною ресурсоємністю забезпечити вибір записів по черзі таким чином, що забезпечується впорядкування за полем, за яким побудований індекс.

В разі, коли необхідно доповнити таблицю новим індексом або оптимізувати структуру індексів після довгострокової активної роботи з таблицею, робиться реіндексація таблиці, при якій індекси формуються з самого початку послідовною обробкою усієї таблиці.

Іноді, після багаторазового видалення і запису до таблиці, використовується процедура пакування таблиці, яка зводиться до її перезапису з фізичним видаленням «видалених» рядків та впорядкуванням за первинним ключем. Така операція ресурсоємніша

за індексацію, крім того, вона вимагає і наступної перебудови індексів.

Візуалізації БД для потреб користувача. Для представлення даних, відфільтрованих з бази, у зручному для користувача вигляді застосовуються два види:

- таблиця;
- форма.

Форма являє собою деяке вікно, в якому відтворюються у вигляді текстових елементів поля таблиці, які візуалізуються. Для зручності користувача зазвичай кожне з полів має пояснювальний підпис (наприклад, його назву). Форма, як правило, відображає один (активний) запис із фільтрованої таблиці. Для перегортання записів таблиці (вибору активного запису з відфільтрованих) використовують операції:

- зсув на перший запис;
- зсув на один запис назад;
- зсув на один запис вперед;
- зсув на останній запис.

Відповідно, елементи керування, що відповідають таким операціям, також включаються у склад форми. Таким чином, формі відповідає прихована від користувача таблиця, один з рядків якої є виведеним на форму.

Методи доступу до даних. Довільна БД спирається на доступ до файлового образу при відкритті таблиці. І хоча користувач запитує тільки частину даних таблиці, при її відкритті деякий програмний компонент СКБД так чи інакше повинен виконати доступ до усього об’єму даних. Цей доступ може відбуватись або засобами локального накопичувача, якщо дані та обробник знаходяться на одному апаратному засобі (сервері або робочій станції), або з повною передачею усього образу через мережу, що може призвести до значних навантажень на мережу.

Персональна БД (тобто призначена для роботи одного користувача) зазвичай використовує дані, які знаходяться на тому ж комп’ютері або є доступними через локальну мережу з файл-сервера. База відкривається в монопольному режимі, її об’єм, як правило, невеликий, що робить файл-серверний режим не дуже критичним.

Групова база розраховується на загальне користування обмеженою кількістю користувачів. Для цього також є прийнятною файл-серверна архітектура, проте доводиться користуватись розділеним (одночасно багатьма користувачами) режимом відкриття файлів. Крім того, значно більший розмір таблиць сповільнює роботу локальної мережі у випадку файл-серверної реалізації. Тому вже при

такому рівні БД краще використовувати клієнт-серверну архітектуру, при якій сервер доступу до даних, який і виконує фільтрацію, знаходиться разом з даними.

БД корпоративного типу через великі об'єми даних та надзвичайно велику кількість запитів вимагає виключно клієнт-серверної архітектури.

Зазвичай користувачі використовують БД не прями чином, а через деяку спеціалізовану програму або ІС, навіть якщо ця ІС розробляється не окремими засобами, а інструментами, передбаченими в комплекті СКБД. Таким чином, безпосередньо з даними працює не користувач, а деяка його програма. Оскільки взаємодія програми з даними відбувається через певний набір файлів, формально можна забезпечити прямий доступ з програми до даних інтерпретацією формату файлу. Але існування багатьох внутрішніх форматів БД призводить до значних незручностей, особливо коли є потреба сумісного використання декількох різних баз. Розв'язати цю проблему можна використанням спеціального засобу – «**двигуна БД**», призначенням якого є узгодження специфічного формату даних з уніфікованим (незалежним від формату файлового образу) логічним інтерфейсом, через який довільна програма може звертатись до даних (рис. 6.3).

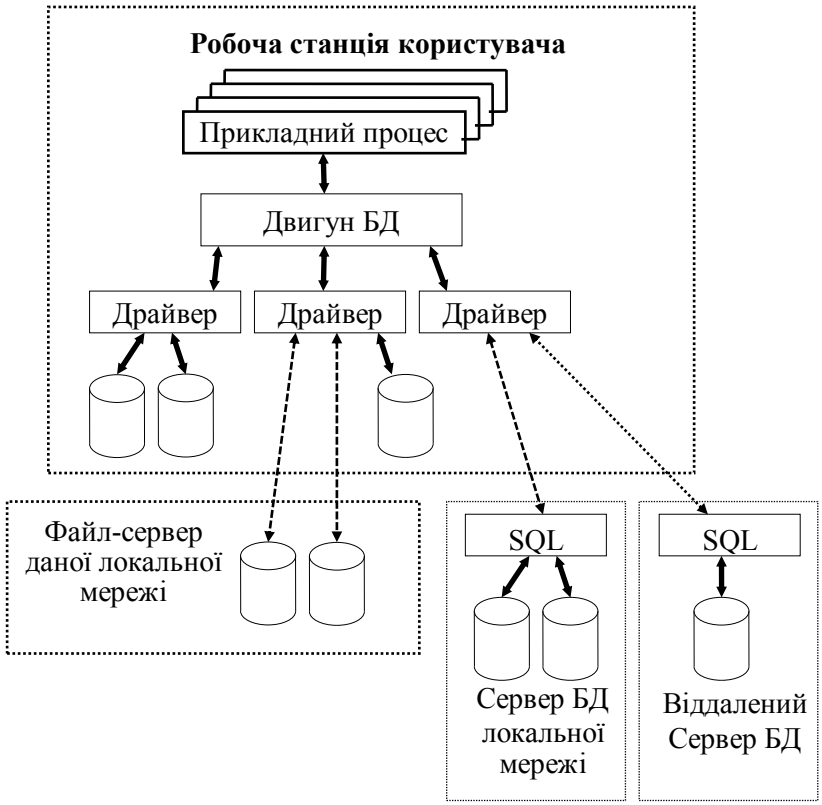


Рис. 6.3. Доступ до даних з використанням двигуна БД

При цьому для кожного з можливих форматів із метою забезпечення гнучкості конфігурування двигуна БД реалізується окремий драйвер (аналогічно узгодженню програмних засобів, наприклад, з принтером чи відеоадаптером різних типів). До цього можна додати використання спеціалізованих програмних засобів – SQL-серверів, які отримали таку назву завдяки використанню спеціальної мови SQL для реалізації логічного інтерфейсу доступу до даних. Саме такі сервери дозволяють відмовитись від розташування даних на файл-сервері і тим самим зменшити мережевий трафік. ***SQL - Structured Query Language (структурована мова запитів)*** – не тільки деяка мова, а й принципово нова ідеологія роботи з базами даних, зручна при віддаленому доступі, що особливо важливо при використанні для потреб телекомунікації і у розподілених інформаційних системах

6.1.6. Основи SQL

SQL – загальновживана зараз мова, яка використовується як стандартний інтерфейс як до БД, так і до двигуна БД. Ця мова була створена на початку 1970-х років в рамках проекту експериментальної реляційної СУБД System R в одній з дослідницьких лабораторій компанії IBM. Перша назва мови для цієї системи була SEQUEL, розшифровувалася як Structured English QUEry Language. З часом цю назву скоротили до SQL, оскільки у проекту з'явились комерційні перспективи, а торгова марка «SEQUEL» виявилась вже зайнятою авіабудівниками, хоча нове скорочення так і залишило повну фонетику першої назви аж до сьогодення.

У 1986 році ANSI (American National Standards Institute) затвердив перший стандарт SQL. У 1979 році були офіційно реалізовані перші СУБД на основі використання нової мови – System/38 і System/R (IBM) та Oracle V2 для комп'ютерів VAX. Розробник V2 – компанія Relational Software Inc. – згодом перетворилась у загальновідому Oracle. У 1986 році інша, тепер вже міжнародна, організація ISO затвердила стандарт цієї мови – SQL-86. Зараз актуальним стандартом є SQL:2003 з невеликими модифікаціями, доданими пізніше.

На поточний момент є багато досить потужних СКБД, основою яких є SQL, серед яких:

- Oracle;
- Microsoft SQL Server;
- Borland InterBase;
- MySQL;

- PostgreSQL.

Крім того, як вже зазначалось у попередньому параграфі, існує потреба уніфікації логічного інтерфейсу звернення застосувань до двигунів БД, для чого також успішно використовується ця мова. Як приклади таких двигунів можна навести:

- ODBC (Open DataBase Connectivity) — відкритий інтерфейс доступу до баз даних, розроблений консорціумом X/Open;
- DAO (Data access object) – об'єкт доступу до даних;
- OLE DB (Object Linking and Embedding, Database), API компанії Microsoft, що включає у себе набір інтерфейсів, реалізованих за допомогою Component Object Model (COM), розглядається як перспективна заміна ODBC.

При роботі з базами даних можна визначити такі групи функцій, як конфігурування бази, безпосередньо робота з даними і адміністрування. Виходячи з цього SQL можна структурувати на три розділи, які зазвичай відносять до мов:

- DDL (Data Definition Language) – мова визначення даних;
- DML (Data Manipulation Language) – мова маніпулювання даними;
- DCL (Data Control Language) – мова керування даними.

DDL. У цьому розділі зібрані команди, які утворюють у базі даних об’єкти (таблиці, індекси і т.д.) та видаляють їх.

Для утворення таблиці використовується команда

```
CREATE TABLE < назва таблиці >
(< назва поля >  < ініціалізація поля >,
 < назва поля > < тип даних > < значення ініціалізації >
... );
```

наприклад:

```
CREATE TABLE Customer
( Name char (50),
N2 char (20) ,
N3 char (20) ,
ID integer,
Street char (30),
House integer,
Apartment integer );
```

Тобто визначається спосіб ідентифікації таблиці (її ім’я) і список полів таблиці, для кожного з яких мінімально визначається тип та ім’я. За потреби у дужках вказується довжина у байтах (для

текстового поля), також за потреби додається початкове значення. Вже за цією командою можна побачити, що при умові надання інтуїтивно зрозумілих імен таблиці та полям, як і у випадку загальних мов програмування, мова SQL дозволяє створювати досить зрозумілі вирази, які приблизно відповідають звичайній мові. Зазначимо, що зроблене у прикладі розбиття на рядки є необов'язковим. Крапка з комою використовується аналогічно до мов програмування як роздільник команд.

Для того, щоб видалити існуючу таблицю, виконується команда:

```
DROP TABLE < ім'я таблиці >;
```

Для створення індексу даної таблиці (ідентифікується за її іменем, яке включається у команду) за заданим полем використовується команда:

```
CREATE INDEX < ім'я поля > ON < ім'я таблиці>;
```

DML. Маніпулювання з даними відповідає найбільшій частині взаємодії з БД. Це пов'язано з тим, що одноразово створена таблиця заповнюється даними (саме виконанням деяких команд) та постачає накопичені дані для інших операцій. Відповідно, майже вся робота з базою відбувається в рамках подібних запитів. Від ефективності відкриття даних залежить ресурсоємність подальшої обробки цих даних, оскільки саме на цьому етапі виконується вибірка потрібної в даний момент підмножини записів з усього обсягу таблиці, а надлишкові дані доведеться фільтрувати додатково.

Далі вважаємо, що таблиці, з якими будемо мати справу, вже утворені. Раніше було зазначено, що для отримання доступу до деякої таблиці її треба відкрити, попередньо сповістивши СКБД, тобто в даному розумінні сервер, про умови відкриття таблиці. Сама таблиця ідентифікується її іменем, але на відміну, наприклад, від файлу, для відкриття якого ім'я надає достатню інформацію для виконання операції, в даному випадку потрібні умови фільтрації, яка визначає, які саме дані потрібні користувачу. Фільтрація може виконуватись:

- по полях, які відкриваються;
- по значенням полів.

Для відкриття використовується команда SELECT. Наприклад,

```
SELECT Name, N2, N3, ID FROM Customer;
```

В даному прикладі використане пряме перерахування полів таблиці, які будуть відкриті. Інші поля у потоці даних, який формує сервер при відкритті таблиці, ігноруються. Для таблиць із великою кількістю

полів це дуже корисно, оскільки дозволяє значною мірою скоротити довжину рядка, що відображається, та, відповідно, і повний розмір підтаблиці, яка передається з сервера (трафік!). Якщо потрібно відкрити усі поля таблиці, можна задати це наступним чином:

```
SELECT * FROM Customer;
```

Зазначені запити поки що відкривають усю таблицю цілком. Ще раз нагадаємо, що цього краще не робити. Обов'язково, особливо при використанні віддалених баз, треба задавати фільтр. Найпростішим способом задати такий фільтр є обмеження числового діапазону даного поля використанням операторів співвідношення даних (> < = >= <= <>) :

```
SELECT Name, N2, N3, ID FROM Customer
WHERE ID>1000 ;
```

В даному прикладі діапазон значень поля ID обмежений з одного боку. Якщо треба використати більш складну умову, аналогічно мовам програмування будується логічний вираз використанням булевих операторів, які в даній мові мають синтаксис AND, OR, NOT:

```
SELECT Name, N2, N3, ID FROM Customer
WHERE ID>1000 AND ID<=1100 ;
```

Трохи складнішою є ситуація, коли треба відфільтрувати дані текстового типу. В даному випадку наведений спосіб відсічки частини області значень не має сенсу. Зазвичай в тексті шукають деяку відповідність за фрагментом самого тексту, наприклад, його початком або закінченням. Задати маску пошуку дозволяють так звані регулярні вирази, для написання яких створюється власна міні-мова, тобто домовленість про те, якими символами визначаються недетерміновані літерні структури і які саме (довільна літера, довільне продовження тощо). Нажаль, і у таких домовленостях, які використовуються у різних ділянках інформаційних технологій, існують деякі розбіжності. За стандартом SQL, наприклад, довільне продовження задається символом % (знак відсотка). В MS Access замість знака відсотка використовується зірочка.

```
SELECT * FROM Customer
WHERE ID>1000 AND Name='Іва%';
```

```
SELECT * FROM Customer WHERE ID>1000 AND
( Name='Іва%' OR Name='Сидо%' );
```

```
SELECT * FROM Customer WHERE ID>1000 AND
NOT Name='Іва%';
```

В першому випадку будуть відкриті дані про замовників, які починаються з «Іва. . . » (Іваненко, Іванов, Івашкін тощо).

У другому прикладі додатково приєднаються дані про замовників із прізвищами, що починаються з «Сидо. . . ». Звертаємо увагу, що в даному випадку робиться запит з відкриттям усіх полів таблиці.

Другий та третій приклади показують комбіновані фільтри за ідентифікаційним номером та прізвищем (номер повинен бути більше 1000) та додаються обмеження на прізвище. Зверніть увагу на можливість для визначення пріоритету операцій використовувати дужки.

При фільтрації за текстовим полем замість знаку рівності використовувати слово LIKE, а крім розглянутого вище довільного продовження можна в умову фільтрації додати ще інші недетерміновані частини тексту. Наприклад, знак відсотка крім довільного продовження може означати ще й довільний початок:

```
SELECT * FROM Customer WHERE Name LIKE 'Д_ _б%';
```

```
SELECT * FROM Customer WHERE Name LIKE '%убо%';
```

```
SELECT * FROM Customer WHERE Name LIKE 'І%в';
```

В даному прикладі '_' позначає довільну літеру (точніше байт з повного позначення літери).

Щоб вибрати записи, для яких поле приймає невелику кількість значень зі списку, особливо, коли це трудно задати аналітично, можна написати:

```
SELECT * FROM Customer WHERE ID IN
(1001,1012,1121);
```

Або для текстового поля

```
SELECT * FROM Customer
WHERE Name IN ('Іванов', 'Петренко', 'Сагайдачний');
```

замість більш довгого

```
SELECT * FROM Customer
WHERE Name='Іванов'
OR Name='Петренко'
OR Name= 'Сагайдачний';
```

Якщо параметр має двобічне обмеження, можна також трохи спростити вираз фільтра:

```
SELECT * FROM Customer
WHERE ID BETWEEN 1012 AND 1030 ;
```

замість

```
SELECT * FROM Customer
WHERE ID>= 1012 AND ID<=1030 ;
```

Тепер можна спробувати відкрити дві таблиці, наприклад, замовників разом з їх замовленнями. В даному випадку є потреба вказування двох таблиць за їх іменами (ось де потрібна унікальність імені таблиці, про яку ми вже говорили раніше). Перелік полів при цьому має певні особливості. У випадку, коли ім'я поля зустрічається тільки в одній з таблиць, що відкриваються, можна використовувати тільки ім'я цього поля, якщо ж це ім'я використане для позначення певного поля в обох таблицях, однозначність ідентифікації забезпечує додавання імені таблиці:

```
SELECT Name, N2, Customer.N3, Customer.ID, Order.N3,
AP
FROM Customer, Order
WHERE Customer.ID = Order.ID;
```

Особливу увагу треба звернути на частину виразу

```
...
WHERE Customer.ID = Order.ID;
```

Якщо її опустити, то виникне дуже велика помилка. У випадку відкриття однієї таблиці пропуск фільтра означав відкриття усіх її записів, що неефективно, але не є формальною помилкою. В даному випадку відсутність правильно написаного фільтра буде означати об'єднання рядків таблиць «кожен з кожним», тобто якщо, наприклад, у таблиці Customer буде 500 записів, а в таблиці Order 2000, то результат відкриття без наведеного фільтра сформує 1000000 (500x2000) рядків! Більш того, більшість з цих рядків не буде мати сенсу, оскільки буде зчіплювати замовника з замовленням, яке він не робив.

Тобто ***при відкритті двох чи більше таблиць потрібно обов'язковим чином задавати вираз зв'язування таблиць через значення відповідних ключів*** (на кшталт наведеного у прикладі).

Зрозуміло, що аналогічним чином можна за потреби відкрити і більшу кількість зв'язаних таблиць.

Якщо в таблиці є повторення і користувач не хоче їх бачити більше одного разу, запит можна модифікувати за допомогою команди DISTINCT (при цьому опускаються усі «зайві» рядки, тобто записи, для яких уся сукупність вибраних параметрів співпадає з іншими, що вже відібрані). Наприклад, ви хочете отримати інформацію, що саме замовляв даний замовник, але саме що, а не скільки разів (тобто без можливих повторень кожного варіанта замовлення):

```
SELECT DISTINCT ProductName FROM Order
WHERE Customer.ID = 1010;
```

При відкритті таблиці часто є потреба впорядкувати відображення за якимось із полів. Це робиться таким чином:

```
SELECT *  FROM Customer
WHERE Name LIKE 'Д %'
ORDER BY ID;
```

```
SELECT *  FROM Customer
WHERE Name LIKE 'Д %'
 GROUP BY Street, House, Apartment;
```

На відміну від першого в другому випадку виконується впорядкування за декількома полями за пріоритетами відповідно до порядку у списку полів після GROUP BY.

Може виникати необхідність не відкривати таблицю, а розрахувати кількість рядків, які вдовольняють певній умові або середнє значення поля в усій таблиці. Звичайно, можна відкрити базу з потрібною умовою і засобами, які пропонує клієнтське програмне забезпечення, отримати такий параметр. Але з точки зору мережевого трафіку і, відповідно, часу очікування це нераціонально. Тому в SQL було передбачено можливість отримувати такі дані навіть без передачі таблиці на клієнтську станцію. Це робиться використанням так званих агрегатних функцій COUNT, SUM, AVG, MAX, MIN (відповідно – кількість, сума, середнє значення, максимальне і мінімальне значення). Наприклад, отримаємо кількість замовників, які замовляли деякий конкретний товар:

```
SELECT COUNT (ProductName ) FROM Order
WHERE ProductName =’Asus Nexus 7’;
```

Для отримання загальної кількості рядків у таблиці можна написати вираз

```
SELECT COUNT (* ) FROM Order;
```

SQL дозволяє винести на сервер і більш складну обробку, для чого один запит інкапсулюється в інший. Це також має призначення мінімізувати трафік і розвантажує робочу станцію, на якій працює програмне забезпечення користувача:

```
SELECT *  FROM Customer
WHERE ID
( SELECT DISTINCT  CustomerID
 FROM Order
 WHERE ProductName ='Apple iPhone 5');
```

При цьому у внутрішньому запиті отримується значення, яке використовується в зовнішньому. В даному прикладі спочатку буде отримано список замовлень (з вилученням повторень, оскільки використано DISTINCT), і за цим списком відкриваються замовники, які саме замовляли цей товар.

Таким чином, ми отримали достатньо потужний засіб відкриття таблиць з фільтрацією і обрахунку деяких інтегральних параметрів таблиці без їх відкриття.

Однак є ще важливі функції, без яких робота з базою даних неможлива. Це внесення нових даних, модифікація та видалення вже існуючих. Такі операції виконуються за допомогою команд INSERT, UPDATE , DELETE. Для того, щоб додати новий запис, можна написати:

```
INSERT INTO Customer
VALUES ('Панченко', 'Юхим', 'Сидорович', 45,
'Заболотного', 12, 34);
```

(див. структуру таблиці Customer на початку параграфу). Якщо треба в даному списку деякі з полів залишити вільними, замість них використовується NULL:

```
INSERT INTO Customer
VALUES ('Johnson', 'David', NULL, 95,
'Long Beach', 1102, NULL);
```

В таблицю можна вставити вибірку з іншої таблиці:

```
INSERT INTO VIP_Customer
SELECT *
```

```
FROM Customer
WHERE ID =1000;
```

Видалення рядків виконується виразом:

```
DELETE FROM VIP_Customer
WHERE ID = 1050;
```

Для зміни значення довільних полів використовують команду UPDATE з переліком відповідних полів і їх нових значень. Для прикладу розглянемо ситуацію, коли замовник змінив адресу:

```
UPDATE Customer
SET Street='Васильківська', House =25, Apartment=12
WHERE ID = 1014;
```

Для модифікації можна використовувати навіть вирази:

```
UPDATE Warehouse
SET Counter = Counter+1000
WHERE ProductName = 'HTC One X';
```

У даному прикладі у таблиці Warehouse (склад) при отриманні нової партії товару модифікована кількість одиниць цього виду товару.

DCL. Для керування доступу до даних (тобто для адміністрування) потрібні засоби, які визначають права користувача на виконання окремих дій (отримання, додавання, редагування, видалення тощо). Ідентифікація користувачів в середовищі СУБД забезпечується використанням умовних імен. Для формування дозволу на окремі методи роботи SELECT, INSERT, UPDATE, DELETE, REFERENCES (відповідно відкриття, додавання нових записів, редагування, видалення, визначення зовнішніх ключів) використовується команда GRANT (допуск), а для заборони – REVOKE (відміна). Наприклад:

```
GRANT INSERT ON Customer TO Dmitrenko;
```

```
GRANT SELECT. INSERT UPDATE DELETE
ON Personal TO SysAdmin,  Director;
```

```
GRANT UPDATE (ProductName, ID)
ON Warehouse TO Director;
```

```
REVOKE INSERT ON Order TO Sidoriv;
```

Зазначимо, що в даному параграфі були описані далеко не усі можливості сучасного SQL, навіть у рамках ANSI обмеження. Метою даного розділу було тільки ознайомлення з основними методами роботи, синтаксисом побудови запитів із використанням найбільш популярних команд. Проте навіть із наведеного можна побачити, наскільки зручним і потужним є текстовий інтерфейс до серверів баз даних, особливо якщо взяти до уваги забезпечення сумісності за цим інтерфейсом основних послуг серверів різного типу. Саме це забезпечило SQL технології місце інтерфейсу зв'язку з даними навіть у випадках, коли не використовується клієнт-серверна архітектура програмного забезпечення. Крім того, SQL технологія непогано інкапсулюється в інше програмне забезпечення, в тому числі має непогану підтримку у MS.Net.

6.2. Інформаційна безпека

6.2.1. Принципи інформаційної безпеки

Значна частина видів діяльності людей так чи інакше пов'язана з потребою у забезпеченні **цільового** доступу до інформації, тобто потребує вирішення двох взаємозв'язаних проблем:

- забезпечити доступ до певної інформації особі, яка повинна цю інформацію отримати;
- запобігти потраплянню інформації до осіб, яким доступ до неї є небажаним.

При цьому розв'язання цих проблем має на меті забезпечити три формальні властивості інформації безвідносно до її джерела походження або носія:

- конфіденційність, тобто безпосередньо обмеження доступу;
- цілісність, а саме неможливість несанкціонованої зміни;
- доступність (санкціоновану).

Зауважимо, що ці задачі виникли задовго до появи комп'ютерів і протягом усієї історії людства викликали жвавий інтерес як з боку тих, хто робить спробу отримати чужу інформацію, так і з боку тих, хто цьому запобігає. Віками ці задачі супроводжували паперовий носій інформації і виробили як витончені способи обмеження доступу (міцні скрині зі складними замками, тобто сейфи, шифри, невидимі чорнила), так і засоби запобіганню підміні інформації (підписи, печатки, водяні знаки, спеціальний папір тощо). До речі, усі перераховані засоби зараз мають своєрідне втілення і в сучасній інформатиці. Це пов'язано з тим, що сучасні інформаційні технології,

засновані на цифрових методах обробки даних, взагалі досить часто розробляються як своєрідні аналоги того, що використовувалось до появи комп'ютерів. Інформаційна безпека, звичайно, не є виключенням. При цьому цифрова інформатика не тільки успадкувала усі проблеми захисту, накопичені віками, а й додала до них низку нових.

Проблеми інформаційної безпеки охоплюють дуже широке коло задач і методів і для глибокого вивчення вимагають окремого навчального курсу. Відповідно, в рамках даного курсу комп'ютерних технологій ці питання можуть бути розглянуті тільки в плані ознайомлення з певними принципами та шляхами реалізації.

В загальному випадку (безвідносно до комп'ютерних технологій) ***інформаційна безпека (англ. Information Security) – стан інформації, в якому забезпечується збереження визначених політикою безпеки властивостей інформації.*** Під політикою інформаційної безпеки розуміють сукупність вимог, правил, обмежень, рекомендацій, які регламентують порядок інформаційної діяльності. Відповідно, задачі інформаційної безпеки можуть вирішуватись:

- на законодавчому рівні;
- організаційними заходами;
- апаратним забезпеченням;
- програмними засобами.

В рамках даного курсу цікавими є саме останні засоби і тільки іноді (за крайньої необхідності) будемо торкатись інших питань.

Перш ніж почати знайомитись із тим, як забезпечити захист інформації, слід чітко визначити причини, які заохочують до несанкціонованого доступу до інформації. Одразу зауважимо, що вимоги інформаційної безпеки забезпечуються на рівні

- держави;
- підприємства;
- особи.

Рівень держави в основному є політичним і охоплює міждержавні відносини. Відтак певні політичні негаразди призводять до спроб втручання у справи інших держав, для чого можуть залучатись кошти, які обмежуються хіба що фінансовими можливостями держави, що і визначає ресурси засобів зламу та потреби необхідного захисту від втручання. Зазвичай більш потужні держави в цьому плані мають беззаперечну перевагу.

Рівень підприємства в плані інформаційної безпеки підігрівається жорсткими умовами господарчої конкуренції, навіть виник цікавий вид діяльності, своєрідний аналог шпигунства

державного рівня – промислове шпигунство. Боротьба за інформацію має під собою чітке економічне підґрунтя – за отримання інформації є сенс платити не більше, ніж та вигода, яка може бути отримана від використання цієї інформації. Звідси витікає один з можливих критеріїв достатності рівня захисту.

Особиста інформаційна безпека є найбільш невизначеним рівнем. Вона спирається на елементарне бажання захиститись від чужих очей, з іншого боку провокує на надлишкову допитливість. Особиста інформаційна безпека може вступати у взаємодію з рівнями держави (представники якої іноді, навіть не виконуючи вимоги законодавства, втручаються в особисту інформацію) або підприємства (не завжди чесного по відношенню до клієнтів або найманих працівників). Відповідно, на цьому рівні доцільність та достатність засобів захисту взагалі визначити важко.

Загалом, основні причини спроб несанкціонованого доступу є такими:

- несанкціоновані фінансові операції (безпосередня швидка вигода);
- промислове і політичне шпигунство;
- отримання інформації для шантажу;
- диверсія відносно конкурента, помста;
- надмірна цікавість, “чесне” хакерство;
- хуліганство.

Несанкціонований доступ до інформації передбачає певні витрати ресурсів (грошей, обладнання, часу). Його результат визначається потенціальною вигодою, але отримання інформація зазвичай відбувається з певною затримкою, що може зменшити і навіть взагалі звести до нуля актуальність (опосередковано і вигоду). Відповідно, заходи щодо захисту головним чином передбачають саме вплив на ці три показника ресурсоємності зламу, додаючи ресурсоємності штатному використанню інформації.

У літературі пропонується наступна класифікація засобів захисту інформації [40].

- засоби захисту від несанкціонованого доступу:
    - засоби авторизації,
    - мандатне управління доступом,
    - вибіркове управління доступом,
    - управління доступом на основі ролей,
    - журналювання (аудит) ;
- системи аналізу та моделювання (CASE-системи) ;
- системи моніторингу мереж:

    - системи виявлення й запобігання вторгненням (IDS / IPS),
    - системи запобігання витокам конфіденційної інформації (DLP-системи) ;
- аналізатори протоколів;
- антивірусні засоби;
- міжмережеві екрани;
- криптографічні засоби:
    - шифрування,
    - цифровий підпис;
- системи резервного копіювання;
- системи безперебійного живлення:
    - джерела безперебійного живлення,
    - резервування навантаження,
    - генератори напруги;
- системи автентифікації:
    - пароль,
    - ключ доступу (фізичний або електронний),
    - сертифікат,
    - біометрія;
- засоби запобігання зламу корпусів і крадіжок устаткування;
- засоби контролю доступу в приміщення;
- інструментальні засоби аналізу систем захисту.

Далі розглянемо ту частину цього списку, яка стосується використання програмних засобів обробки інформації для несанкціонованого отримання інформації та його запобігання.

6.2.2. Обмеження доступу до даних

З врахуванням цифрового характеру сучасних методів роботи з інформацією для несанкціонованого використання або змінення інформації потрібно в першу чергу отримати доступ до самих даних або скопіювати дані, які відповідають шуканій інформації.

У разі використання паперових технологій носій зазвичай зберігали або розміщали при транспортуванні у місцях, злам яких є ускладненим (сейф, банківський бронеавтомобіль, наявність озброєного охоронника тощо) або його місцезнаходження є прихованим (тайник). При цьому отримання носія передбачало крадіжку (подія, що достовірно з’ясовується володарем носія) або копіювання дані з носія, наприклад фотографуванням (що технічно дещо складніше).

У комп'ютерних технологіях доступ до даних у багатьох випадках стає технічно більш простим і протистояти йому значно важче. Дані зазвичай отримуються копіюванням, відповідно є достатнім:

- отримати доступ до запам'ятовуючого пристрою;
- отримати доступ до процесу передачі даних;
- відгалузити частину аналогового сигналу, який є носієм при передачі інформації на деяку відстань.

Для доступу до інформації, що зберігається на довільному комп'ютері (в тому числі сервері), є достатнім запустити на ньому спеціальний програмний засіб. Часто такий засіб може охоплювати надзвичайно велику кількість комп'ютерів, розповсюджуючись застосуванням вірусних технологій. Така програма може виконувати відбір та відправку через мережу потрібної зламнику інформації (особливо небезпечним є перехоплення персональних та автентифікаційних даних), може забезпечити отримання від зламника та запуск інших програмних засобів. Крім того стає можливим використання даного комп'ютера для участі в масованих атаках на інші (незламані) системи для порушення їх працездатності або як допоміжний засіб їх зламу. Довільний програмний засіб може бути запущений або інстальований тільки вручну через команди ОС або за допомогою іншого програмного засобу, який вже працює. Тобто несанкціоноване встановлення або інсталяція такого засобу (на ще незламаному комп'ютері) може відбуватись:

- безпосередньо діями зламника (для цього потрібен безпосередній доступ зламника до системи через незачинений сеанс);
- помилковими діями користувача (для цього засіб маскується під інші загальновживані засоби або навіть деякий блок інформації);
- використанням програмних помилок у системах автоматики широковживаного програмного забезпечення, в тому числі ОС.

Відповідно, як захист від подібної спроби крадіжки даних повинні застосовуватись:

- дотримання рекомендацій щодо збереження автентифікаційної інформації;
- сучасне антивірусне програмне забезпечення з актуальною базою даних відомих вірусів;
- блокування зайвих, тим більш невідомих, процесів та сервісів (особливо на серверах);

- відмова від використання неперевіреного або отриманого з ненадійного джерела програмного забезпечення;
- використання захищених мережевих протоколів (https замість http; ssh замість telnet; ftps замість ftp);
- мережеві екрани, проксі, трансляція адрес для запобігання прямої адресації до комп'ютера.

Багато уваги приділяється забезпеченню доступу до певних програмних засобів (як віддалено, так і локальним чином) тільки певній групі користувачів. Зазвичай використовується ідеологія сеансу, тобто надання доступу до певних дій з системою з зазначеного пристрою протягом певного проміжку часу. Для старту сеансу відбувається авторизація, для якої частіше за все використовуються два слова:

- обліковий запис (login) – несекретне слово, яке однозначним чином ідентифікує даного користувача;
- пароль – секретне слово, яке відіграє роль ключа авторизації.

Програмна реалізація цілісності сеансу та перевірки авторизації за своєю логікою є досить простою і особливих проблем у розв'язанні не викликає. Єдине, на що треба звернути певну увагу – запобігання її крадіжці, для чого необхідно забезпечити збереження бази паролів у захищеному вигляді та у випадку віддаленої авторизації уникати передачі пароля у відкритому вигляді.

При розробці програмного забезпечення системи автентифікації треба додатково до цього звертати увагу ще одну небезпеку – загрозу сканування, тобто автоматичного підбору спеціальною програмою паролів із деякої бази або послідовною генерацією пароля. Боротись з такою загрозою досить просто. Для цього може використовуватись:

- обмеження кількості спроб із подальшим блокуванням авторизації (цей метод використовується, наприклад, у банкоматах);
- примусова часова затримка кожної спроби (в тому числі з послідовним зростанням паузи при кожній новій спробі);
- збільшення довжини пароля та його «нестандартність» (це стосується безпосередньо користувачів) та настоювання системи авторизації на несприйняття слабких паролів при реєстрації нового паролю (задача адміністраторів).

У сучасних системах замість або в доповнення до введення пароля можуть використовуватись спеціальні апаратні та програмні засоби для зчитування ключа з usb-накопичувача або картки, зчитування біометричних даних.

Трохи складнішою є задача забезпечення цілісності віддаленого сеансу без спеціалізованого клієнта (програми, розміщеної на комп'ютері користувача), наприклад, з реалізацією такої взаємодії з користувачем на основі тільки універсального браузера. При цьому робота з сервером виконується через web-інтерфейс, що вимагає в рамках сеансу багатьох перемикань сторінок за запитом з клієнтського комп'ютера. Тобто є потреба надати інформацію серверу про те, який користувач надіслав запит. Реалізовувати парольний доступ для кожного запиту незручно і недоцільно. Роль засобу збереження цілісності сеансу в цьому випадку відіграють ***кукі*** (Cookie) – невеликий набір даних, які передаються однією програмою іншій. Після авторизації (початок сеансу) сервер залишає на комп'ютері відвідувача кукі для автентифікації клієнта, які надалі передається серверу назад разом із кожним запитом на перемикання сторінки до коректного завершення сеансу. Кукі є досить вживаним засобом, наприклад вони є компонентом найбільш загального методу автентифікації, використовуваного в X Window. Деякі кукі (наприклад, у протоколі HTTP) можуть мати цифровий підпис або можуть бути зашифровані, щоб зловмисники не могли підробити і передати їх відправнику для отримання несанкціонованого доступу.

Перехоплення інформації можливе не тільки безпосередньо на комп'ютері, на якому ця інформація зберігається, а й на шляху її доставки, оскільки сучасні мережі використовують ланки загального використання з обробкою пакетів на проміжних вузлах. Відповідно, встановлення шпигунського програмного забезпечення на довільному проміжному вузлі (який не контролює адміністратор від організації, яка є володарем відправленої інформації) також дозволяє скопіювати дані. Так само можливе і перехоплення автентифікаційної інформації при віддаленому доступі.

Копіювання даних можна реалізувати і відгалуженням сигналу, що є цілком можливим при використанні багатьох ліній передачі, які застосовуються у сучасних мережах. Найменш захищеною з ліній передачі є такий розповсюджений (поки що) кабель як вита пара. Цей кабель в звичайному виконанні не має екрана, через що без порушення його цілісності поруч може бути встановлений ретранслятор. У випадках, коли цьому треба запобігти, застосовується екранована вита пара, але і в цьому випадку можливе перехоплення сигналу, хоча і ускладнене потребою на деякий час безпосереднього доступу до кабелю для порушення цілісності екрана. Значно краще захищеною лінією є оптоволокно, для якого створення відгалуження вимагає на деякий час, хоч і досить невеликий, порушити фізичну цілісність лінії, що за потреби може бути відстежень. Крім дротових

ліній зараз часто використовуються бездротові технології, які взагалі не передбачають захисту від можливості несанкціонованого отримання сигналу-носія.

Відповідно, мережеві застосунки повинні використовувати інший спосіб захисту інформації, який орієнтується не на унеможливлення доступу до даних, а на максимальне ускладнення (в ідеалі – неможливість) несанкціонованої інтерпретації цих даних.

Є ще один досить цікавий спосіб запобігання несанкціонованому доступу до даних. Цей спосіб має свого «доком'ютерного» аналога. На початку цього параграфу згадувався як спосіб захисту тайник. І однією з реалізацій тайника для блоку корисних даних є інший документ. При використанні паперового носія для цього використовували спеціальні чорнила, які стають видимими тільки за умови особливого зовнішнього впливу, наприклад хімічної або теплової обробки.

6.2.3. Стеганографія

Технологія приховування інформації отримала назву ***стеганографія*** – (з грец. στεγανός — прихований + γράφω — пишу), тобто тайнопис. В даному випадку захист забезпечується таким чином, що повідомлення маскується під інше (непотрібне зловмиснику) або взагалі інший вид інформації, наприклад, зображення. Таким чином, непосвячена людина не знає про факт існування даного повідомлення і не витрачає ресурси на перевірку його наявності. Зрозуміло, що тайнопис повинен виконуватися так, щоб не було натяку на схований контент.

Реалізація стеганографії цифровими методами має багато спільних рис зі звичайним тайнописом. Інформація імплантується у контейнер, адресат отримує цей змінений контейнер та, використовуючи перетворення, зворотне до того, що використовувалось при імплантації повідомлення, виділяє його. Шпигун при цьому не помічає у контейнері ознак, які можуть підказувати на наявність імплантату. В загальному випадку стеганографія охоплює два напрямки, які розв'язуються приблизно однаковим чином:

- маскування процесу передачі інформації;
- маркування інформації для запобігання підробці або незаконному використанню.

В обох із цих випадків на початку стеганографічного процесу існують два потоки інформації:

- повідомлення (прихована корисна інформація або дані маркування);
- контейнер (блок даних, у який імплантується повідомлення або маркер).

При маскуванні повідомленням є корисна інформація, що приховується, а контейнер – деякий блок іншої інформації (в даному випадку без особливої користі для відправника та адресата), у який імплантується повідомлення. При маркуванні навпаки – контейнер є блоком корисної інформації, яку треба помітити з метою захисту, а повідомлення являє собою мітку, яку в ідеальному випадку неможливо прибрати, але легко виявити. При цьому способи реалізації імплантації можуть бути спільними, хоча специфічність зазначених задач може вимагати і деякої оптимізації під одну з них.

Спочатку розглянемо маскування. Секретним засобом є спосіб (процедура) імплантації повідомлення у контейнер. Ця процедура повинна бути доступною адресату (у вигляді відповідної програми) і не доступна шпигуну. Процедура імплантації повинна за можливістю забезпечувати:

- високу ефективність (співвідношення кількості схованої інформації до загального розміру контейнера);
- складність виявлення ознак наявності імплантату.

Ось тут криється перше ускладнення цифрової стеганографії від звичайного тайнопису. Уявімо собі ситуацію, що шляхом використання невидимих чорнил відправляється деякий відсоток повідомлень в рамках загального потоку іншої пошти. Для перевірки наявності повідомлення кожен з листів треба відкрити (бажано так, щоб його можна було повернути до попереднього стану), проявити (не завжди проявник є таким, що не впливає на звичайні чорнила або папір). Така задача є надто ресурсоємною та технічно складною, що і запобігає подібним спробам. У разі цифрових технологій, оскільки обробці піддається копія досліджуваного документа, сліди зламу, майже обов'язкові у випадку паперових технологій, взагалі не виникають. Автоматизм програмного засобу та швидкодія сучасних обчислювальних систем дозволяють при малій ресурсоємності процедури перевірки наявності повідомлення (тобто при відомій процедурі імплантації) швидко обробити величезні потоки даних.

Оскільки реалізувати програмно надвелику кількість різних процедур (окремо для кожної можливої пари відправник–адресат) неможливо, в даному випадку (і у наступному параграфі при розгляді криптографії) ***є необхідним розділити процедуру обробки потоку даних на***:

- ***метод, як відкриту його частину;***

- ***ключ, як закриту варіативну частину процедури.***

В цьому випадку шпигуну (точніше програмному засобу шпигуна) доведеться виконати перебір методів імплантації та варіювання ключа. Загалом така перевірка повинна включати наступну реалізацію пари метод–ключ, обробку контейнера за такою парою, виділення ознаки наявності повідомлення. У такому варіанті вже можна говорити про забезпечення досить великої ресурсоємності подібної перевірки усієї придатної для стеганографії інформації, що проходить, наприклад, через деякий вузол мережі.

За типом інформації контейнер повинен давати змогу імплантації додаткових даних, тобто така імплантація не повинна порушувати основні властивості контейнеру, вибір якого є досить принциповим. Свого часу для приховування інформації розглядалось використання деякої надлишковості окремих комп'ютерних технологій роботи з даними (ця ідея отримала назву «комп'ютерна стеганографія»). Її основною ідеєю було приховування даних в невикористаних областях форматів файлів (наприклад, у метаданих), невикористаних місцях дисків (наприклад, незайнятих основними даними ділянках наприкінці кластерів диску) тощо. Але певна прив'язка до «стандартних» технологій з одного боку спрощує перевірку наявності схованих даних, а з іншого боку не забезпечує високу ефективність.

Щоб важче було виявити наявність імплантату, контейнер повинен максимально зберігати початкову функціональність, тобто на зображенні не повинні з'явитись артефакти, звук не повинен отримати додаткові помітні на слух завади, виконуваний файл (якщо гіпотетично використати такий дивний контейнер), повинен виконувати звичайні дії. Тобто цифрові технології в призводять до певних обмежень у виборі контейнера, але вони надають і величезні можливості, які розглянемо незабаром. Крім того, контейнер повинен обиратись таким чином, щоб шпигун не мав змоги порівняти його з «чистим» оригіналом, а з іншого боку щоб не виникав сумнів у потребі його зберігання або відправки через мережу. Для того, щоб розібратись, як краще обирати контейнер, порівняємо такі варіанти:

- виконуваний файл (програми);
- текстове повідомлення одного зі стандартних форматів;
- звуковий файл;
- зображення;
- відеопотік.

У першому випадку спеціальним чином треба розробити програму з «рихлим» кодом і скомпілювати її. Вимоги до програми включають наявну реальну функціональність та фрагменти

невикористаного коду (наприклад невикористаний при роботі програми текст або інші вкомпільовані дані), для якого треба створити іншу програму, яка буде його знаходити і заміняти на повідомлення. Все це виходить дуже складним, ненадійним та малоефективним.

Текстовий файл є компактним форматом, тіло якого складається з елементів, що відтворюються (літер) та елементів завдання способів їх відображення (розмітки). Загалом, деякі формати мають вбудовані структури, які формально можна підмінити, але виникає майже та ж проблема, що і в попередньому випадку.

Три інші типи даних мають суттєві переваги, оскільки значну частину їх об'єму складають дані, які відповідають сигналам з реального світу, тобто вводяться до комп'ютера з деякою похибкою та викривленнями, конкретні параметри яких не відомі шпигуну. Більш того, для створення контейнера можна навіть безпосередньо перед відправкою переробити деяким чином файл із мультимедійною інформацією (наприклад, зменшити роздільну здатність взятого з загального джерела зображення, замінити втратний формат стиснення на безвтратний тощо). Це є можливим завдяки тому, що адресату для вилучення імплантату немає потреби порівнювати отриманий контейнер з оригіналом. Більш того, наявність вільного контейнера у загальному доступі взагалі є небажаним саме через можливість порівняння з оригіналом. Відповідно, як контейнер краще використати щойно отриману фотографію улюбленої кицьки (особливо, якщо при пересиланні супроводити відповідною фразою), ніж взяту в Інтернеті репродукцію Джоконди.

Таким чином, тип контейнера обраний, залишилось розібратись, яким чином імплантувати секретне повідомлення. Найпростішим методом імплантації є метод LSB (Least Significant Bit, найменший значущий біт). Сутність методу полягає у тому, що для мультимедійного контейнера виконується заміна молодшого біту відліків на наступний біт з потоку імплантату. З врахуванням того, що значення цих відліків зазвичай мають шуми на рівні останнього біта, фактично замість шуму вноситься інший сигнал, близький до шумоподібного, що практично не змінює загальні характеристики контейнера. При цьому може бути досягнута ефективність, близька до 12 % (один біт із восьми несе корисну інформацію). Зауважимо, що прагнути такої високої ефективності без додаткових засобів захисту небажано, оскільки в даному методі немає варіативності.

Відповідно, реально застосовувані методи на основі LSB додають спосіб розміщення бітів в матриці контейнера з можливістю настроювання. Для цього може використовуватись генератор псевдовипадкової послідовності з використанням значення початкової

ініціалізації генератора в якості ключа. При цьому можуть замінюватись молодші біти не кожного відліку або може використовуватись їх заповнення не у порядку слідування байтів контейнера, наприклад пік селів зображення. Може використовуватись і шифрування повідомлення деяким доволі простим шифром, який в даному випадку має задачу не захисту, а додаткового приховування присутності змісту при вилученні його з контейнеру. Для збільшення надійності приховування може використовуватись також попередній аналіз зображення та його фільтрація при підготовці до імплантації.

Можуть використовуватись і більш витончені алгоритми, наприклад, «врізання» інформації у високі просторові частоти блоків при JPEG-кодуванні. Іншим варіантом кодування є Ехо-методи (використовують для кодування нерівномірні проміжки між ехо-сигналами), такі методи є стійкими до амплітудних і частотних атак, але нестійкі до атак за часом. Фазове кодування використовує заміну вихідного звукового елемента на відносну фазу, яка і є секретним повідомленням (цей метод є одним з найефективніших методів приховування інформації). Дуже потужним є метод розширеного спектра (схожий на широкосмужний зв'язок, який використовують сучасні системи стільникового зв'язку). Метод полягає у тому, що використовується спеціальна псевдовипадкова послідовність для внесення інформації та узгоджений фільтр для її вилучення (детектування).

Задачею маркування інформації є не захист від отримання інформації зловмисником, а доведення незаконного використання інформації іншою особою, наприклад порушення авторського права. Це може бути виконане, якщо на копії використаного незаконним чином зображення можна проявити маркер. На відміну від маскування інформації, коли контейнер часто досить помітно викривляється у процесі його підготовки, у випадку маркування пріоритетним щодо збереження інформації є саме контейнер. Звичайно, так чи інакше маркування вносить деякі втрати, але вони повинні бути прийнятними за критеріями, аналогічними до задачі втратного стиснення. Зрозуміло, що повинні бути враховані вид інформації, що маркується, та конкретних задач її використання. При маркуванні є бажаним забезпечити збереження маркування при використання деякої обробки маркованого зображення крадієм, наприклад, використання його фрагмента або масштабування. Виходячи зі специфіки використання та широковживаних методів обробки графічної та звукової інформації, ідеальною є система міток, яка як мінімум частково

зберігається при використанні інформації в іншому форматі, зокрема, при використанні втратних методів стиснення.

Найпростішим способом маркування може бути багаторазова періодична імплантація в тіло контейнера деякого текстового повідомлення, яке і можна буде вилучити навіть із фрагмента. Але вже при масштабуванні такий спосіб може давати хиби. Тому частіше використовується створення в тілі зображення так званого цифрового водяного знака, названого знову за аналогією з відповідною «докомп'ютерною» технологією. Традиційно водяний знак – зображення на папері (лінії різної форми, букви або монограми), яке виглядає світліше на просвіт або темніше у відбитому світлі. Він формується при виготовленні паперу вдавлюванням металевого сітчастого валика. За аналогією цифровий водяний знак зазвичай виконують як бінарний (глибина пікселя – 1 біт) логотип, який матрицюється на все поле зображення, що маркується. Цей логотип не повинен бути видимим звичайним чином, але повинен проявлятись спеціальною програмою. Проявлення може бути суттєво втратним, достатньо забезпечити візуальне розпізнавання проявленого логотипу. Як і при маскуванні повідомлення, при маркуванні контейнера можуть використовуватись ті ж технології. Найбільш розповсюдженими є нанесення маркування:

- в просторовому домені (зображення);
- в часовому домені (звук);
- в частотному домені (звук та зображення).

І в даному випадку найпростішим є метод LSB, при якому біти логотипу з деяким проріджанням вносяться в молодший біт пікселів контейнера. Крім цього можна використовувати додавання деякого шумового сигналу, статистичні властивості якого будуть мати просторовий (для зображення) або часовий (для звуку) розподіл, який відтворює логотип. Цифровий водяний знак може бути також варіюванням деякого параметра текстури синтезованого зображення.

Слід зауважити, що для захисту авторських прав маркування є більш надійним для забезпечення автентичності, ніж система електронних підписів (буде розглянута у п. 6.2.5), особливо якщо взяти до уваги можливість редагування мультимедійної інформації. Це пояснюється тим, що зловмисник використовує саме інформаційну частину і може просто вилучити редактором блок, де знаходиться кодований підпис.

6.2.4. Криптографія

З широким поширенням мережевих технологій швидко зростає потреба у забезпеченні конфіденційності помітної частини інформаційного трафіку і з часом ця частка буде зростати. Якщо в «докомп'ютерну» еру секретність повідомлень асоціювалась зі «шпигунськими іграми» на рівні державної та комерційної таємниці, зараз майже усі користувачі комп'ютерів так чи інакше використовують подібні засоби, навіть іноді не здогадуючись про це. Для передачі конфіденційного повідомлення може використовуватись:

- канал, недоступний іншим абонентам (цей шлях розглянутий у п. 6.2.2);
- загальні засоби зв'язку, але використовується маскування самого факту передачі інформації, цим питанням займається стеганографія (п. 6.2.3);
- загальні канали, але з передачею інформації у вигляді, який правильно інтерпретувати може тільки адресат, тобто шифрування інформації.

Перший спосіб є складним технічно та організаційно, другий спосіб є прийнятним тільки у випадку потреби захисту невеликого відсотка трафіку і в деяких випадках є неприйнятим взагалі. Розглянемо прийнятність третього підходу. Уявімо собі комічну ситуацію, що сеанс починається з відправки деякої фотографії. Оскільки першим етапом сеансу зазвичай є авторизація, це провокує шпигуна шукати пароль саме у цій фотографії. Крім того, загалом ефективність стеганографії сягає декількох відсотків. Все це робить використання цього методу захисту прийнятним при досить нечастому застосуванні і не може вдовольнити усіх потреб захисту інформації.

Таким чином, основним шляхом забезпечення захисту повідомлення є таке його кодування, яке робить неможливим (або, як мінімум, дуже ресурсоємним) декодування без володіння наперед передбаченим інструментом декодування. Цей спосіб отримав назву шифрування, а за розробку та використання його методичних особливостей відповідає криптологія (від грецького κρυπτός – прихований, скритний і λόγος – слово) – наука, що вивчає методи шифрування і дешифрування інформації. Вона включає в себе два розділи:

- криптографію;
- криптоаналіз.

Ці дві складові постійно знаходяться у стані протиборства, що призводить до бурхливого розвитку методів та засобів. Криптографія займається розробкою шифрування даних, у той час як криптоаналіз займається оцінкою сильних і слабких сторін методів шифрування, а також розробкою методів, які дозволяють зламувати криптосистеми. Зрозуміло, що для зламу шифру потрібно знати основи самого шифрування, а при створенні нових засобів захисту треба орієнтуватись на те, якими методами може бути зламаний цей захист.

У загальному випадку схема ***криптозахисту*** полягає у тому, що ***використовується деякий засіб перекодування, який на основі обробки (вона і називається шифруванням) початкового (відкритого) повідомлення створює такий блок даних (він називається шифроповідомленням), який (в ідеалі) може бути повернений до первинного виду (розшифроване повідомлення) тільки використанням засобів, передбачених при шифруванні***.

Шифрування загалом може включати деякі спеціальні апаратні засоби та методи. Тобто ***під шифром розуміється сукупність методів і технічних засобів шифрування***. Для забезпечення захищеного каналу між довільною парою абонентів з множини усіх абонентів, кожна пара повинна мати свій окремий засіб шифрування. З врахуванням того, що варіюванням апаратних та методологічних засобів це неможливо (N абонентів, виходячи з принципу «кожен з кожним» потребують N! варіантів), є потреба виділити з апаратного або методичного забезпечення ***варіативну частину – ключ***, який може створюватись для кожного каналу окремо, крім того, він є легко замінюваним через деякий час, навіть в межах одного каналу.

Криптологія – одна з найстаріших наук, її історія налічує кілька тисяч років. Перші методи захисту були пов’язані з механічними способами перетворення повідомлення. Зрозуміло, що ці методи в більшості своїй спирались на літерну дискретність традиційного способу запису повідомлення на папір. І досить довго основою була заміна однієї системи символів на іншу. Оскільки шифрування виконувалось виключно людиною, алгоритми перекодування передбачали відносно прості дії (використання таблиці шифрування, яка і відігравала роль ключа). Значно пізніше (вже у XX сторіччі) до процесу шифрування були залучені механічні, а згодом і електронні пристрої, які дозволили використати значно складніші та ресурсоємніші алгоритми.

Але тільки цифрові методи роботи з інформацією дозволили по-справжньому відірватись від прив’язки до літери, перейшовши на рівень бітового потоку, не розділеного на окремі літери у їх традиційному розумінні. З врахуванням специфіки сучасних

цифрових підходів має сенс дати більш вузьке визначення ***шифру як сукупності алгоритму секретного кодування цифрової (бінарної) інформації і ключа, який є сукупністю параметрів настроювання цього алгоритму.***

При використанні шифрування основною задачею є максимально можливим чином ускладнити роботу зламнику. ***Під зламом шифру розуміється процес отримання інформації, що захищається, без знання використаного шифру (ключа)***. ***Під стійкістю шифру розуміють спроможність протистояти зламу,*** тобто стійкість визначається сукупністю ресурсів (часу, кількості необхідних комп'ютерів, людських ресурсів тощо), які необхідні для його зламу. До основних методів зламу можна віднести:

- аналіз шифрованого потоку з метою визначення алгоритму шифрування;
- підбір ключа;
- статистичний аналіз шифрованого потоку для попереднього визначення характеристик ключа;
- розрахунок ключа на основі шифрованого потоку.

Оскільки методів шифрування існує досить багато, першою задачею криптоаналітика є визначення саме метода. З врахуванням того, що різні алгоритми шифрування (зрозуміло, що з даного моменту ми обмежуємось саме цифровими методами) можуть призводити до певних особливостей потоку шифроповідомлення, аналіз окремих його ознак може надати таку інформацію. Зазначимо, що при широкому використанні певного алгоритму шифрування для окремої задачі (наприклад, у певному протоколі або в певній інформаційній системі) метод шифрування можна визначити саме за ознакою відповідності повідомлення протоколу або системі.

При відомому методі весь захист покладається на ключ. Тривіальним і найбільш простим алгоритмічно є саме підбір ключа, який зводиться до послідовної генерації різних реалізацій ключа (аналогічно підбору пароля, п. 6.2.4). Але просто згенерувати ключ замало, для кожного екземпляра ключа треба виконати спробу дешифрування та застосувати деякий аналіз результату на предмет успішності дії. Зрозуміло, що ймовірний час підбору прямим чином залежить від очікуваної кількості переборів, тобто від довжини ключа. Але загальна ресурсоємність визначається ще ресурсоємністю аналізу результату (залежить виключно від того, що саме шифрується) та ресурсоємністю алгоритму дешифрування. Таким чином, надійнішим є більш повільний (складніший) алгоритм. Відповідно, є сенс використовувати для шифрування певні математичні функції, які за своєї природи є повільними у розрахунках. Зауважимо, що для

запобігання використанню табулювання таких функцій з метою пришвидшення може використовуватись надвеликий діапазон вхідного параметра, що може бути досягнуто розглядом бінарного потоку як одне наддовге ціле значення.

Крім підбору ключа можуть робитись спроби статистичного аналізу шифрованого потоку для попереднього визначення характеристик ключа. В даному випадку прив'язка дискретності повідомлення до літери (або коду окремої літери) дозволяє використати частотний аналіз, який спирається на властивість мови, що кожна з літер має певну частоту використання. Як правило, частотний аналіз не дає можливості повного відкриття ключа, але часткове відкриття ключа, який воно забезпечує, дозволяє додати рівень аналізу лексем даної мови (характерних сполучень літер та слів). Зрозуміло, що чим довшим є повідомлення, тим точніше може бути виконаний частотний аналіз.

Відповідно, засобами боротьби з застосуванням частотного аналізу може біти відмова від вирівнювання на літеру, чому сприяє саме використання цифрового кодування літер у сучасних комп'ютерних технологіях. Для цього як «умовна літера» може використовуватись їх пара, що значною мірою розширює розмір формального алфавіту. Можна навіть не використовувати вирівнювання на байт і розглядати як таку «умовну літеру», наприклад, 10 бітів. Завдяки цьому границя коду крім поточної «справжньої» літери чіпляє частину наступної, і частотний аналіз стає непридатним.

Розрахунок ключа на основі шифрованого потоку включає у себе інші методи математичної обробки шифропотоку, виходячи зі знання алгоритму. Це питання є дуже складним з точки зору математики і далеко неоднозначним, тому в рамках даного посібника докладніше не розглядається.

Надійність шифру завжди була основним параметром, за який боролися криптологи, навіть жертвуючи заради надійності швидкістю штатного процесу шифрування–дешифрування. При цьому періодично з'являються повідомлення про нову «абсолютно надійну» систему шифрування, яка все одно з часом виявляється зламаною (хоча і з застосуванням дуже великих ресурсів). Але є спосіб шифрування, який є абсолютно надійним. Цей спосіб витікає з теореми Шенона.

Стрічка однократного використання, тобто використання в якості ключа потоку випадкових значень тієї ж довжини, що і текст, що шифрується, не підлягає зламу. В даному випадку шифрування кожного елемента повідомлення відбувається новим елементом

ключа. Таким чином, якщо взяти довільний (який ми забажаємо отримати) елемент розшифрованого повідомлення з відповідного вхідного для даного алгоритму кодування, можна розрахувати елемент ключа. Розв'язок задачі стає гарантовано багатозначним, що і забезпечує абсолютну надійність. При цьому взагалі немає різниці, який саме метод шифрування використовується, це жодним чином не впливає на надійність. Єдиною проблемою практичного використання такого метода шифрування є саме одноразовість використання ключа, тобто адресат наперед повинен отримати ключ відповідної довжини іншими шляхами, ніж канал, який захищається шифруванням.

В інших випадках ключ має обмеження за довжиною і його використання повторюється з наступним фрагментом (блоком) повідомлення. Довжина цього блока визначається довжиною ключа. При надто довгому використанні ключа криптоаналітик може накопичувати статистику для кожного елемента ключа, що може бути використане для зламу. Відповідно, збільшення надійності шифрування передбачає якомога більш часту заміну ключа.

Тепер розглянемо основні методи, які використовуються в рамках цифрових методів обробки даних.

Загалом системи шифрування можна поділити на

- симетричні:
- асиметричні.

Симетричним є спосіб, при якому той самий метод із тим самим ключем використовується як для шифрування, так і для дешифрування. При несиметричному шифруванні методи або ключі для шифрування та дешифрування є різними, тобто той, хто закодував повідомлення, не може його декодувати.

Для реалізації симетричного шифрування можуть бути використані:

- функціональні шифри;
- шифри заміни;
- шифри перестановок.

Прикладом простого (але ненадійного) **функціонального шифру** є використання булевої операції XOR (виключної диз'юнкції), табл. 6.2.

Особливістю цієї операції є те, що $(a \oplus b) \oplus b = a$, тобто якщо b – елемент ключа, вона виконує як шифрування, так і дешифрування.

Шифр заміни, який зводиться до встановлення деякої таблиці співвідношення відкритих і шифрованих даних. Така таблиця і є ключем. Замість готової таблиці може використовуватись деякий варіабельний алгоритм формування такої таблиці, параметр початкового настроювання якого є ключем. Проста підстановка

відповідає ситуації, коли кожній літері вхідного алфавіту відповідає лише одна літера вихідного (шифрованого) алфавіту. Головним недоліком простого шифру заміни є невиконання умови протидії використанню частотного та лексемного аналізу, що дозволяє використанням комп'ютерних засобів достатньо легко його розкривати. Для ускладнення може використовуватись диграмна або триграмна підстановка, коли як літера при кодуванні розглядається сукупність двох-трьох літер. На сучасному рівні метод підстановки часто спирається на матричну алгебру, коли ключ являє собою матрицю, яка помножується на вектор повідомлення. Для дешифрування використовується обернена матриця.

Таблиця 6.2. Таблиця істинності XOR

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

Варіантом підстановки (досить ненадійним) є гамірування, при якому символ, що кодується, додається до символу ключа за модулем, рівним числу літер в алфавіті. Такий метод кодування, наприклад, використовувався в текстовому редакторі Microsoft Word версій 6-7 при парольному записі файлу.

Шифри перестановок засновані на зміні порядку слідування знаків у потоці. Ключем у цьому випадку є алгоритм перестановок або таблиця відповідності положення знаків шифрованого і відкритого повідомлень. Проста транспозиція обмежується діленням потоку на фрагменти і використанням однакової циклічної перестановки для кожного фрагмента. Для ускладнення може використовуватись складна транспозиція – послідовне виконання двох транспозицій з різним періодом (розміром фрагмента).

Замість перестановки порядку слідування символів (байтів) може використовуватись циклічна перестановка бітів (що добре реалізується апаратно), тобто потік даних розглядається як деякий бітовий потік без вирівнювання на розмір байта.

Першою вдалою реалізацією відкритого шифрування для надзвичайно широкого застосування була система DES (Data Encrypting Standard), запропонована IBM у 1974 році. Ця система використовує досить складний алгоритм роботи і ключ довжиною 56 бітів, що забезпечує $2^{56} = 10^{17}$ комбінацій, що з точки зору швидкодії

обчислювальних засобів того часу було цілком достатнім. При кодуванні потік даних розбивається на блоки розміром 64 біти, які незалежно кодуються вказаним ключем (режим кодової книги). Тобто фактично використовується підстановка в алфавіті з 264 літер. У 1978 році на основі DES було прийнято стандарт ISO8372.

Асиметричне шифрування (інакше називається система з відкритим ключем) використовує в основі алгоритму так звані односторонні математичні функції. Особливістю цих функцій є велика різниця у розрахунку прямої та зворотної функції, наприклад, експоненти та логарифма. При цьому для більшого ускладнення обрахунку зворотної функції весь потік даних розглядається як єдине наддовге ціле число. Такий рівень асиметрії, що призводить практично до неможливості за розумний час виконати розрахунок, і визначив цю назву. В такому варіанті односторонні функції є непридатними для шифрування, хоча, наприклад, можуть бути використані для шифрування паролів (п. 6.2.4). Якщо зберігати паролі у зашифрованому таким чином вигляді, крадій буде мати достатньо великі проблеми зі зворотним розрахунком, а система при автентифікації зможе перевірити результат шифрування щойно введеного пароля з результатом аналогічного шифрування, що зберігається в системі. Для криптографії використовуються односторонні функції з лазівкою, тобто з деяким секретом, який допомагає виконати розшифрування за прийнятний час.

Ідея асиметричного шифрування належить Ральфу Меркле і отримала значний розвиток в роботі Уитфилда Діффі і Мартіна Хеллмана «Нові напрямки в сучасній криптографії» (1976 р.), результатом чого була поява метода, відомого як обмін ключами Діффі–Хеллмана. Цей алгоритм був першим опублікованим практичним методом для встановлення поділу секретного ключа між завіреними користувачами каналу. У 2002 році Хеллман запропонував його називати алгоритмом Діффі–Хеллмана–Меркле. У 1977 році Рональд Рівест, Аді Шамір і Леонард Адлеман (Массачусетський технологічний інститут) запропонували для реалізації асиметрії обрахунків використовувати складність факторизації добутку двох великих простих чисел. Для шифрування використовується зведення у степінь за модулем великого числа. Щоб дешифрувати за розумний час (зворотна операція), необхідно вміти обчислювати функцію Ейлера від даного великого числа, для чого необхідно знати розкладання числа на прості множники, що і є «лазівкою». Практична реалізація такого алгоритму отримала назву RSA.

У сучасних умовах асиметричне шифрування є дуже розповсюдженим методом. Це пов’язано з тим, що в цьому випадку

зникає потреба передачі ключа шляхом, не пов'язаним з шифроканалом, що створюється.

Загалом, в рамках системи з відкритим ключем створюється деяка організаційна структура для реєстрації користувачів з можливістю публікації так званого відкритого ключа кожного з користувачів. Кожен учасник системи має у своєму розпорядженні закритий ключ (тримається у секреті) та відкритий ключ (публікується у системі). Відкритий та закритий ключі абонента створюють «узгоджену пару» в тому сенсі, що є взаємозворотніми. Таким чином, використання для шифрування власного закритого ключа з відкритим ключем адресата (для розшифровки адресат використовує власний закритий ключ з відкритим відправника) створюють окремий шифроканал.

Схожим чином (за асиметричною схемою) реалізується шифрування у досить популярному в технологіях Інтернету протоколі SSL (Secure Sockets Layer — рівень захищених сокетів). Цей протокол було розроблено Netscape Communications, пізніше стандартизований (актуальне оновлення стандарту – RFC 5246.). В даному випадку ключі (сертифікати) не публікуються, а генеруються для кожного сеансу окремо з цільовою передачею відкритого ключа на інший бік, який відповідає даному сеансу.

6.2.5. Електронний цифровий підпис

В рамках інформаційного захисту є ще одна дуже важлива задача, яка на основі цифрових технологій вирішується досить специфічним чином – забезпечення автентичності документа. Загалом, ***автентифікація документа є операцією перевірки дійсної належності автору і відсутності змін, які могли з'явитись під час його транспортування або збереження***.

Протягом історії людства для засвідчення незмінності документа виробили певні правила, дотримання яких в основному цю задачу вирішувало. Основу захисту автентичності паперового документа складає засвідчення аркуша деяким формальним зображенням, яке вважається таким, що важко підробити. При цьому документ або не повинен мати виправлень, або усі виправлення повинні бути позначені та окремо засвідчені аналогічним чином. Роль такого зображення відіграють підписи і печатки. Перевагою підпису є належність окремій особі, перевагою печатки є краща повторюваність відбитка в порівнянні з деякою розбіжністю екземплярів підпису. Через це підробка значно ускладнюється, особливо, якщо зображення печатки є достатньо складним.

Цікаво, що за (поки що) діючими паперовими технологіями у дуже багатьох випадках багатосторінковий паперовий документ затверджується печаткою і підписами традиційно на першій або останній сторінці, а ідентифікація інших сторінок перекладається на рівень співпадіння типу паперу, якості тонеру, суміщення дірочок від степлера...

До зазначених способів захисту можуть додаватись спеціальні бланки типографського виготовлення з додатковими зображеннями складної структури, що є логічним продовженням скріплення автентичності саме зображенням. За потреби забезпечення більшої надійності усі зазначені варіанти скріплення автентичності об'єднуються.

Подібний графічний спосіб є далеким від ідеалу. Як підпис, так і печатку можна підробити, причому іноді підробки настільки якісні, що навіть ретельна експертиза не дає 100-відсоткової надійності, особливо в разі підробки підпису. А в більшості випадків відповідність визначається не експертизою, а досить поверхневим оглядом. Сучасні цифрові засоби копіювання та автоматизованого виготовлення печаток надійність такого захисту взагалі звели нанівець, що призвело до потреби захисту паперових документів вводити значно складніші для копіювання елементи, наприклад, голограми.

Сучасне суспільство активно переходить з паперових технологій на електронний документообіг, що є дуже важливим, оскільки саме такий варіант взаємодії окремих осіб та підрозділів в умовах розподіленої організації або виробництва є єдиним можливим способом забезпечити необхідну для конкурентоспроможності динаміку керування. В таких умовах традиційні методи захисту автентичності взагалі стають неприйнятними. Факсимільна копія печатки або сканований підпис при відправці електронної копії паперового документа взагалі реально нічого не засвідчують. На відміну від паперового документа в електронному на рівні бітового доступу можна виправити все, включаючи такий атрибут файлу, як дата останньої модифікації. Взагалі, такі атрибути скоріше помітка для себе, а не захист від підробки. Не набагато складніше зробити несанкціоноване виправлення перехопленням або формування фальшивого документа при використанні віддаленого зв'язку.

Але цифрові технології обробки даних не тільки створили таку проблему, а й дозволили її досить ефективно розв'язати за рахунок технології електронного цифрового підпису (далі ЕЦП) – своєрідного аналогу за назвою, хоч і не за принципом (рис. 6.4).

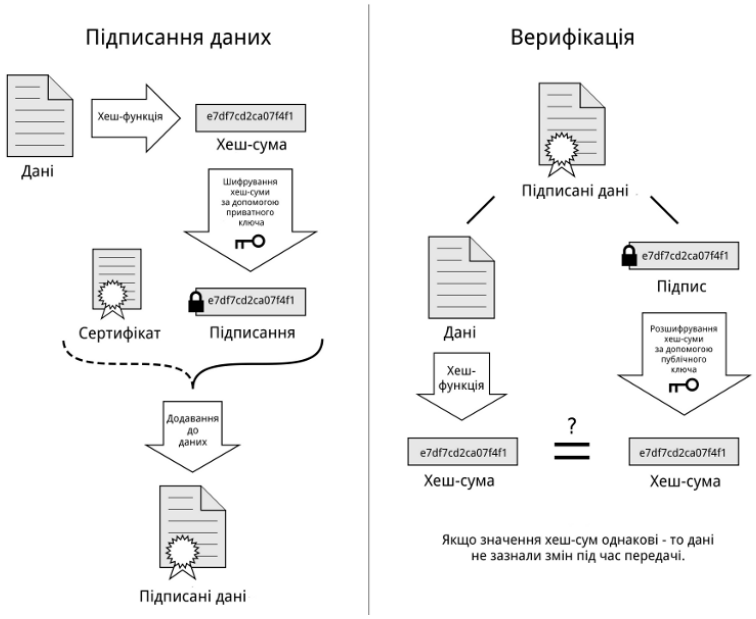


Рис. 6.4. Загальна схема використання електронного цифрового підпису

Накладання ЕЦП завершує утворення електронного документа, надаючи йому юридичної сили. Юридична сила електронного документа з нанесеними одним або множинними ЕЦП та допустимість такого документа як доказу не може заперечуватися виключно на підставі того, що він має електронну форму (ст. 8 Закону України «Про електронні документи та електронний документообіг»). Послуги з надання ЕЦП в Україні впроваджуються акредитованими центрами сертифікації ключів, перелік яких публікується на сайті Центрального засвідчувального органа.

За правовим статусом ЕЦП прирівнюється до власноручного підпису або печатки. За умови правильного зберігання власником секретного (особистого) ключа його підробка неможлива. Електронний документ також неможливо підробити: будь-які зміни, несанкціоновано внесені в текст документа, будуть виявлені. Особистий ключ ЕЦП є унікальною послідовністю символів довжиною 264 біта, яка формується на основі використання генератора випадкових чисел, а відкритий ключ обчислюється з особистого ключа відповідно до принципів асиметричного шифрування.

Сертифікат відкритого ключа містить в собі персональну інформацію про його власника (ім'я, реквізити), унікальний реєстраційний номер, термін дії Сертифікату. З метою забезпечення цілісності представлених у Сертифікаті даних він підписується особистим ключем Центру сертифікації ключів. Сертифікат

відкритого ключа може публікуватися на сайті відповідного ЦСК відповідно до Договору про надання послуг ЕЦП.

При підписанні електронного документа його початковий зміст не змінюється, а додається блок даних – так званий Електронний цифровий підпис. Отримання цього блоку можна розділити на два етапи.

Для закріплення автентичності новоствореного документа за допомогою спеціальної математичної функції (так званої хеш-функції) обчислюється «відбиток повідомлення» (message digest), який має такі властивості:

- фіксовану довжину, яка не залежить від довжини повідомлення;
- унікальність відбитку для кожного повідомлення;
- неможливість відновлення повідомлення за його відбитком.

Під час перевірки автентичності документа значення хеш-функції обчислюється за тим самим алгоритмом, що і при створенні відбитку. Тобто будь-яка зміна даних, що утворюють документ, буде виявлена. Далі відбиток документа шифрується використанням особистого ключа автора.

Розшифрувати ЕЦП і одержати відбиток, який відповідатиме документу, можна тільки використовуючи Сертифікат відкритого ключа автора, що і є захистом від модифікації сторонніми особами та авторства.

6.3. Питання для самоконтролю

1. Чому терміни «база даних» та «інформаційна система» не є синонімами?
2. Чому режим файлового сервера є неприйнятним для корпоративної інформаційної системи?
3. Для яких задач і чому в сучасних умовах використовується архітектура системи «мейнфрейм-термінал»?
4. Чи є однозначним розподіл функціонального навантаження між «клієнтом» та «сервером»?
5. Чим «тонкий клієнт» відрізняється від терміналу?
6. Чи можуть бути в таблиці бази даних два окремих первинних ключа?
7. Чи можуть поля різних таблиць мати однакове ім’я?
8. Чи можуть існувати в різних таблицях однієї бази два однакових записи?

9. Чим відрізняються терміни «база даних» та «система керування базами даних»?
10. В яких випадках і для чого використовують зв’язування двох чи декількох таблиць?
11. Чому в разі відсутності підтримки транзакцій база даних може втрачати цілісність інформації?
12. Для яких цілей використовується «двигун баз даних» та драйвери до нього?
13. Чи є SQL мовою програмування?
14. Які критерії можна визначити як достатній рівень криптографічного захисту?
15. Чи є стеганографія повноцінним замінником шифрування?
16. Чи є абсолютно надійний спосіб шифрування?
17. У чому переваги використання асиметричного шифрування?
18. Яким чином зламники використовують віруси?
19. Чому первинне повідомлення не може бути відновлене зі значення його хеш-функції?
20. Чи можна реалізувати електронний цифровий підпис на основі симетричного шифрування?

Список літератури

До глави 1

1. Яшин В.Н. Информатика: аппаратные средства персонального компьютера. – М.: «ИНФРА-М», 2008. – 254 с.
2. Информатика. Базовый курс. / Под ред. С. В. Симоновича. – СПб.: «Питер», 2005. – 640 с.
3. Рудометов Е.А. Современное железо: настольные, мобильные и встраиваемые компьютеры. – СПб.: «БХВ-Петербург», 2010. – 464 с.

До глави 2

4. Тотосько О.В., Микитишин А.Г., Стухляк П.Д. Комп'ютерна графіка. – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2017. – 304 с.
5. Габибов И.А., Маликов Р.Х. Инженерная графика. – Баку: «АГНА», 2011. – 177 с.
6. Кидрук М.И. КОМПАС-3D V10 на 100 %. – СПб.: «Питер», 2009. – 115 с.
7. Бочков А.Л. Трехмерное моделирование в системе Компас-3D (практическое руководство). – СПб.: СПбГУ ИТМО, 2007. – 84 с.
8. Афанасьев В.А. National Instruments/MultiSim 10.1: быстрый старт. 2009. – 39 с.
9. Роджерс Д.Ф. Алгоритмические основы машинной графики. – М.: «Мир», 1989. – 511 с.
10. Севастьянов А.Г., Севастьянов П.А. Моделирование технологических процессов. – М.: «Легкая и пищевая промышленность», 1984. – 344 с.

До глави 3

11. Гонсалес Р., Вудс Р. Цифровая обработка изображений. – М.: «Техносфера», 2005. – 1072 с.
12. Прэтт У. Цифровая обработка изображений. кн.1. – М.: «Мир», 1982. – 312 с.

13. Прэтт У. Цифровая обработка изображений. кн.2. – М.: «Мир», 1982. – 480 с.
14. Ивашко А.В. Методы и алгоритмы цифровой обработки сигналов. Х.: НТУ “ХПИ”. – 2003. – 233с.
15. Фисенко В.Т., Фисенко Т.Ю., Компьютерная обработка и распознавание зображений. – СПб.: СПбГУ ИТМО, 2008. – 192 с.

До глави 4

16. Орнатский П.П. Автоматические измерения и приборы. – К.: «Вища школа», 1988. – 504 с.
17. Бутырин П., Васьковская Т. Автоматизация физических исследований и эксперимента: компьютерные измерения и виртуальные приборы. – «ДМК», 2014. – 256 с.
18. В. П. Колодийчук. Шаговые двигатели. – «Электротехнический рынок». 2007. – № 12 (18). – С.50-53.
19. Шандров Б. В. , Чудаков А. Д. Технические средства автоматизации. М.: Издательский центр «Академия», 2010. – 368 с.
20. Черевко О.І., Кіптела Л.В., Михайлов В.М., Загорулько О.Є. Автоматизація виробничих процесів. – Х.: Харківський держ.унів.харч., 2014. – 186 с.
21. Демиденко С.Н., Апанасенко Л.С., Дашук В.Н., Куновский Э.Б. Модульные КАМАК системы автоматизации эксперимента. – Минск: «Навука і технік», 1990. – 208 с.
22. Райс В. "Компоненты и технологии", № 3. – 2005.
23. Соломенчук В.Г., Соломенчук П.В. Железо ПК 2012 – СПб.: «БХВ-Петербург», 2012. – 384 с.
24. Таненбаум Э., Остин Т. Архитектура компьютера. СПб.: «Питер», 2013. – 816 с.
25. Матвієнко М.П., Розен В.П., Закладний О.М. Архітектура комп’ютера. – К.: «Ліра-К», 2016. – 264 с.
26. Матюшин А.Ю. Программирование микроконтроллеров: стратегия и тактика. – М.: «ДМК Пресс», 2017. – 356 с.
27. Ревич Ю.В., Практическое программирование микроконтроллеров Atmel AVR на языке ассемблера. – СПб.: «БХВ-Петербург», 2014. – 368 с.
28. Петин В.А. Проекты с использованием контроллера Arduino. – СПб.: «БХВ-Петербург», 2015. – 464 с.

До глави 5

29. Гук М. Аппаратные интерфейсы ПК. Энциклопедия. – СПб.: «Питер», 2002. – 528 с.
30. Олифер В.Н., Олифер Н.А.. Компьютерные сети. Принципы, технологии, протоколы. − С.П.: «Питер», 2003. − 864с.
31. Антонов В.М. Сучасні комп'ютерні мережі. – К.: «МК-Прес». – 2005.– 480с.
32. Грицунов О. В. Інформаційні системи та технології. – Х.: ХНАМГ, 2010. – 222 с.
33. Бьюли А. Изучаем SQL. – СПб.: «Символ», 2007. – 309 с.
34. Кузнецов М.В. MySQL 5. – СПб.: «БХВ-Петербург», 2010. – 1024 с.
35. Mell P., Grance T. The NIST Definition of Cloud Computing. Recommendations of the National Institute of Standards and Technology. – Special Publication 800-145.

До глави 6

36. Барабаш А.В., Баранова Е.К. Криптографические методы защиты информации. – М.: «КНОРУС», 2016. – 192 с.
37. Бабенко Т.В., Гулак Г.М., Сушко С.О., Фомичова Л.Я. Криптологія у прикладах, тестах і задачах. – Д.: Національний гірничий університет, 2013. – 318 с.
38. Корченко О.Г., Сіденко В.П., Дрейс Ю.О. Прикладна криптологія: системи шифрування. – К. : ДУТ, 2014. – 448 с.
39. Конахович Г., Прогонов Д., Пузыренко А. Комп'ютерна стеганографічна обробка й аналіз мультимедійних даних. – Центр навчальної літератури, 2018. – 560 с.
40. Домарев В.В. Безопасность информационных технологий. Системный подход – К.: ООО «ТИД Диасофт», 2004. – 992 с.